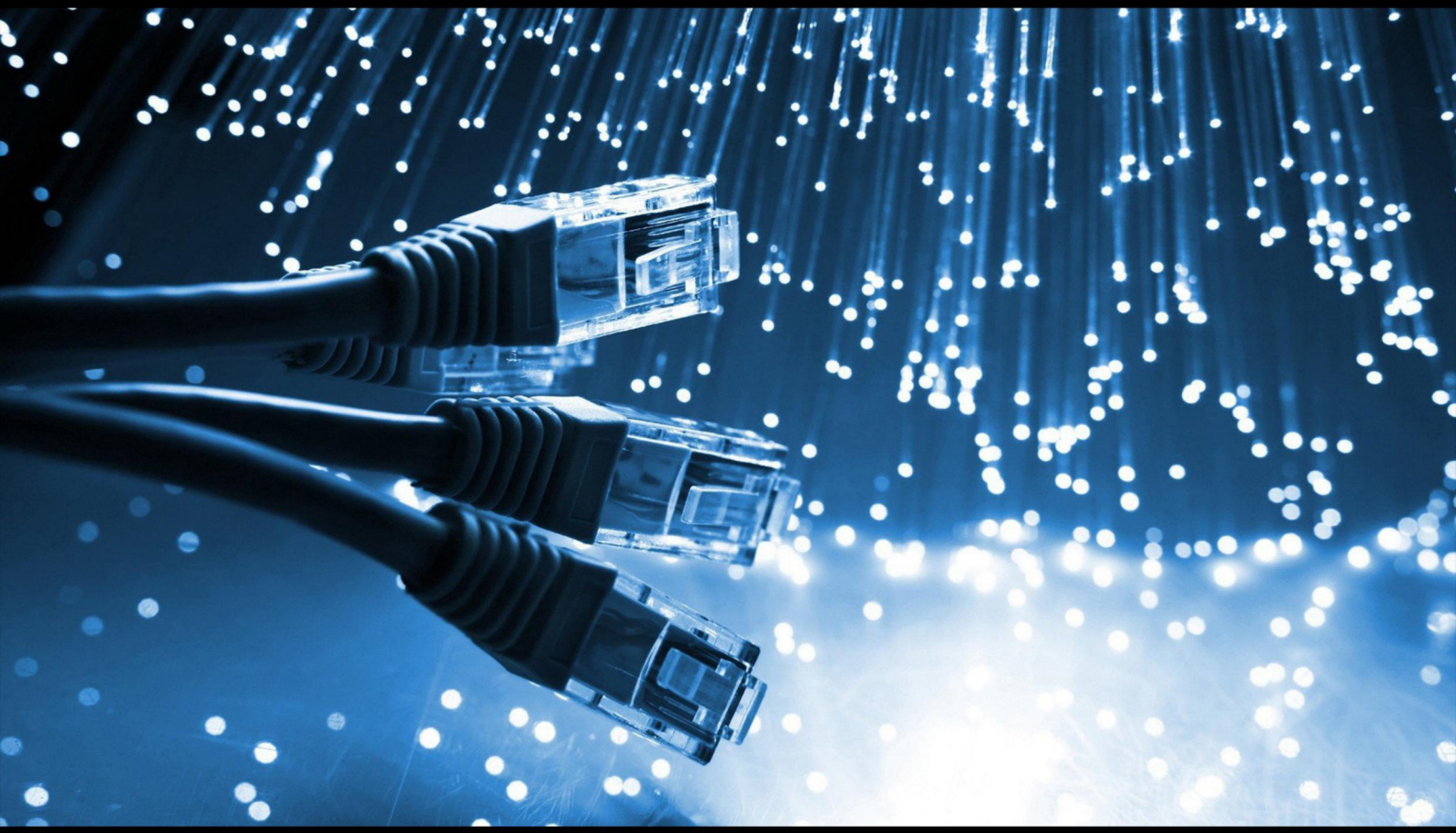
JCSI

Journal of Computer Sciences Institute

Volume 21/2021



Department of Computer Science
Lublin University of Technology

jcsi.pollub.pl

ISSN: 2544-0764

Redakcja JCSI
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Katedra Informatyki
Wydział Elektrotechniki i Informatyki

Politechnika Lubelska
ul. Nadbystrzycka 36 b
20-618 Lublin

JCSI Editorial
e-mail: jcsi@pollub.pl
www: jcsi.pollub.pl
Department of Computer Science
Faculty of Electrical Engineering and Computer Science
Lublin University of Technology
ul. Nadbystrzycka 36 b
20-618 Lublin, Poland

ISSN 2544-0764

Spis treści

Contents

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 259-278
Received: 06 October 2021
Accepted: 08 November 2021

Comparison of selected mathematical functions for the analysis of growth behavior of items and physical interpretation of Avrami–Weibull function

Porównanie wybranych funkcji matematycznych do analizy przebiegu wzrostu podmiotów oraz interpretacja funkcji Avrami'ego–Weibulla

Keshra Sangwal*
Department of Applied Physics, Lublin University of Technology, Nadbystrzycka 38, 20-618 Lublin, Poland

Abstract

Empirical data of sigmoidal-shaped $y(t)$ growth behavior of different types of items, such as papers and citations earned by individual and all successively published papers of selected top-cited authors, germination of tomato seeds and three different bacteria, are analyzed and compared by Avrami–Weibull, Verhulst (logistic) and Gompertz functions. It was found that: (1) Avrami–Weibull function describes different types of the data better than Gompertz and Verhulst funtions, and (2), in comparison with Verhulst and Gompertz functions, Avrami–Weibull function, expressed in the form: $y(t)/y_{max} = 1-\exp[(t/\Theta)^q]$ (where y_{max} is the maximum value of $y(t)$ when $t \to \infty$, and Θ and q are constants), is equally very versatile in explaining the generation rate $dy(t)/dt$ of items in terms of its parameters Θ and q. Using the basic concepts involved in the derivation of Avrami–Weibull function for overall crystallization from melt and supersaturated solution, the growth behavior of cumulative number $y(t)$ of items produced at time t by individual (simple) sources and collectives or groups of simple sources (i.e. complex or composite sources) is presented. Comparison of the process of receiving of citations by papers with the processes of occurrence of chemical reactions and crystallization of solid phases from melts and supersaturated solutions shows that this process is similar to that of overall crystallization of solid phases from melts and solutions. Analysis of growth of citations using Avrami–Weibull function to individual papers published by different authors shows that $1 < q < 4$ for most cases. This suggests that the process of citations to individual articles is mainly determined by progressive nucleation mode involving both diffusion and integration of published knowledge.

Keywords: Avrami–Weibull function; Gompertz and Verhulst functions; Growth behavior of items; Citation analysis

Streszczenie

Przeanalizowano i porównano stosowalność funkcji Avrami'ego–Weibulla, Verhulsta (logistycznej) i Gompertza do empirycznych danych sygmoidalnego przebiegu wzrostu $y(t)$ takich różnorodnych podmiotów jak: liczba artykułów i cytowań otrzymywanych przez pojedyncze i wszystkie kolejne artykuły publikowane przez wybranych wysoko cytowanych autorów, liczba kiełkowań nasion pomidorów i liczba trzech różnych bakterii. Zaobserwowano, że: 1) funkcja Avrami'ego–Weibulla opisuje różne dane lepiej niż funkcje Gompertza i Verhulsta, oraz 2) w porównaniu z funkcjami Verhulsta i Gompertza, funkcja Avrami'ego–Weibulla, wyrażona w postaci: $y(t)/y_{max} = 1-\exp[(t/\Theta)^q]$ (gdzie: y_{max} jest maksymalną wartością $y(t)$ gdy $t \to \infty$, oraz Θ i q są stałymi), jest równie wszechstronna w wyjaśnieniu szybkości wytwarzania $dy(t)/dt$ wyżej wymienionych podmiotów przy pomocy parametrów Θ i q. Korzystając z podstawowych pojęć zawartych w wyprowadzeniu równania Avrami'ego–Weibulla do opisania całkowitej krystalizacji z fazy roztopionej i z roztworu przesyconego, przedstawiono przebieg wzrostu kumulacyjnej liczby $y(t)$ podmiotów wytwarzanych w czasie t poprzez pojedyncze (proste) źródła i zbiory lub grupy pojedynczych źródeł (tj. złożonych źródeł). Porównanie procesu otrzymywania cytowań przez artykuły z procesami występowania reakcji chemicznych i krystalizacji ciał stałych ze stopów i roztworów przesyconych pokazuje, iż proces ten jest podobny do całkowitej krystalizacji ciał stałych ze stopów i roztworów. Analiza wzrostu cytowań według równania Avrami'ego–Weibulla pojedynczych artykułów publikowanych przez różnych autorów pokazuje, że w większości przypadków $1 < q < 4$. Z powyższego można wnioskować, że proces cytowania pojedynczych artykułów zachodzi w głównej mierze przez zarodkowanie progresywne oparte na dyfuzji i integracji opublikowanej wiedzy.

Słowa kluczowe: Funkcja Avrami'ego–Weibulla; Funkcje Gompertza i Verhulsta; Przebieg wzrostu podmiotów; Analiza cytowań

*Corresponding author (Emeritus Professor).
Email address: k.sangwal@pollub.pl (K. Sangwal)

1. Introduction

Since the World War II the scientific literature has witnessed steady growth of scientific research due to its increasing scope and volume. As measures of growth of the scientific literature one can mention the number of authors working in different scientific fields, the number of articles published in different fields, the number of journals published in different scientific disciplines, the number of papers published in individual journals, and the number of journals published in individual countries. Three types of growth of cumulative number $y(t)$ of items with time t are observed: (1) *normal* S-shaped plots with initial convex curvature followed by concave curvature, (2) *inverse* S-shaped plots with initial concave curvature followed by convex curvature, and (3) linear plots of $y(t)$ data. In order to describe the growth behavior of the above type of plots a variety of mathe-

matical functions involving empirical parameters have been developed and applied over years [1–8]. Among the different equations to describe the growth behavior of various phenomena, power-law, exponential and logistic (also called Verhulst's) functions are commonly used but Gompertz' function has also been reported [2].

In his classic work, Price [1] observed that in many cases the growth of science follows exponential dependence. In a detailed study devoted to the analysis of 20 different databases in humanities, social sciences, and science and technology (Sci-Tech), Egghe and Ravichandra Rao [2] found that the power-law function with an exponent greater than 1 is the best for Sci-Tech databases, whereas Gompertz' function, which is a generalization of exponential function, is the best to describe the growth of social sciences and humanities databases. Gupta et al. [4] applied selected functions to the data of cumulative growth of publications in six sub-disciplines of social sciences in the world and found that the power-law and logistic functions are the best to describe the growth of publications in the above sub-disciplines. Ravichandra Rao and Srivastava [5] analyzed the growth of journals, articles and authors in malaria research for the period 1955–2005 and found that the exponential function best fits the data. Wong and Goh [6] applied three competing growth functions, namely, simple logistic growth function, bi-logistic growth function and logistic function within a dynamic carrying capacity (LGDCC) to explain the growth behavior of the number of publications and patents of South Korea, Taiwan, Japan and Malaysia. These authors found that the bi-logistic growth function is the best for the number of publications as well as the number of patents of South Korea and Taiwan, the LGDCC function for the number of publications and the bi-logistic growth function for the number of patents of Japan, whereas the LGDCC for the number of publications and simple growth function for the number of patents of Malaysia.

Description of growth of a scientific phenomenon using mathematical functions is important for understanding their course in terms of the parameters of the functions and for predicting future trends. The values of the parameters of different functions are especially important for understanding the cause-and-effect relationship of the course of the phenomena. However, apart from fitting empirical data using various mathematical functions, sometimes erroneously called modeling [3,4], in the informetric literature there are several studies devoted to the statistical modeling of citation behavior of publications of individual authors [9-15] and to the theoretical development of statistical formulas for citation distribution [16,17]. In these models, the papers published by an author and the citations received by the papers are generated according to some statistical function. For example, in his stochastic model, Burrell [9-12] assumed that: (1) papers are published by an author according to a Poisson process at a constant rate, (2) citations to a paper are received according to a Poisson process at a constant rate, and (3) citation rate of the papers for the author varies according to gamma distribution. In order to describe the time dependence of growth of cumulative citations of papers published by an author, Hirsch [18] proposed a deterministic model based on stretched exponential with exponent $q \leq 1$ (see Eq. (13)). In the case of a researcher publishing a constant number of papers and each paper fetching a fixed number of citations per year every subsequent year, stochastic and deterministic models predict approximately quadratic growth of the total number of his/her citations with publication time [11,18].

Application of various mathematical functions is not confined to describe growth of scientific literature alone. For example, logistic function and its modifications, Gompertz function and its modifications, and standard Avrami–Weibull function have been used in diverse areas such as various bacterial growth in different media (for example, see: refs. [19-30]), population growth of individual biological species (for example, see: ref. [31]), growth of Tumor cells [32,33], germination of seeds [34], and overall crystallization of various compounds (for example, see: [35-37]). However, among these functions, Avrami–Weibull function has been applied for the data analysis in relatively few studies [19,24,27,36,38]. In these studies this function is referred to as Weibull function.

The aim of the present study is two-fold: (1) to compare best fits of data of the growth behavior of different types of items by Avrami–Weibull, Verhulst and Gompertz functions, and (2) to analyze the growth of cumulative papers $N(t)$ by an author and cumulative citations $L(t)$ of his/her individual as well as progressively published papers, with special emphasis on their sigmoidal-shaped growth, as a function of time t using Avrami–Weibull function. Since the two fitting parameters of the Avrami–Weibull function as used in overall crystallization have well-defined physical interpretation, an additional aim of the study is to demonstrate that the process of growth of scientific literature is similar to overall crystallization from melts and solutions but differs fundamentally from chemical reactions described by chemical kinetics. The general concepts of occurrence of chemical reactions and formation of crystallization nuclei are briefly presented.

In the paper the following topics are discussed: (1) presentation of standard Avrami–Weibull, Verholst and Gompertz equations for the growth of items with time, (2) analysis of growth of different phenomena using Avrami–Weibull, Verholst and Gompertz functions in order to compare best fit of the data and then using Avrami–Weibull function examination of the effect of external factors on the growth of these phenomena, (3) description of basic concepts involved in the generation of items and their growth as a function of time in the form of Avrami–Weibull function, (4) predictions of Avrami–Weibull function on the growth and growth rate behavior of citations of papers in relation to changes in its parameters Θ and q and their confrontation with the empirical data of the dependence of yearly citations of all successively published papers and four individual top-cited papers of two highly-cited authors, and (5)

summary of the main findings of the study. Moreover, in order to introduce the reader to the topics discussed in the paper rudiments of occurrence of chemical reactions and crystallization, basic concepts of overall crystallization and comparison of predictions of different functions are briefly described in Appendices A, B and C, respectively.

2. Relevant mathematical functions

In order to describe the S-shaped curves observed for different phenomena we selected Avrami–Weibull, Verhulst (logistic) and Gompertz functions relating the growth of cumulative number $y(t)$ of items with time t. According to the Avrami–Weibull function the relationship between $y(t)$ and t is given by (for example, see: refs. [27,39,40])

$$y(t) = y_{\max}\left\{1 - \exp\left[-\left(\frac{t}{\Theta}\right)^q\right]\right\}, \quad (1)$$

where $y_{\max}$ is the value of $y(t)$ when $t \to \infty$, and Θ and q are constants. The constants Θ and q are called the location and shape factors, respectively [27].

According to the Verhulst (logistic) function the cumulative number $y(t)$ of items is of the form (for example, see: refs. [2,22,23,25,41,42])

$$y(t) = \frac{y_{\max}}{1 + \left(\frac{y_{\max}}{y_0} - 1\right)\exp(-\beta t)}, \quad (2)$$

where y_0 is the number of items at $t = 0$, the maximum number of items $y_{\max}$ is the so-called carrying capacity, and β is the so-called Malthusian instantaneous growth rate parameter. According to the Gompertz function the relationship between $y(t)$ and t is given by (for example, see: [2,42-44])

$$y(t) = y_{\max}\left\{1 - \exp\left[-\frac{\lambda}{c}(\exp ct - 1)\right]\right\}, \quad (3)$$

where λ and c are fitting parameters.

For situations when $t \ll \Theta$, Avrami–Weibull equation (1) takes the power-law form:

$$y(t) = y_{\max}\left(\frac{t}{\Theta}\right)^q, \quad (4)$$

which takes a linear form when $q = 1$. The notation "«" denotes that the time t is much smaller than the time constant Θ. Note that, for $ct \ll 1$, Gompertz function (3) reduces to the form of Avrami–Weibull function (1), i.e.

$$y(t) = y_{\max}\{1 - \exp(-\lambda t)\}. \quad (5)$$

when $q = 1$ and $\lambda = 1/\Theta$. For $\lambda t \ll 1$, Eq. (5) reduces to the linear form of Eq. (4).

The above functions (1), (2) and (3) of growth of cumulative number $y(t)$ of items with time t describe cumulative distribution function $F(t) = y(t)/y_{\max}$ and frequency density function $f(t) = dF(t)/dt = [dy(t)/dt]/y_{\max}$, where $y_{\max}$ is the maximum number (saturation limit) of items. These functions represent original versions of the three models and the differential $dy(t)/dt$ may be considered as rate of growth of $y(t)$ items whereas $[dy(t)/dt]/y_{\max}$ as dimensionless rate. As mentioned in the Introduction, various modified versions of Verhulst (logistic) and Gompertz models have been proposed in the literature to explain real $y(t)$ data satisfactorily but until now the Avrami–Weibull function has been used in its original version. It is beyond the scope of this paper to discuss the modified versions of different models.

3. Analysis of growth behavior of different items

The growth data were analyzed using commercially available "Origin 9.1" package. This package employs nonlinear least-squares fitting of the data and yields values of the fitting parameters of an equation, their standard deviations, chi-square (χ^2) residual and the corresponding goodness-of-the-fit parameter R_y^2.

For the analysis we used typical examples of data on the cumulative growth of papers and citations earned by individual and all successively published papers, up to 2013, of selected top-cited authors reported by Chuang and Ho [45] and data on the growth behavior of different phenomena such as germination of seeds, three different bacteria, and overall crystallization of polypropylene. The basic bibliometric data of the selected authors comprised cumulative number $N(t)$ of papers of J. Felsenstein, cumulative number $L(t)$ of citations received by two top-cited papers of D.R. Cox and cumulative number $L(t)$ of citations received by four top-cited papers and by all papers published by U.K. Laemmli, taking publication and citation duration $t = Y - Y_0$, with Y and Y_0, as the years corresponding, respectively, to t and $t = 0$. The data were collected during 20-25 October 2014 from the Thomson Reuters' Web of Knowledge database. The growth data on the other phenomena were collected from the published literature.

3.1. Some specific examples of growth behavior of items and their analysis by different functions

Figures 1a and 1b show examples of the plots of cumulative number $N(t)$ of papers of Felsenstein and cumulative number $L(t)$ citations of top-cited papers of Cox with time t, respectively, exhibiting S-shaped curves. In the figures solid, dashed and dotted curves are drawn according to Avrami–Weibull, Verhulst and Gompertz functions, respectively, with the best-fit values of parameters given in Tables 1–3.

From the values of goodness-of-the-fit parameter R_y^2 listed in these tables it may be noted that Avrami–Weibull function fits the data better than Verhulst and Gompertz functions. As seen from the best-fit plots of the data according to Verhulst function (2), noticeable deviations in the fit of the data are encountered in the initial and later stages. However, the fit improved when

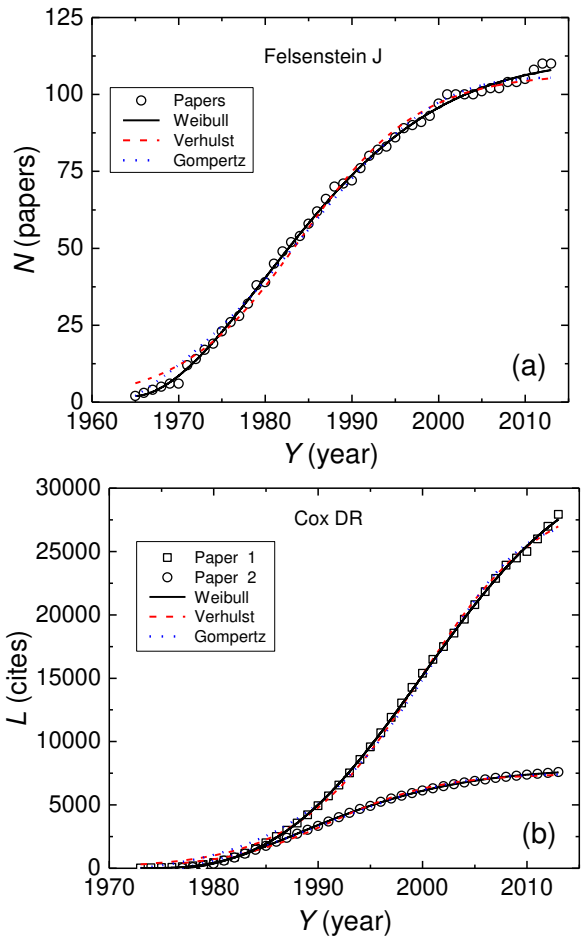


Figure 1: Example of evolution of (a) cumulative number $N(t)$ of papers of Felsenstein and (b) cumulative number $L(t)$ citations of top-cited papers of Cox with time t showing S-shaped curves. Solid, dashed and dotted curves represent plots according to Avrami–Weibull, Verhulst and Gompertz functions, respectively, with the best-fit values of parameters given in Tables 1–3.

the initial $y(t)$ corresponding to Y_0 was taken as $C > 0$ (see Tables 2 and 3).

Figure 2 illustrates a typical example of the plots of cumulative number $L(t)$ of citations of all progressively published papers and the top-cited paper of Laemmli with time t showing S-shaped curves. As in Figure 1, solid, dashed and dotted curves represent plots according to Avrami–Weibull, Verhulst and Gompertz functions, respectively, with the best-fit values of parameters given in Tables 1–3. The last three steeply rising points of the data beyond the arrow were excluded from analysis.

The values of the goodness-of-the-fit parameter R_y^2 given in Tables 1–3 again show that Avrami–Weibull function fits the data better than Verhulst and Gompertz functions which show similar R_y^2 parameter. As in the plots of Figure 1, noticeable deviations in the plots are encountered in the initial and later stages of the data.

Figure 3 shows the evolution of germination of cumulative number $N(t)$ of unirradiated and irradiated tomato seeds at 15 °C with time t. Solid, dashed and dotted curves represent plots according to Avrami–Weibull, Verhulst and Gompertz functions, respectively, with the best-fit values of parameters given in Tables 1–3. As seen from the values of the R_y^2 parameter, the $N(t)$ data are best described by Avrami–Weibull function but worst by Verhulst function. This poor fit is mainly due to the fact that, apart from large deviations from the fit at $t > 200$ h, according to this relation $N_0 > 0$ at $t = 0$ but according to the original $N(t)$ data $N_0 = 0$.

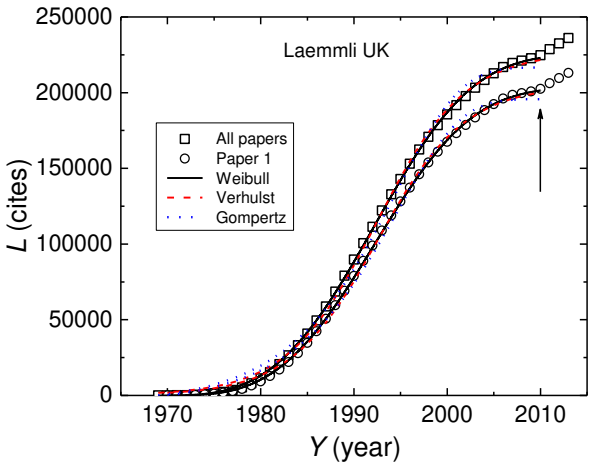


Figure 2: Example of evolution of cumulative number $L(t)$ of citations of all progressively published papers and the top-cited paper of Laemmli with time t showing S-shaped curves. Solid, dashed and dotted curves represent plots according to Avrami–Weibull, Verhulst and Gompertz functions, respectively, with the best-fit values of parameters given in Table 1. The last three points of the data were excluded from analysis.

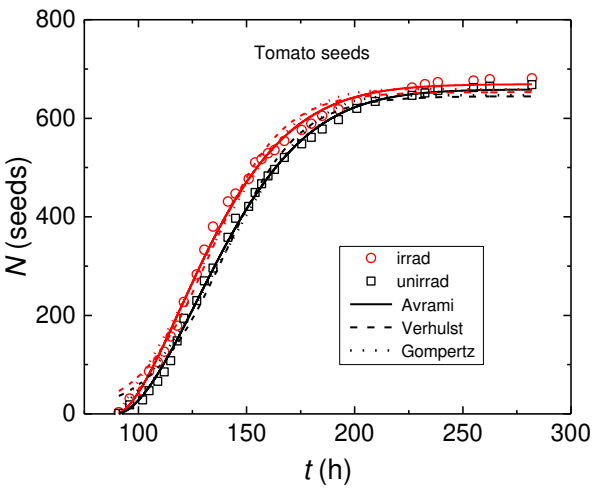


Figure 3: Evolution of germination of cumulative number $N(t)$ of unirradiated and irradiated tomato seeds at germination temperature T = 15 °C with time t showing S-shaped curves. Solid, dashed and dotted curves represent plots according to Avrami–Weibull, Verhulst and Gompertz functions, respectively, with the best-fit values of parameters given in Table 1. Original data from Gładyszewska [34].

The above examples show that Avrami–Weibull function (1), in general, describes different types of the data better than Gompertz and Verhulst functions. Keeping in view the fact that Avrami–Weibull relation (1) describes the time dependence of cumulative citations $L(t)$ received by individual papers better than the other two functions, the $L(t)$ data of three subsequently

Table 1: Values of parameters of Verhulst equation (2)

Data	Source	y_{max}	y_0	β	t_0	χ^2	Ry^2
$N(t)$	Felsenstein, papers	106.8	6.62	0.1459	1965 yr	7.163	0.9947
$L(t)$	Cox, paper 2	7461	414.1	0.1963	1977 yr	35809	0.9947
	Cox, paper 1	29798	314.6	0.1700	1973 yr	151915	0.9983
	Laemmli, paper 1	204971	1662	0.2134	1970 yr	$7.3730 \cdot 10^6$	0.9987
	Laemmli, all papers	227048	1680	0.2096	1969 yr	$1.0013 \cdot 10^7$	0.9986
$N(t)$	Tomato, 15°C, unirr[a]	644.8	34.63	0.0577	90 h	494.41	0.9909
	Tomato, 15°C, irr[a]	653.0	43.73	0.0609	90 h	566.76	0.9892

[a] Abbreviations: unirr – unirradiated; irr – irradiated; Tomato – tomato seed germination.

Table 2: Values of parameters of Gompertz equation (3)

Data	Source	C	y_{max}	$10^3\lambda$	10^2c	t_0	χ^2	Ry^2
$N(t)$	Felsenstein, papers	2	103.6	17.23	6.87	1965 yr	5.595	0.9958
$L(t)$	Cox, paper 2	23	7391	24.23	8.73	1977 yr	14515	0.9978
	Cox, paper 1	9	27189	3.53	12.49	1973 yr	229894	0.9975
	Laemmli, paper 1	4	195749	3.88	14.72	1970 yr	$2.282 \cdot 10^7$	0.9962
	Laemmli, all papers	2	216577	3.50	14.57	1969 yr	$3.021 \cdot 10^7$	0.9959
$N(t)$	Tomato, 15°C, unirr[a]	0	646.6	7.14	2.46	90 h	432.75	0.9920
	Tomato, 15°C, irr[a]	0	658.8	9.69	2.20	90 h	482.23	0.9908

[a] Abbreviations: unirr – unirradiated; irr – irradiated; Tomato – tomato seed germination.

Table 3: Values of parameters of Avrami–Weibull equation (1)

Data	Source	C	y_{max}	Θ	q	t_0	χ^2	Ry^2
$N(t)$	Felsenstein, papers	2	109.7	24.2 yr	1.766	1965 yr	1.591	0.9988
$L(t)$	Cox, paper 2	23	7864	18.2 yr	1.706	1977 yr	607.2	0.9999
	Cox, paper 1	9	31655	31.3 yr	2.888	1973 yr	16960	0.9998
	Laemmli, paper 4	1	963.5	16.4 yr	2.083	1971 yr	--	0.9997
	Laemmli, paper 3	16	3743	17.4 yr	1.784	1974 yr	--	0.9983
	Laemmli, paper 2	16	5670	11.2 yr	1.667	1977 yr	--	0.9996
	Laemmli, paper 1	4	203935	25.2 yr	3.193	1970 yr	$8.7679 \cdot 10^5$	0.9998
	Laemmli, all papers	2	225782	26.0 yr	3.258	1969 yr	$1.5617 \cdot 10^6$	0.9998
$N(t)$	Tomato, 15°C, unirr[a]	0	658.5	60.0 h	1.758	90 h	91.20	0.9983
	Tomato, 15°C, irr[a]	0	669.1	53.2 h	1.614	90 h	157.35	0.9970

[a] Abbreviations: unirr – unirradiated; irr – irradiated; Tomato – tomato seed germination.

top-cited papers (i.e. papers 2, 3 and 4) were analyzed by using Eq. (1). The best-fit parameters for these data are included in Table 3. It is interesting to note that the values of the time constant Θ and the exponent q differ widely among the individual papers but their values for the combined cumulative $L(t)$ citations of all papers are higher than those for the citations of individual papers. This observation is consistent with the modeling of items in a previous paper [39].

The above analysis shows that Avrami–Weibull function (1) can be used to describe the data of growth behavior of both abstract (imaginary) items such as papers published by an author and citations received by his/her individual and all successively published papers as well as material (real) items such as germination of seeds and bacteria.

3.2. Effect of temperature on the growth behavior of material items

Every plot of the growth of cumulative number $y(t)$ of items against time t is characterized by three parameters: dimensionless growth rate $R = dy/y_{max}dt$ determined from the linear part of a $y(t)/y_{max}$ plot, time lag t_0 corresponding to the onset of initial growth, and time constant Θ when the rate R reaches a maximum value. In the case of growth of material items such as bacteria, it is usually observed that the growth rate R increases whereas the time lag t_0 and the time constant Θ decrease with increasing temperature T. Some typical examples of the effect of temperature on these parameters are presented below.

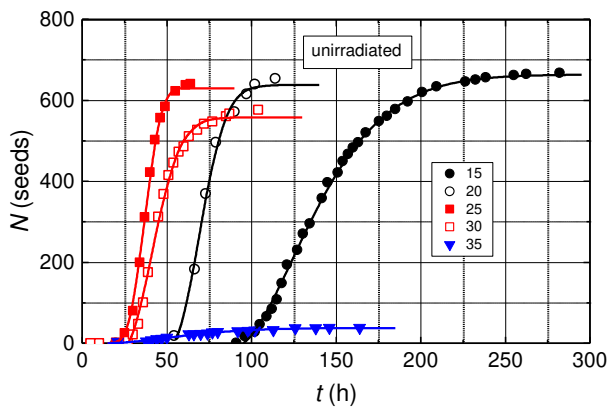


Figure 4: Evolution of germination of cumulative number $N(t)$ of unirradiated tomato seeds at different T with time t showing S-shaped curves. Curves represent plots according to Avrami–Weibull function with the best-fit values of parameters given in Table 1. Original data from Gładyszewska [34].

Figure 4 shows, as an example, the evolution of germination of cumulative number $N(t)$ of unirradiated tomato seeds at different temperature T with time t

showing S-shaped curves. Germination temperature T given in the figure is in °C. Curves represent plots according to Avrami–Weibull function (1) with the best-fit values of parameters given in Table 4. A similar behavior was observed in the case of germination of irradiated seeds by specific dose of γ radiation. The best-fit parameters for these data are included in Table 4.

It may be seen from Table 4 that, irrespective of the initial irradiation treatment, at every germination temperature there is a time lag t_0 for germination. With an increase in temperature, this time lag t_0 shows a general decreasing trend. The time constant Θ also shows a similar decreasing trend up to 30 °C and then it increases to a high value at 35 °C. In contrast to the trends of time lag t_0 and time constant Θ, the exponent q initially increases from a relatively low value at 15 °C, then attains a maximum at 25 °C and subsequently at 35 °C decreases to a value even lower than that at 15 °C.

Table 4: Values of parameters N_{max}, Θ, q and t_0 of Avrami–Weibull equation for $N(t)$ data of tomato seed germination

Sample[a]	T (°C)	N_{max}	Θ (h)	q	t_0 (h)
Unirradiated	15	658.5	60.06	1.758	90
	20	635.6	25.03	2.291	50
	25	629.5	24.66	3.611	15
	30	557.5	21.80	1.874	26
	35	37.8	57.07	1.720	15
Irradiated	15	669.1	53.17	1.614	90
	20	682.1	23.17	1.626	50
	25	660.9	22.90	3.448	15
	30	559.4	27.00	2.328	20
	35	73.6	61.29	1.103	5

Examination of the plots of the germination of both unirradiated and irradiated cumulative number $N(t)$ of seeds as a function of germination time t reveals that the seed germination rate $R = \mathrm{d}N/N_{max}\mathrm{d}t$, determined from the linear parts of the $N(t)$ plots, initially increases from a relatively low value at 15 °C, then attains a maximum at 25 °C and subsequently at 35 °C decreases to a value even lower than that at 15 °C. This trend is similar to that followed by the temperature dependence of the exponent q.

The effect of temperature on time lag t_0 and time constant Θ is not confined to germination of tomato seeds alone. The effect of temperature is observed, among others, on various bacterial growth in different media (for example, see: refs. [19-23, 25-30]), and population growth of individual biological species (for example, see: ref. [31]).

Data on the parameters Θ and q of Avrami–Weibull function for two toxin-producing *Bacillus cereus* (BC) and *Escherichia coli* (EC) microorganisms in carrot broth obtained at different temperatures have been reported by Fernandez et al. [19] and Aragao et al. [27], respectively. From these studies one observes that Θ decreases with an increase in temperature in the temperature range investigated in the growth of the microorganisms. However, the values of q suggest that it is independent of temperature. These trends of Θ and q are somewhat different from those observed in the case of germination of tomato seeds.

Growth behavior of various bacteria in different media under different experimental conditions such as temperature and pH has been analyzed in several studies using different functions [21,23,29,30]. In these studies microorganism cell concentration at time t, concentration at $t = 0$, and their maximum concentration, denoted by $y(t)$, y_0 and y_{max}, respectively, in Eqs. (1)-(3), are customarily expressed in log(cfu mL^{-1}). Examination of the data of the growth rates $R = \mathrm{d}y/y_{max}\mathrm{d}t$ of various microorganisms grown at different temperatures shows that their value increases with increasing temperature and that, at a particular temperature, the rate R is frequently inversely related to the time lag t_0, with a proportionality constant K characteristic of the bacteria–medium system, which lies between 1 and 3 (cf. refs. [21,29]). In the case of tomato seeds this inverse relationship between time lag t_0 for germination and growth rate R is observed up to 25 °C. Obviously, here the trend at temperatures of 30 and 35 °C is anomalous. A possible explanation of this anomalous trend in the germination of tomato seeds is associated with differences in the kinetics of *testa* and *endosperm* ruptures during germination [46,47].

We assume that the dependence of germination rate R of tomato seeds in the sand and growth rates R of microorganisms in different media are instances of chemical reactions with reaction rate constants k described by Arrhenius-type relation (A3) and that the time lag t_0 and time constant Θ for germination of tomato seeds and growth of microorganisms are inversely proportional to the reaction rate constant k with a proportionality constant K. Then from Eq. (A3) the dependence of rate constant k, and time lag t_0 and time constant Θ on temperature T, taken in Kelvin, may be described by

$$\ln k = \ln A - \frac{\Delta G_R^0}{R_G T}, \tag{6}$$

$$\ln t_0, \Theta = \ln\left(\frac{K}{A}\right) + \frac{\Delta G_R^0}{R_G T}, \tag{7}$$

respectively. Here $\Delta G_R{}^0$ is the energy difference between reactants and products, all in their respective ground states, A is the frequency factor, and R_G is the gas constant. These relations predict linear dependences between ln$k$ or lnt_0,Θ and $1/T$, with intercept ln$k$ or ln(K/A) and slope $\Delta G_R{}^0/R_G$, which enables to calculate frequency factor A or relative frequency factor A/K and activation energy $\Delta G_R{}^0$ for the reaction.

Figure 5a shows the dependence of growth rate R, taken as a measure of rate constant k, of different bacteria on temperature T according to relation (6). To facilitate a comparison of the trends of the dependence according to Eq. (6) with those of the dependence of lnt_0 and lnΘ on $1/T$ according to Eq. (7), the $R(T)$ data are shown as plots of –ln$R$ against $1/T$. The original data for *Listeria monocytogenes* (LMC) in 2% milk and 12 and

30% milk creams are from Lobacz and Kowalik [30], for *Yersinia enetrocolitica* in Camembert-type cheese from Kowalik and Lobacz [29], for *Bacillus cereus* AVTZ415 (BC 415) and AVZ421 (BC 421) in nutrient carrot broth and for *Bacillus cereus* AVTZ415 (BC 415) in natural carrot broth from Valero et al. [21], and for *Escherichis coli* (EC) in nutrient broth from Fujikawa et al. [23]. In view of large scatter in the values of R for LMC in different samples of milk cream and for BC in different broths the data for LMC in 12% milk cream and BC 415 in neutral carrot broth were fitted according to Eq. (6) with the best-fit values of intercept ln$A$ and slope $\Delta G_R^0/R_G$ listed in Table 5. The values of A and ΔG_R^0 calculated from the above parameters are included in the table.

Figure 5b illustrates the plots of lnt_0 for germination of unirradiated and irradiated tomato seeds and for growth of different bacteria as a function of $1/T$ according to Eq. (7). The data of time lag t_0 for the germination of tomato seeds were obtained from the original cumulative $N(t)$ data reported by Gładyszewska [34] using Avrami–Weibull (AW) function (present author) and Verhulst/logistic) (V/log) function by Gładyszewska [34], whereas those for the growth of *Listeria mono-cytogenes* (LMC) in milk are from Lobacz and Kowalik [30], for *Yersinia enetrocolitica* (YE) in Camembert-type cheese from Kowalik and Lobacz [29], and for *Bacillus cereus* AVTZ415 (BC 415) and AVZ421 (BC 421) in nutrient carrot broth and for *Bacillus cereus* AVTZ 415 (BC 415) in natural carrot broth from Valero et al. [21]. For tomato germination and bacteria growth in carrot broth linear plots are drawn for the data obtained by Avrami–Weibull function of unirradiated tomato seeds and for *Bacillus cereus* AVTZ415 with the best-fit values of intercept ln(K/A) and slope $\Delta G_R^0/R_G$ given in Table 5. The calculated values of A/K and ΔG_R^0 for different growths are also listed in Table 5.

Figure 5c shows the dependence of time constant Θ for growth of three different bacteria on temperature T according to Eq. (7). The original data for *Bacillus cereus* AVTZ415 (BC 415) and AVZ421 (BC 421) in neutral carrot broth are taken from Fernandez et al. [19] and for *Escherichis coli* (EC) in nutrient broth are from Aragao et al. [27]. The best-fit plots of the data according to relation (7) are drawn with the values of intercept ln(K/A) and slope $\Delta G_R^0/R_G$ listed in Table 5. From these best-fit parameters the calculated values of A/K and ΔG_R^0 are included in Table 5.

From Table 5 it may be noted that the value of ΔG_R^0 for bacteria like *Bacillus cereus* AVTZ415 and *BC* AVZ421 in carrot broth, *Listeria monocytogenes* in milk and *Yersinia enerocolitica* in Camembert-type cheese increases in the sequence: $R(t)$, $t_0(T)$ and $\Theta(T)$ data, whereas the value of A and A/K for them increases in the sequence: $t_0(T)$, $R(t)$ and $\Theta(T)$ data. The values of ΔG_R^0 and A (or A/K) corresponding to a particular growth parameter such as growth rate R or time lag t_0 vary enormously among different samples. For example,

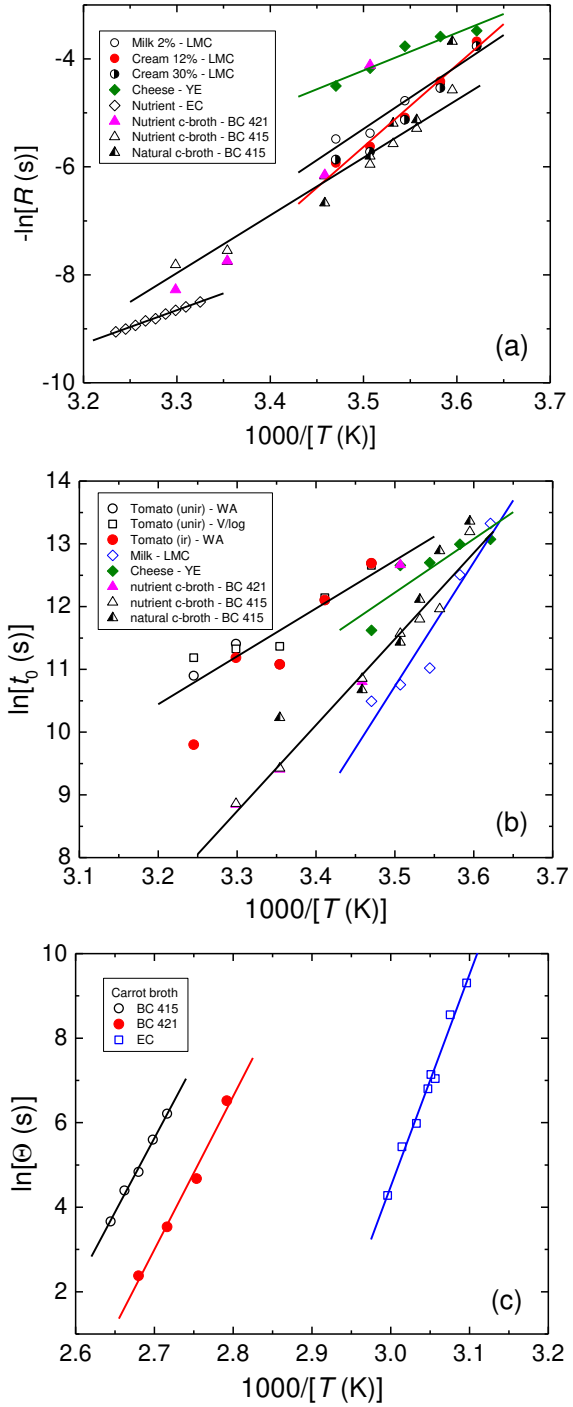


Figure 5: Dependence of (a) dimensionless growth rate R of different bacteria on temperature T according to relation (6), and (b) time lag t_0 for growth of different items and (c) time constant Θ for the growth of three different bacteria in carrot broth on temperature T according to relation (7). For the purpose of comparison with the $t_0(T)$ and $\Theta(T)$ data according to Eq. (8), in (a) data of growth rate $R(T)$ are presented as $-\ln R$. See text for details.

the values of ΔG_R^0 obtained from the $R(T)$ data are relatively low for the growth of *Escherichis coli* in nutrient broth and *Yersinia enerocolitica* bacteria in Camembert-type cheese and in comparison with that for the growth of *Listeria monocytogenes* in milk products. Similarly, the value of ΔG_R^0 obtained from $t_0(T)$ data is relatively

Table 5: Parameters of Eqs. (6) and (7) from $R(T)$, $t_0(T)$ and $\Theta(T)$ data

Data	Sample	Medium	$-\ln A$, $\ln(K/A)$	$10^3\Delta G_R^0/R_G$ (K)	A, A/K (s^{-1})	ΔG_R^0 (kJ/mol)	Ry^2	Ref.
$R(T)$	*BC* AVT415, AVTZ421	Nutrient c-broth	–43.25	10.69	$6.1\cdot10^{18}$	88.9	0.9754	[21]
	BC AVT415	Natural c-broth	–43.25	10.69	$6.1\cdot10^{18}$	88.9	0.9754	[21]
	E Coli	Nutrient c-broth	–29.18	6.22	$4.7\cdot10^{12}$	51.7	0.9974	[23]
	L-monocytogenes	UHT milk 2%	–45.82	11.58	$7.9\cdot10^{19}$	96.3	0.9473	[30]
		UHT cream12%	–58.65	15.15	$3.0\cdot10^{25}$	126.0	0.9744	[30]
	Y-enterocolotica[a]	Cheese	–28.57	6.96	$2.6\cdot10^{12}$	57.9	0.9249	[29]
$t_0(T)$	Tomato seeds	Soil	–14.0	7.64	$1.2\cdot10^{6}$	61.3	0.7715	[34]
	BC AVT415, AVTZ421	Nutrient c-broth	–36.53	13.72	$7.3\cdot10^{15}$	114.1	0.9744	[21]
	L-monocytogenes	UHT Milk	–58.37	19.74	$2.2\cdot10^{25}$	164.1	0.8829	[30]
	Y-enterocolotica[a]	Cheese	–17.65	8.54	$4.6\cdot10^{7}$	71.0	0.6938	[29]
$\Theta(T)$	*BC* AVTZ421	Nutrient c-broth	–89.06	35.07	$4.8\cdot10^{38}$	291.1	0.9937	[19]
	BC AVT415	Nutrient c-broth	–95.01	36.30	$1.8\cdot10^{41}$	301.8	0.9808	[19]
	E Coli	Culture c-media	–146.05	50.18	$2.7\cdot10^{63}$	417.2	0.9846	[27]

low for the germination of tomato seeds and for the growth of *Yersinia enerocolitica* bacteria in Camembert-type cheese in comparison with that for the growth of *Listeria monocytogenes* in milk products. From these results it may be concluded that, for a particular bacterial growth, the differences in the values of ΔG_R^0 and A obtained from $R(T)$ data from those of ΔG_R^0 and A/K from $t_0(T)$ and $\Theta(T)$ data are due to different processes associated with them.

3.3. Overall crystallization of various compounds

Overall crystallization of compounds from melts and solutions also exhibits features similar to those of growth of material items discussed above. However, in this case, the plots of growth of overall crystallization are presented in terms of cumulative mass $y(t)$ of the crystallized compound instead of cumulative number $y(t)$ of material items against time t. These $y(t)$ plots are also characterized by three parameters: dimensionless growth rate $R = \mathrm{d}y/y_{max}\mathrm{d}t$, time lag t_0, and time constant Θ.

The process of overall crystallization of a compound involves the formation of stable nuclei and their subsequent development to visible dimensions in the bulk medium (see Appendix B). Both of these processes depend, among others, on crystallization temperature (for example, see: refs. [35-37]). Figure 6 illustrates a typical example of the dependence of overall crystallization fraction $y(t)/y_{max}$ of polypropylene on time t at different crystallization temperatures. Original data of the figure are from Lopez-Manchado et al. [48] but best-fit curves are drawn by Padar et al. [36], according to Avrami–Weibull function (1) assuming the initial time $t_0 = 0$ for the onset of overall crystallization. It may be noted that the best-fit plots somewhat deviate from the experimental data points at low and high crystallization times. These deviations are mainly associated with the assumption that $t_0 = 0$ [37]. The experimental data reveal that the initial time t_0 when crystallization fraction begins to increase is not zero and increases with increasing crystallization temperature.

The crystallization rate $R = \mathrm{d}y/y_{max}\mathrm{d}t$ of polypropylene, determined from the linear part of the $y(t)/y_{max}$ plots at a particular temperature, decreases with an increase in crystallization temperature T. According to Arrhenius-type relation (A3), the higher the reaction temperature T, the higher is the value of the reaction rate constant k. This inference is contrary to the dependence of crystallization rate $R = \mathrm{d}y/y_{max}\mathrm{d}t$ of polypropylene on crystallization temperature T (Figure 6). Therefore, unlike the cases of temperature dependence of cumulative growth of germination of tomato seeds and growth of microorganisms discussed above, Arrhenius-type relation (A3) does not describe the dependence of time lag t_0 for crystallization from melts on T.

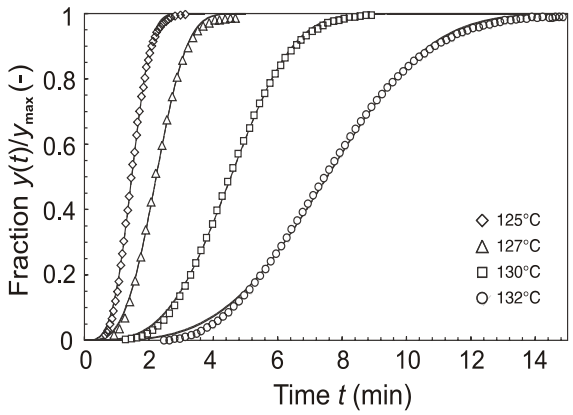


Figure 6: Example of dependence of overall crystallization fraction $y(t)/y_{max}$ of polypropylene on time t at different crystallization temperature. Solid curves represent plots according to Avrami–Weibull function. Original data from Lopez-Manchado et al. [48] but best-fit plots are drawn by Padar et al. [36]. Adapted from Padar et al. [36].

Figure 6 shows that the crystallization rate R is related to the corresponding time lag t_0 in the plots. The lower the crystallization rate at a particular temperature T, the higher is the value of the time lag t_0 for detectable crystallization. There is also a similar relationship between R and time constant Θ. In order to explain the temperature dependence of crystallization rate R, time lag t_0 and time constant Θ for crystallization from melts on T, one uses Eq. (A10) and assumes that $R = K_1J$, and t_0 and $\Theta = K_2/J$, where K_1 and K_2 are new proportionality constants different from K. Then from Eq. (A10) one obtains

$$\ln R = \ln(K_1 J_0) - \frac{B'}{T(\Delta T)^2}, \quad (8)$$

and

$$\ln t_0, \Theta = \ln\left(\frac{K_2}{J_0}\right) + \frac{B'}{T(\Delta T)^2}, \quad (9)$$

where B' is given by Eq. (A11), and the temperature difference $\Delta T = (T_m - T)$, with T_m as the melting point of the crystallizing compound. Eqs. (8) and (9) predict $B' > 0$ because the main contribution to R and t_0 or Θ comes from ΔT rather than from T.

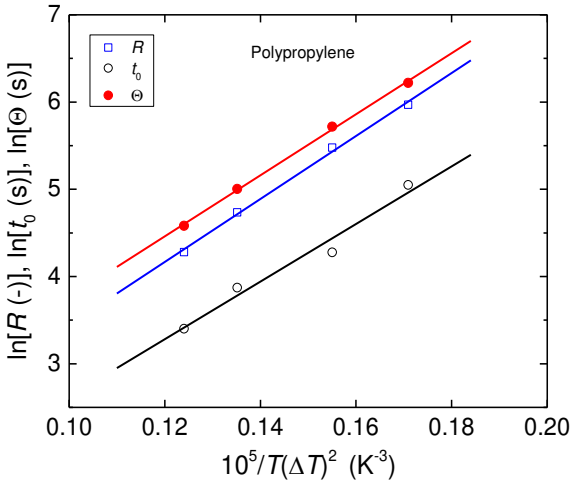


Figure 7: Plots of $-\ln R$, $\ln\Theta$ and $\ln t_0$ against $1/T(\Delta T)^2$ for crystalization of polypropylene according to Eqs. (8) and (9). For the purpose of comparison with the $t_0(T)$ and Θ(T) data according to Eq. (8), data of crystallization rate $R(T)$ are presented as $-\ln R$. See text for detail.

Table 6: Parameters of relation (A10) from $R(T)$, $t_0(T)$ and $\Theta(T)$ data of crystallization of polypropylene

Data	$-\ln J_0$, $\ln(K'/J_0)$	$10^5 B'$ (K^3)	J_0, J_0/K' (s^{-1})	Ry^2
$R(T)$	–0.168	36.1	1.18	0.9955
$t_0(T)$	–0.684	33.0	1.98	0.9572
$\Theta(T)$	0.257	35.0	0.77	0.9980

Figure 7 presents the $R(T)$, $t_0(T)$ and $\Theta(T)$ data of crystallization of polypropylene (T_m = 170 °C) from the melt according to relation (8) whereas the linear plots are drawn with the values of the parameters listed in Table 6. The $t_0(T)$ data were visually determined from the original plots of evolution of crystallization at different temperatures. The $R(T)$ data were recovered from the figure in the paper of Padar et al. [36] but the $\Theta(T)$ data were calculated from the values of best-fit constants q and Θ^{-q} reported in that paper. It may be noted from the table that the value of B' for the $R(T)$, $t_0(T)$ and $\Theta(T)$ data is essentially constant but the value of $\ln(K_2/J_0)$ for the $\Theta(T)$ data is higher than that for the $t_0(T)$ data. The constancy of B' is associated with the interfacial energy γ of the crystallites (see Eq. (A11)) but the different values of J_0/K_2 are associated with different sizes of the crystallites corresponding to t_0 and Θ. Obviously, the process of overall crystallization is entirely different from the processes involved in bacterial growth discussed above.

3.4. Distinction between processes of chemical reactions and overall crystallization

The plots of the growth of cumulative number $N(t)$ of material items such as seeds (Figures 3 and 4) and the fraction of cumulative mass $y(t)/y_{max}$ of crystallized compounds (Figure 6) are similar and can be described reliably by Avrami–Weibull relation (1). However, the temperature dependence of rate R, time lag t_0 and time constants Θ for the growth of material items and overall crystallization of compounds differs fundamentally from each other. In the former cases the temperature dependence of these parameters can be described by Eqs. (6) and (7). These equations follow from the Arrhenius relation (A3) where the three parameters are related to the activation energy ΔG_R^0 involved in a reaction. However, in the latter case the temperature dependence of these parameters can be described by Eqs. (8) and (9). These relations follow from Eq. (A8) where the three parameters of crystallization are associated with the driving force ΔG_R. Obviously, the time lag t_0 for a phenomenon corresponds to an initial time period in which an equilibrium is attained in the system from the standpoint of occurrence of chemical reactions with rate k (see Eq. (A3)) or crystallization with nucleation rate J (see Eq. (A8)).

As discussed in Appendix B, the reaction rate constant k of a qth order chemical reaction and the time constant Θ of overall crystallization are mutually related by: $k = \Theta^{-q}$. Therefore, for overall crystallization occuring by instantaneous and progressive nucleation modes the relationships may be given by (see Appendix B)

$$k = \Theta^{-q} = \frac{\kappa g^q}{V} N_m, \quad (10)$$

and

$$k = \Theta^{-q} = \frac{\kappa g^{q-1}}{q} J, \quad (11)$$

which hold when $q \geq 0$ and $q \geq 1$, respectively.

From the above considerations it may be concluded that the processes associated with chemical reactions follow simple Arrhenius-type relation (A3) of the dependence of parameters R, t_0 and Θ on T with an activation energy ΔG_R^0, but those of overall crystallization follow relation (A8) in which the parameters R, t_0 and Θ are related to dependence of the number N_m of nuclei formed or the nucleation rate J on driving force ΔG_R.

4. Basic concepts about generation of abstract items

4.1. Motivation threshold for nucleation/growth of abstract items

The generation of *abstract* or *imaginary* items such as the number of papers and their citations is, by its nature, similar to a chemical reaction between the initial reactants involving the formation of an activated complex

followed by its dissociation into reaction products (for example, see [49]). It also has its analog in behavioral psychology as stipulated in the stimulus–response behavior theory involving attainment of a goal by human beings and animals through generation of inner tendency (i.e. restless or transition state) of the behaving organism, aroused by the external stimulus (see ref. [50]; Chapters 5 and 6). In this case, the ultimate goal is a consequence of decision making and is associated with the problem of motivation, a term "often used in reference to the conscious feeling of desire and the whole complex of ideas and feelings which together seem to constitute the conscious antecedents of behavior according to traditional wisdom" (ref. [50]; page 273).

It should be noted that the processes of chemical reactions and decision making differ from each other. The occurrence of chemical reactions is described in terms of thermodynamics but it is difficult to give a thermodynamic interpretation for the transition state involved in decision making. However, results of motivations of authors to cite published papers indeed reveal that there is always a cognitive pressure on authors towards citing a given paper and there is a threshold value of this cognitive pressure for the citation of a paper [51]. Threshold cognitive pressure on an author in citing previously published papers is equivalent to the free energy change ΔG_a involved in the formation of an activated complex X^* whereas cognitive pressure in citing is analogous to the total free energy change ΔG_a' associated with the dissociation of the activated complex X^* into the formation of reaction products (see Figure A1). Intuitionally, the process of citation of a paper P_{W1} of an author W1 by the citing author W2 in his/her paper P_{W2} bearing citation C_{W1} to the paper PW1 of author W1 may be represented in the form of the reaction (cf. Eq. (A1))

$$P_{W1} + W2 \rightarrow X^* \rightarrow P_{W2} + C_{W1} + \Delta G_R, \qquad (12)$$

with $\Delta G_a' = \Delta G_a + \Delta G_R$. Here $\Delta G_a'$, ΔG_a and ΔG_R of Fig. A1 now represent the overall, the threshold and the resulting cognitive citation pressures, respectively. The threshold cognitive pressure is the so-called motivation threshold for the citation of an article. Note that the resulting cognitive citation pressure ΔG_R is essentially associated with the attractiveness of the cited paper P_{W1}. The higher the attractiveness of this paper, the higher is its citability.

With the above information on the resulting cognitive citation pressure ΔG_R as the driving force for the citations received by a paper, we may apply the concepts of occurrence of chemical reactions and crystallization of solid phases from melts and supersaturated solutions to understand the process of citations of papers (see Appendix A). While applying Eq. (A3) to describe the occurrence of chemical reactions and Eq. (A8) to describe the process of nucleation of solid phase we ignore the role of temperature and assume it as a constant parameter. As seen from Eq. (A3), in the case of chemical reactions, the value of cumulative volume (reactant concentration), determined by reaction rate constant k, decreases with an increase in ΔG_R. In contrast to this, Eq. (A8) shows that, in the case of formation and growth of nuclei, the cumulative volume, determined by nucleation rate J, increases with an increase in ΔG_R. Therefore, it may be argued that the process of citations of papers is similar to that of overall crystallization of solid phases from melts and solutions and differs fundamentally from chemical reactions.

Citations to individual published articles of authors with time is a typical example of information production process. We use below the concepts of the occurrence of overall crystallization of solid phase to understand these processes of growth of *abstract* items.

4.2. Growth behavior of cumulative volume of abstract items

The time dependence of cumulative number of abstract items such as journals, articles and authors in a scientific field is a continuous information production process in which new items are produced progressively. An author publishing N papers in his research career and a research paper published by a given author receiving a total of L citations during its citation life are typical examples of closed systems. Here the author publishing N papers in his/her academic career and each ith paper receiving L_i citations are primary sources producing primary items (i.e. papers and citations, respectively). These source–items isolated systems are simple in nature because the source is an individual entity. However, when a group of primary items (for example, $N(t)$ papers published by the author in time t) act as secondary sources which produce cumulative secondary items (for example, cumulative number $L(t)$ of citations such that $L(t) = \Sigma L_i(t)$), one deals with complex or composite systems.

The processes of generation of primary items A from a source S and of secondary items B from primary items A acting as secondary sources of (secondary) items are schematically shown in Figure 8. In the figure cumulative number of primary and secondary items from individual primary and secondary sources are indicated by N and n, respectively. However, in general, when an individual source is not distinguished as primary or secondary source, the cumulative number of items is denoted by y.

In order to describe the time dependence of cumulative number y of primary items (e.g. papers) produced by a primary source (e.g. by an author) we may follow the concepts of overall crystallization of a solid phase occurring in a closed liquid system of fixed volume V. We make the following assumptions:

(1) Every source–items system is confined to its own production space (i.e. volume V) and is analogous to the fixed crystallization volume in the case of overall crystallization. This abstract (imaginary) space available for the production of items by a source (i.e. in a system) is fixed.

(2) The process of production of items is similar to a chemical reaction and involves the formation of an activated complex and its dissociation into products. This means that there are threshold free energy

changes ΔG_a and overall free energy changes $\Delta G_a+\Delta G_R$ for the generation of items. In the case of citing of published papers by authors, Vinkler [52] calls ΔG_a and ΔG_R cognitive pressure and threshold cognitive pressure, respectively.

(3) Items are produced at active centers in the available space by the formation and growth of 3D clusters or nuclei such that the time dependence of the volume $V_c(t)$ of items produced at time t occurs in the available volume $V-V_c(t)$.

(4) Items can be generated both by instantaneous and progressive nucleation (cf. Fig. B1).

(5) As in the solid nuclei composed of n growth units, nuclei of items are composed of n information units. This assumption allows the application of basic concepts and equations of 3D nucleation (see Appendix A).

(6) The fraction α of items is given as $y(t)/y_{max}$, where $y(t)$ is the cumulative number of items at time t and y_{max} is the maximum number of items.

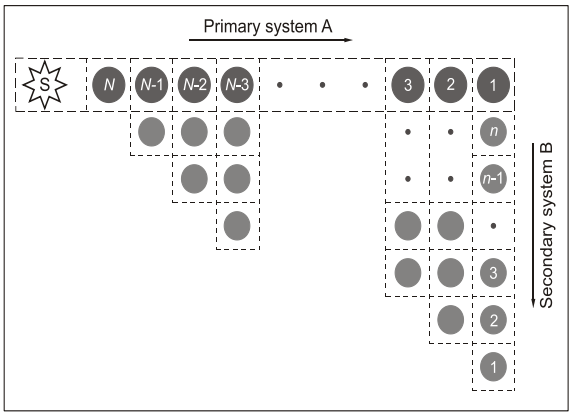


Fig. 8: Schematic illustration of processes of generation of primary items A from a source S and of secondary items B from primary items A acting as secondary sources of (secondary) items. Adapted from Sangwal [54].

In view of the above, upon replacing the fraction $\alpha = V_c(t)/V$ by $y(t)/y_{max}$, the cumulative fraction of items generated at time t from an individual source such as S in Figure 8, from Eq. (B1) of Appendix B one obtains Eq. (1), called hereafter Avrami–Weibull equation, with the time constant Θ and the exponent q given by Eqs. (B2) and (B3) for items generated by instantaneous and progressive nucleation modes. When q is given by Eq. (B3), Eq. (1) represents the progressive nucleation mechanism for the production of items [7,8.54].

It is usually observed that the cumulative fraction $\alpha(t)$ of secondary items such as citations produced by a complex system composed of successively appearing primary items (e.g. papers) from the primary source (e.g. an author) also follows a relation similar to that of Eq. (B1) with new time constant Θ and exponent q. In fact, a modeling experiment, carried out by the present author [39,53] of secondary items such as citations produced by progressive nucleation from a complex system composed of successively appearing primary items (e.g. papers) from a primary source (e.g. author) at equal intervals of time Δ, characterized by different values of time constant Θ and exponent q, showed that Eq. (1) describes equally well the time dependence of cumulative fraction $y(t)/y_{max}$ of secondary items. However, it was observed that the values of both Θ and q increase with increasing duration t of generation of secondary sources of items.

It should be mentioned that during 1939-1941 M. Avrami proposed and popularized Eq. (1). It is known as Weibull function in the literature on microbial growth and provides a physical interpretation of its parameters. It has the form of Weibull distribution function [55]. As in a previous paper [40], in this paper Eq. (1) is referred to as Avrami–Weibull equation to honor its proponents.

From Eq. (1) one may define fraction of generated fraction of items as

$$x = \frac{y_{max} - y(t)}{y_{max}} = \exp\left\{-\left(\frac{t}{\Theta}\right)^q\right\}. \tag{13}$$

This is the so-called extended exponential used by Hirsch [18] to discuss the relationships of the maximally cited papers N_{max} and the total number $N(t_m) = 1$ of papers with at least one citation with the Hirsch index h. However, in the above function the exponent $q \leq 1$. In contrast, depending on the nucleation mode in Eq. (B1), the exponent q can take values both less than and greater than unity.

5. Predictions of Avrami–Weibull function

From analysis of data on the growth behavior of a wide range of material items, such as germination of tomato seeds, growth of bacteria in different media, and overall crystallization from melts, discussed in the preceding sections it may be concluded that Avrami–Weibull function (1) with its two parameters Θ and q explains the data much better than Verhulst (logistic) function (2) and Gompertz function (3) and the temperature dependence of its time constant Θ may be interpreted in terms of the dependence of rate constant k of chemical reactions and nucleation rate J on temperature T according to relations (A3) and (A6), respectively.

Relation (1) also explains the dependence of nucleation rate J on driving force ΔG_R involved in overall crystallization. Although the effect of temperature on the growth behavior of abstract items, such as papers and their citations, is not expected, one finds that the growth behavior of the cumulative number $y(t)$ of abstract items with time t by individual (simple) sources and collectives or groups of sources (complex sources) also follows Avrami–Weibull function (1). Moreover, as discussed in Appendix C, Avrami–Weibull function (1) is superior to Verhulst (logistic) and Gompertz functions in explaining the generation rate $dy(t)/dt$ of items in terms of the parameters Θ and q. In view of these

features, the predictions of Avrami–Weibull function are described below.

5.1. Behavior of growth curves

Figures 9a and 9b illustrate the cumulative fraction $F(t) = y(t)/y_{max}$ of items, produced by individual sources, characterized by different q at $\Theta = 20$ arbitrary units (a.u.) and by different Θ at $q = 2$, respectively, using Eq. (1). As seen from the plots of Figure 9a, irrespective of the value of Θ, curvatures of the curves are concave and convex for $q \leq 1$ and $q > 1$, respectively, but for all q the curves attain $F(t) = 1$ at sufficiently high values of Θ such that the S-shaped curves are obtained only in the latter case of $q > 1$. However, the plots show that their convex curvature increases with increasing values of Θ for a given value of $q > 1$ (Figure 9b) whereas their concave curvature increases with decreasing $q < 1$ for a given value of Θ (Figure 9a). It may be demonstrated easily that a linear dependence is obtained when $t/\Theta \ll 1$ and $q = 1$ in Eq. (1); see Appendix C.

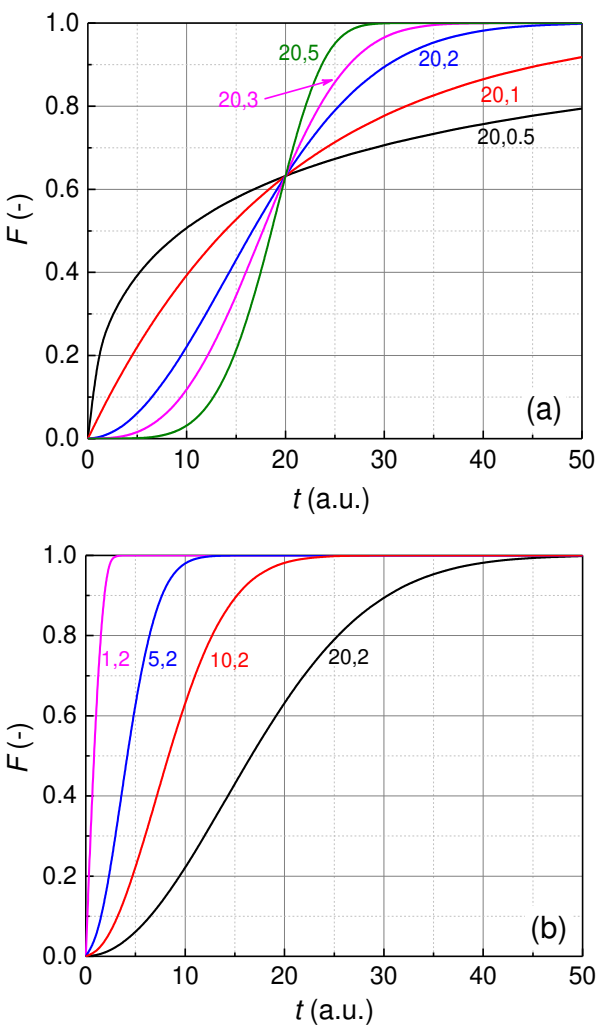


Figure 9: Cumulative fraction $F(t) = y(t)/y_{max}$ of items produced according to Avrami–Weibull function (1) by individual sources characterized by (a) different q at $\Theta = 20$ arbitrary units (a.u.) and (b) different Θ at $q = 2$. Values of Θ and q are indicated as (Θ,q) alongside the plots.

Using citation data of articles written by selected Chinese American Nobel prize winners in physics, Liu and Rousseau [56] reported three types of cumulative citation $L(t)$ plots: (1) *normal* S-shaped plots with initial convex curvature followed by concave curvature, (2) *inverse* S-shaped plots with initial concave curvature followed by convex curvature, and (3) linear plots of $L(t)$ data. Avrami–Weibull function (1) satisfactorily describes the nature of the above curves [40].

5.2. Behavior of growth rate curves

It is frequently observed that the absolute number of items per unit time (e.g. citations per year of an author; also called citation frequency and citation rate), usually defined as $\Delta L = [L(t)–L(t–1)]$ when t is taken in years Y, initially increases, then, after going through a maximum value, slowly decreases and finally attains a zero value with increasing time (for example, see: [8,39,45,57-59]). This phenomenon of slowly decreasing growth of items with time is usually called obsolescence [60-64], aging [63] or decay. In the case of citations of individual articles, typical curves of ΔL against citation time t are of the following types [62]: (a) initially much-praised articles, (b) recognized basic work, (c) scarcely reflected work, (d) well received but later erroneously qualified work, and (e) general work. According to this classification, citation rate curves with steep positive slope are characteristic of initially much-praised articles or articles that recognized basic work. This classification of growth of citation rate curves has been used recently by Ho and Kahn [59] in the discussion of top-cited single-author papers.

The above trends of citation frequency are usually explained by using empirical exponential functions [7,61,63,64]. The main criticism of using these empirical exponential functions to describe their decaying behavior is that they contain parameters to which it is difficult to assign any physical significance. However, as shown in Appendix C, the fitting parameters Θ and q of the Avrami–Weibull function have well-defined meaning.

Figures 10a and 10b show the plots of growth rate $f(t) = dF(t)/dt$ of items corresponding to the cumulative fraction $F(t) = y(t)/y_{max}$ of items of Figure 9. From these plots the following features may be noted:

(1) For a given value of the time constant Θ, with an increase in the value of the exponent q the maximum value of f_c for the items shifts to a higher t such that the value of t_c corresponding to the peak approaches the value of the time constant Θ; see Figure 10a.

(2) For a given value of the exponent q, with an increase in the value of the time constant Θ the maximum value of f_c for the generation of items shifts to a higher t such that the value of t_c corresponding to the peak is lower than the value of the corresponding time constant Θ; see Figure 10b.

(3) For a given set of Θ and q, the area under the plot of $f(t)$ for the items over the entire generation period t

represents the maximum cumulative fraction F_{max} = 1 of the items.

It may be seen from Figure 10 that the distribution of items generated by a source usually has a skew to the right for different sets of Θ and q. However, for a parti cular value of Θ, there is a value of q when the distribution is symmetrical. Conversely, for a particular value of q, there is a value of Θ when the distribution of items is symmetrical. Corresponding to Θ = 20 in Figure 10a, the value of q is about 4 when the distribution of citations is symmetrical. One also expects that the distribution has a skew to the left for q exceeding about 4.

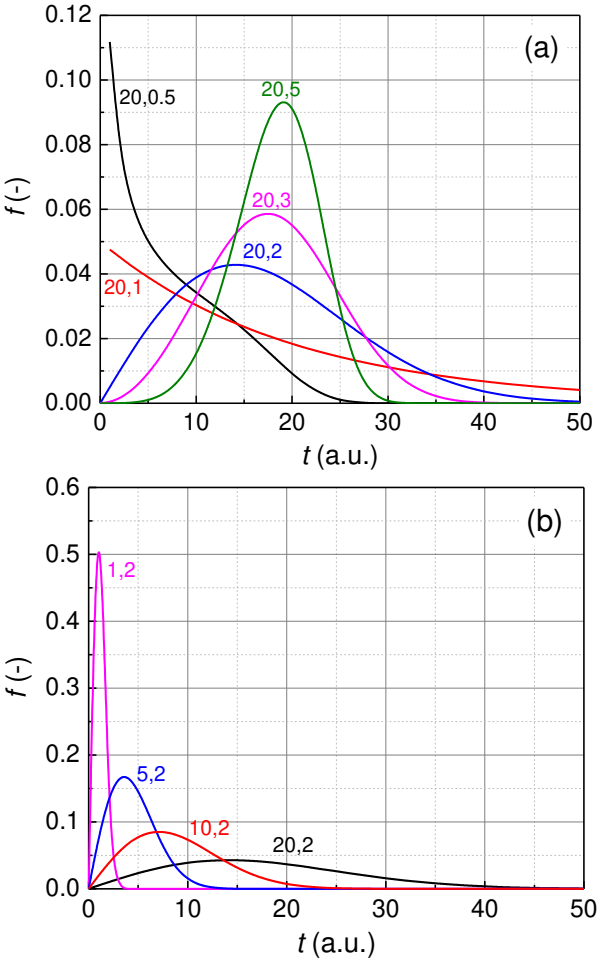


Figure 10: Plots of growth rate $f(t) = dF(t)/dt$ of items produced according to Avrami–Weibull function (1) by individual sources characterized by (a) different q at Θ = 20 arbitrary units (a.u.) and (b) different Θ at q = 2. Values of Θ and q are indicated as (Θ,q) alongside the plots.

Figure 11 shows some typical examples of evolution of yearly citations ΔL of papers published by Laemmli and Cox with time t. In Figure 11a yearly citations of all subsequently papers published by Laemmli are compared with his top-cited paper 1 whereas in Figure 11b yearly citations to his next three top-cited papers are presented. Figure 11c shows yearly citations received by the top-cited two papers of Cox. In these figures the curves are drawn with the values of the

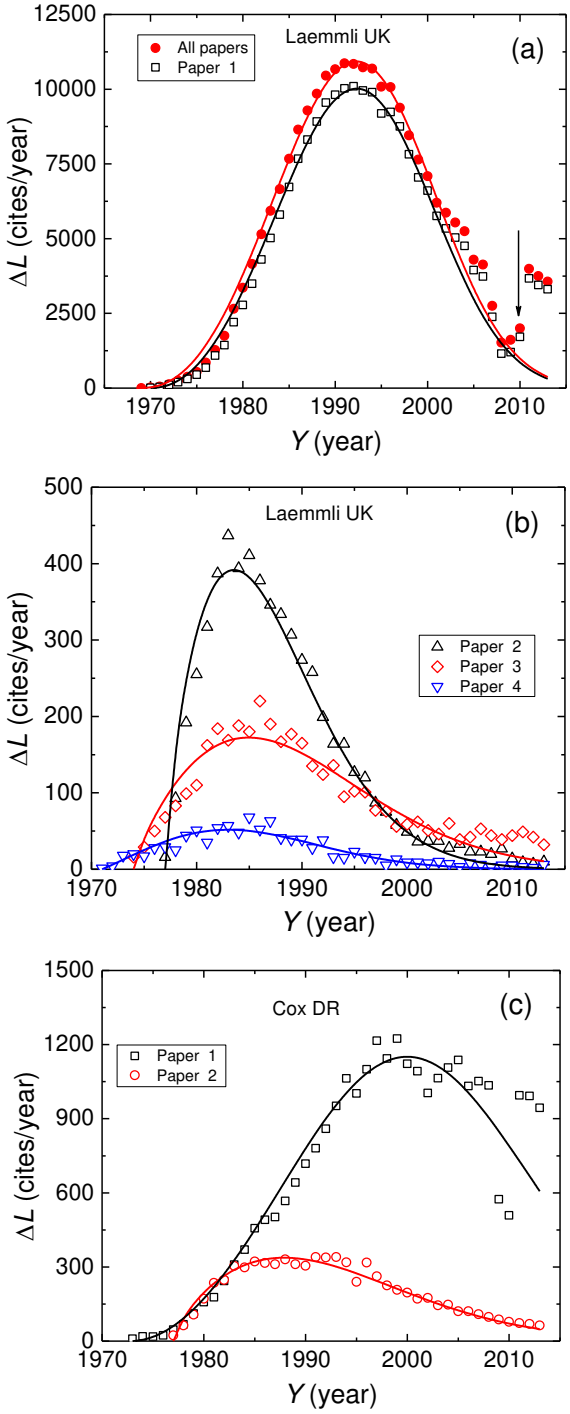


Figure 11: Typical examples of evolution of yearly citations ΔL of papers published by different authors: (a,b) Laemmli and (c) Cox. In (a) citations of all subequently papers published by Laemmli are compared with his top-cited paper 1. Curves are drawn with best-fit parameters given in Table 4.

parameters of Eq. (1) given in Table 3. It may be seen from this table that the total yearly citations of all successively papers mainly come from the top-cited paper 1 and that the exponent q lies between 1 and 4 for all individual papers.

Analysis of growth of citations using Avrami–Weibull function to individual papers published by different authors indeed shows that 1 < q < 4 for most cases. For example, the values of q lies between 1 and 2.7 for

citations to individual 27 top-cited papers of 4 selected Polish professors [8], between 1 and 3.2 for citations to articles by 4 of 5 Chinese American Nobel Prize winners [40], and between 0.9 and 3.5 for citations to 41 of 43 top-cited papers by 12 authors (Sangwal, unpublished results). This suggests that the process of citations received by individual articles is mainly determined by progressive nucleation mode involving both diffusion and integration of published knowledge (cf. Appendix B).

6. Summary and conclusions

Some examples of empirical data of sigmoidal-shaped $y(t)$ growth of different types of real and abstract items are analyzed by Verhulst, Gompertz and Avrami–Weibull functions. It was found that Avrami–Weibull function describes the growth behavior of different types of items better than Gompertz and Verhulst functions. Moreover, in comparison with Verhulst (logistic) and Gompertz functions, Avrami–Weibull function is relatively simple and mathematically convenient for explaining different trends of the generation rate $dy(t)/dt$ of scientific literature in terms of its two parameters: the time constant Θ and the exponent q.

Employing the basic concepts of occurrence of chemical reactions between reactants producing reaction products and overall crystallization of solid phase in melts and solutions (Appendices A and B), Avrami–Weibull function (1) for the growth behavior of cumulative number $y(t)$ items produced at time t by individual (simple) sources and collectives or groups of sources (complex sources) is presented. Then the data of germination of tomato seeds, growth of various bacteria in different media and evolution of overall crystallization of polypropylene are analyzed from the standpoint of occurrence of chemical reactions and crystallization from melts and solutions.

It is observed that every plot of the growth of cumulative number $y(t)$ of items against time t is characterized by three parameters: dimensionless growth rate $R = \mathrm{d}y/y_{\max}\mathrm{d}t$ determined from the linear part of a $y(t)/y_{\max}$ plot, time lag t_0 corresponding to the onset of initial growth, and time constant Θ when the rate R reaches a maximum value. In the case of growth of material items such as bacteria, the growth rate R increases whereas the time lag t_0 and the time constant Θ decrease with increasing temperature T. However, an opposite trend is observed in overall crystallization from the melt. In this case, the crystallization rate R decreases but the time lag t_0 and the time constant Θ increase with crystallization temperature T. The former processes are associated with chemical reactions which follow simple Arrhenius-type relation (A3) in which the cumulative volume (concentration) of reaction products decreases with an increase in ΔG_R, whereas overall crystallization follows relation (A8) in which the cumulative volume of crystallized phase increases with an increase in ΔG_R. In the latter case, the parameters R, t_0 and Θ are related to the dependence of the number N_m of nuclei formed or the nucleation rate J on driving force ΔG_R.

There exists a cognitive citation pressure ΔG_R as the driving force for the citation received by a paper [51,52]. The mechanisms and processes of motivations for citation have been discussed in the scientometric research [69-71]. The cognitive citation pressure ΔG_R is essentially associated with the attractiveness of a paper to be cited. The higher the attractiveness of a paper, the higher is its citability. Comparison of the process of receiving of citations by papers with the processes of occurrence of chemical reactions and crystallization of solid phases from melts and supersaturated solutions in terms of this driving force ΔG_R shows that the process of citations of papers is similar to that of overall crystallization of solid phases. The process of overall crystallization involves the formation of nuclei of crystallized phase instantaneously or progressively with time in a fixed volume. Using an analogy with overall crystallization, it is argued that similar processes occur during the citation of papers. However, citation process differs from overall crystallization. Overall crystallization of solid phase depends on crystallization temperature but no effect of temperature can be conceived in citation process.

Analysis of growth of citations using Avrami–Weibull function to individual papers published by different authors shows that $1 < q < 4$ for most cases. This suggests that the process of citations received by individual articles is mainly determined by progressive nucleation mode involving both diffusion and integration of published knowledge.

Finally, it should be emphasized that, despite arguments in favor of similarity of citations of papers and other information production processes with overall crystallization presented here for their explanation by Avrami–Weibull function, all workers in the field of informetrics may not be convinced. However, the relatively simple and mathematically convenient form of this function does deserve due attention to describe the dynamics of growth of citations of papers and other similar processes.

Appendix A. Occurrence of chemical reactions and crystallization

The driving force for the occurrence of any chemical reaction is the difference ΔG_R in the Gibbs free energy G_I of initial reactants, say A and B, and the free energy G_{II} of the products, say C and D, and may be expressed by the relation

$$\mathrm{A} + \mathrm{B} \rightarrow \mathrm{X}^* \rightarrow \mathrm{C} + \mathrm{D} + \Delta G_R, \qquad \text{(A1)}$$

where X^* is a transient, activated complex of free energy G^* higher than G_I of the reactants (see Figure A1). The rate of formation of products is determined by the values of these three free energies. The formation of products is possible when the free energy change $\Delta G_R > 0$, which is the necessary driving forces for the reaction, but the rate of the reaction is determined by the relative

increase in the free energy ΔG_a ($\Delta G_a = G^* - G_I$) involved in the formation of the activated complex X^*. The higher the free energy change ΔG_a required for the formation of the activated complex, the more difficult it is for the reaction to occur. Similarly, the greater the value of ΔG_R, the higher is the stability of the reaction products.

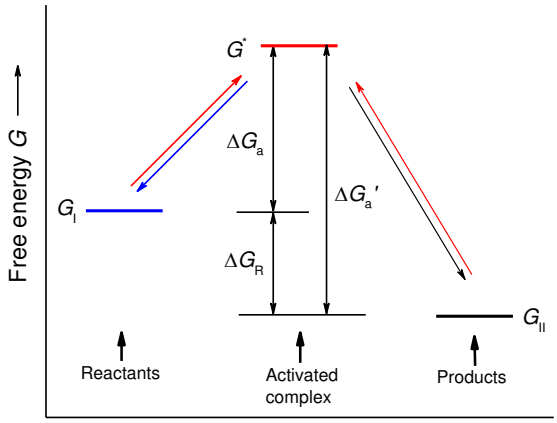


Figure A1: Schematic presentation of free energy changes associated with a chemical reaction.

The rate of the reaction (A1) may be given by (see ref. [47]; Chap. 14)

$$\text{rate} = \frac{dC_{\text{Pr}}}{dt} = kC_A C_B\,, \tag{A2}$$

where C_A, C_B and C_{Pr} are concentrations of A, B and products, and k is the rate constant. According to the transition state theory of reaction rates, the rate constant

$$k = \frac{\text{rate}}{C_A C_B} = A\exp\left(-\frac{\Delta G_R^0}{R_G T}\right), \tag{A3}$$

where ΔG_R^0 is the difference between energy of reactants and complex, all in their respective ground states, the pre-exponential factor A is frequently called the frequency factor, R_G is the gas constant, and T is the temperature in Kelvin.

According to Eq. (A3) the rate constant k for a chemical reaction involving a constant energy change ΔG_R^0 increases with temperature T. Eq. (A3) is similar in form to the Arrhenius equation relating the temperature dependence of rate constant k with activation energy ΔG_R^0 for a chemical reaction. An Arrhenius-type equation also holds for diffusion and fluidity processes in solutions [65].

As in the case of chemical reactions, crystallization from melts and solutions is also associated with two energy changes: an activation barrier ΔG_{3D}^* for the formation of three-dimensional (3D) clusters or nuclei in the liquid phase, given by Eq. (A7), and an overall change in the free energy ΔG_R, given by Eq. (A1). The value of the activation barrier ΔG_{3D}^* is associated with the formation of 3D clusters by aggregation of growth entities (e.g. atoms, ions and molecules) present in the liquid, whereas the overall change in the free energy ΔG_R determines whether the 3D clusters formed in the growth system remain stable after their formation. Note that the free energy change ΔG_R is a measure of deviation from equilibrium state when $\Delta G_R = 0$, and $\Delta G_R > 0$ for supersaturated solutions. While discussing the process of formation of clusters it is usually assumed that they prefer to attain a rounded shape of radius r because the surface tension γ of a sphere is the lowest. However, in view of analysis of the number of items in this paper we discuss the process of formation of these clusters in the medium in terms of free energy change as a function of the number n of atoms, ions or molecules comprising them.

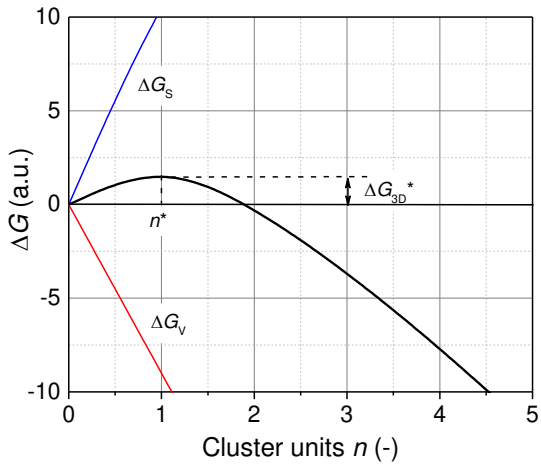


Figure A2: Change in Gibbs free energy ΔG as a function of size n of clusters forming in a supersaturated medium. Values for surface energy $\gamma = 11.25$ and driving force $\Delta G_R = 9$ are chosen to obtain critical nucleus with $n = 1$. ΔG_R and γ are expressed in the same arbitrary energy units (a.u.). Curves of energy contributions ΔG_S and ΔG_V are also shown.

The reduction in the Gibbs free energy ΔG of a system due to the formation of a 3D cluster composed of n growth units is equal to the sum of the surface excess free energy ΔG_S and the volume excess free energy ΔG_V. The resulting free energy change may be given by [66,67]

$$\Delta G = \Delta G_S + \Delta G_V = n^{2/3}\gamma - n\Delta G_R\,, \tag{A4}$$

where γ is a surface-energy term. The two terms in the right-hand side of Eq. (A4) depend differently on n. This behavior of ΔG associated with the formation of the cluster is shown in Figure A2 as a function of its size n. It may be seen from the figure that ΔG passes through a maximum and the maximum value ΔG_{3D}^* corresponds to the critical size n^*. The value of n^* may be obtained by maximizing Eq. (A4), taking $d\Delta G/dn = 0$, i.e.

$$\frac{d\Delta G}{dr} = \frac{2}{3}n^{-1/3}\gamma - \Delta G_R = 0\,, \tag{A5}$$

or

$$n^* = \left(\frac{2}{3}\frac{\gamma}{\Delta G_{\mathrm{R}}}\right)^3 . \tag{A6}$$

Upon substituting the value of n^* from Eq. (A6) into Eq. (A4), one obtains the energy barrier

$$\Delta G^*_{3\mathrm{D}} = \frac{4}{27}\frac{\gamma^3}{(\Delta G_{\mathrm{R}})^2} = \frac{1}{3}n^{*2/3}\gamma . \tag{A7}$$

Note that the value of $\Delta G_{3\mathrm{D}}{}^*$ is always a positive quantity. Obviously, increasing free energy difference ΔG_{R} and decreasing interfacial energy γ facilitate the formation of 3D clusters.

Occurrence of a crystalline phase in a supersaturated medium depends on the size n of the nuclei. When $n < n^*$, the nuclei dissolve. However, when $n > n^*$, the nuclei are stable and grow. The critical size n^* is the minimum size of a stable nucleus. As seen from Figure A2, only when the nucleus size $n > n^*$, the free energy ΔG for the formation of a nucleus decreases with an increase in its size n.

At constant temperature and supersaturation conditions, the occurrence of 3D cluster formation (or nucleation) is described by the so-called stationary nucleation rate J, given by [66]

$$J = J_0 \exp\left(-\frac{\Delta G^*_{3D}}{R_{\mathrm{G}}T}\right) = J_0 \exp\left(-\frac{B}{(\Delta G_{\mathrm{R}})^2}\right), \tag{A8}$$

where J_0 is a kinetic factor and is associated with the frequency of attachment of basic units to the nucleus at equilibrium, $\Delta G_{3\mathrm{D}}{}^*$ is given by Eq. (A7) and $B = 4\gamma^3/27R_{\mathrm{G}}T$.

Eq. (A8) represents the temperature dependence of nucleation rate J, and is usually referred to as the classical theory of 3D nucleation. Obviously, it is an Arrhenius-type relation where the activation barrier $\Delta G_{3\mathrm{D}}{}^*$ is essentially a measure of the "difficulty" for atomic/molecular aggregates to attain the size n^* of the stable clusters in a growth medium.

It should be mentioned that the free energy difference ΔG_{R} is not a temperature independent quantity, and is given by (for example, see: [36,67])

$$\Delta G_{\mathrm{R}} = \left(\frac{\Delta H_{\mathrm{m}}}{T_{\mathrm{m}}}\right)(T_{\mathrm{m}} - T) , \tag{A9}$$

where ΔH_{m} is the enthalpy of melting of the compound, T_{m} is the its melting point, and T is the crystallization temperature. Obviously, crystallization from the melt is possible when the temperature difference $\Delta T = (T_{\mathrm{m}}-T) > 0$. Then the nucleation rate (for example, see: [36,67])

$$J = J_0 \exp\left(-\frac{B'}{T(\Delta T)^2}\right), \tag{A10}$$

with the constant

$$B' = \frac{\kappa}{\phi^2}\left(\frac{\gamma}{R_G}\right)^3 , \tag{A11}$$

where κ is the geometrical factor for the shape of clusters (here $\kappa = 4/27$), and the parameter $\phi = \Delta H_{\mathrm{m}}/R_{\mathrm{G}}T_{\mathrm{m}}$ is about 2 for metals, 3 for anhydrous inorganic salts, and 6 for organic compounds.

Eq. (A10) differs fundamentally from Arrhenius-type relation (A3) in the T-term. Under crystallization conditions, in Eq. (A10) the crystallization temperature $T \approx T_{\mathrm{m}}$ but the temperature difference $\Delta T = (T_{\mathrm{m}}-T)$ rapidly increases with small increases in T. Therefore, the nucleation rate J essentially depends on ΔT instead of T, with an activation energy B' in the plots of $\ln J$ against $T(\Delta T)^2$.

Appendix B: Basic concepts of overall crystallization

Description of overall crystallization of a solid phase in the volume of a melt or supersaturated solution is one of the various applications of the theory of nucleation in the field of crystal growth. The theory of overall crystallization is based on the following concepts [66]:

(1) The fraction α of solid phase is the total volume $V_{\mathrm{c}}(t)$ of solid phase crystallized after time t in the initial fixed volume V of the crystallizing system: $\alpha = V_{\mathrm{c}}(t)/V$.
(2) The volume $V_{\mathrm{c}}(t)$ of the solid phase is formed by its nucleation at material points at a rate $J(t)$ in the volume V of the crystallizing medium (melt or supersaturated solution) and each nucleus grows independently of the other nucleating and growing crystallites. This means that crystallites can be nucleated only in the noncrystallized volume $V-V_{\mathrm{c}}$ of the crystallizing medium.
(3) Nuclei can form on active centers in the medium either instantaneously at $t = 0$ or progressively during the entire crystallization process, thereby determining the time dependence of nucleation on active centers and finally the overall crystallization of the solid phase resulting from the growth of the nuclei in the liquid phase. These types of growth are known to occur by instantaneous and progressive nucleation modes, respectively, and are illustrated in Figure B1.

With the above ideas of the fraction $\alpha(t)$ of solid phase crystallized at time t, the total volume $V_{\mathrm{c}}(t)$ of the solid phase crystallized after time t in the initial fixed volume V of the crystallizing system by instantaneous and progressive nucleations may be given by the unified relation [66]

$$\alpha(t) = \frac{V_{\mathrm{c}}(t)}{V} = 1 - \exp\left\{-\left(\frac{t}{\Theta}\right)^q\right\}, \tag{B1}$$

where Θ is a time constant, and the exponent $q > 0$. The exponent q and the time constant Θ are given by

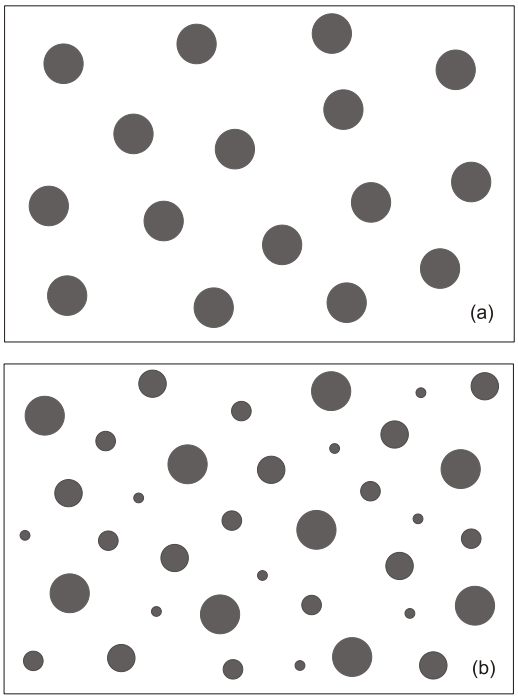


Figure B1: Schematic illustration of (a) instantaneous and (b) progressive formation of nuclei of a solid phase at active centers in a fixed volume of melt or supersaturated solution.

$$q = \nu d, \quad \Theta = \frac{1}{g}\left(\frac{V}{\kappa N_{\mathrm{m}}}\right)^{1/\nu d}, \tag{B2}$$

for instantaneous nucleation, and

$$q = 1 + \nu d\ ,\ \Theta = \left(\frac{1+\nu d}{\kappa g^{\nu d} J_{\mathrm{s}}}\right)^{1/(1+\nu d)} \tag{B3}$$

for progressive nucleation. In Eqs. (B2) and (B3) N_{m} is the maximum number of nucleation centers, J is the rate of stationary nucleation given by Eq. (A8), κ is the shape factor for the nuclei (for example, $\kappa = 4\pi/3$ for spherical nuclei) and the growth constant g is defined by

$$g = \frac{r^{1/\nu}}{t}, \tag{B4}$$

where r is the radius of the growing nucleus and the constant $\nu > 0$ is a number. Eq. (B4) describes the dependence of the radius r of the growth of *individual* nuclei on time t according to the traditional power-law relation

$$r(t) = Zt^{\nu}, \tag{B5}$$

where the values of ν are 1/2 and 1 for growth controlled by volume diffusion and interface transfer, respectively, and $Z = g^{\nu}$. In Eqs. (B2) and (B3) the parameter d denotes the dimensionality of growing nuclei. For nuclei growing in one-, two- and three-dimensions, d = 1, 2 and 3, respectively. However, when the nuclei do not grow, $d = 0$.

Aggregation of growth entities into nuclei and their subsequent growth into stable entities involves diffusion of growth entities to active nucleation centers present in the melt or solution volume and integration of these growth units into the surface of the nuclei. Therefore, the value of the exponent q in Eqs. (A2) and (A3) depends on the values of the exponent ν, the dimensionality d and the nucleation mode. In the case of instantaneous nucleation mode, $0 < q < 1.5$ and $0 < q < 3$ for crystallization controlled by volume diffusion and interface integration, respectively. However, when crystallization occurs by progressive nucleation mode, $1 < q < 2.5$ and $1 < q < 4$ for crystallization controlled by volume diffusion and interface integration, respectively. The lowest values of 0 and 1 for q, corresponding to crystallization by instantaneous and progressive nucleation mechanisms, respectively, are obtained when the nuclei do not grow with t.

Note that the exponent q is a non-integer parameter in diffusion-controlled crystallization but it is usually found that it also has non-integer values even in mass-transfer-controlled crystallization due to different assumptions used in the derivation of Eq. (B1). The time constant Θ is determined by the growth constant g and either by the maximum number N_{m} of crystallites in instantaneous nucleation mechanism (Eq. (B2)) or by stationary nucleation rate J_{s} in progressive nucleation mechanism (Eq. (B3)).

The above treatment is usually called the Kolmogorov–Johnson–Mehl–Avrami (KJMA) theory. However, in the literature on crystallization of fats it is also known as the Avrami equation in which mass instead of volume is used and the exponential term $(t/\Theta)^q = kt^q$, where k is the Avrami constant and q is the Avrami exponent [68]. Theoretical aspects of this theory for overall crystallization are discussed by Kashchiev [66].

It should be mentioned that real kinetic rate laws are not as simple as described by Eq. (A2) where the rate of formation of reaction product is directly related to the concentrations C_{A} and C_{B} of reactants A and B. For example, there are consecutive reactions in which intermediates interact with the reactants and the products and in which various types of elementary reactions and their combinations take place [49]. In this sense, the Avrami constant k is a complex constant of qth order.

Appendix C. Comparison of predictions of different functions

All of the above three functions relating cumulative distribution function $F(t) = y(t)/y_{\max}$ with t, described in Section 2, are two-parameter functions but they predict different trends of the plots of $y(t)$ and $dy(t)/dt$ against time t. Avrami–Weibull and Gompertz functions predict $y(t) = 0$ at $t = 0$ but according to Verhulst function $y(t) = y_0 > 0$ at $t = 0$. This means that Verhulst function is not expected to explain satisfactorily situations of empirical data with $y(t) = 0$ at $t = 0$. However, the three functions predict an initial increase in $dy(t)/dt$, followed by a subsequent decrease, with t such that $dy(t)/dt$ attains a max-

imum value $[dy(t)/dt]_c$ at a particular value of t, say t_c. The trends of the dependence of $dy(t)/dt$ on t predicted by the three functions are different. The value of t_c when $dy(t)/dt$ attains its maximum value may be obtained by maximizing Eqs. (1), (2) and (3).

In the case of Avrami–Weibull relation (1) the rate R of generation $y(t)$ of items with t may be given by

$$R = \frac{dy(t)}{dt} = y_{\max}\frac{q}{\Theta^q}t^{q-1}\exp\left\{-\left(\frac{t}{\Theta}\right)^q\right\}. \quad \text{(C1)}$$

Eq. (C1) predicts an initial increase in the generation rate R of items according to power law and, after reaching a maximum value at time t_c, it decreases exponentially. From Eq. (C1) one has

$$\frac{dR}{dt} = \frac{d^2y(t)}{dt^2} = y_{\max}\frac{q}{\Theta^q}t^{q-1}\exp\left\{-\left(\frac{t}{\Theta}\right)^q\right\}\left[\frac{q-1}{t} - \frac{q}{\Theta^q}\right]$$
$$= \frac{dy(t)}{dt}\left[\frac{q-1}{t} - \frac{q}{\Theta^q}t^{q-1}\right] = 0, \quad \text{(C2)}$$

which gives

$$t_c = \left(\frac{q-1}{q}\right)^{\frac{1}{q}}\Theta. \quad \text{(C3)}$$

Substitution of t_c from Eq. (C3) into Eq. (C2) gives the maximum rate $R_c = (dy(t)/dt)_c$ in the form

$$R_c = \left(\frac{dy(t)}{dt}\right)_c = y_{\max}\frac{q}{\Theta}\left(\frac{q-1}{q}\right)^{\frac{q-1}{q}}\exp\left[-\left(\frac{q-1}{q}\right)\right]. \quad \text{(C4)}$$

According to the above equations t_c and R_c depend on the parameters q and Θ. Note that a maximum rate R_c is not achieved when $0 < q < 1$ (see Figure 10). Then $t_c = 0$ and $(dy(t)/dt)_c = 0$ (see Eqs. (C3) and (C4)).

According to Verhulst function (2) the items' generation rate R with time t is given by

$$R = \frac{dy(t)}{dt} = \beta y(t)\left(1 - \frac{y(t)}{y_{\max}}\right). \quad \text{(C5)}$$

From (C5) one obtains

$$\frac{dR}{dt} = \frac{d^2y(t)}{dt^2} = \beta\left(1 - 2\frac{y(t)}{y_{\max}}\right) = 0, \quad \text{(C6)}$$

which gives

$$y(t_c) = \frac{y_{\max}}{2}. \quad \text{(C7)}$$

Substitution of $y(t_c)$ from Eq. (C7) into Eq. (C5) gives the maximum R_c in the form

$$R_c = \left(\frac{dy(t)}{dt}\right)_c = \frac{\beta y_{\max}}{4}. \quad \text{(C8)}$$

Obviously, R_c depends only on the value of the parameter β and is independent of y_0 and time t.

In the case of Gompertz function (3) the items' generation rate

$$R = \frac{dy(t)}{dt} = y_{\max}\lambda(\exp ct)\exp\left\{-\frac{\lambda}{c}(\exp ct - 1)\right\}. \quad \text{(C9)}$$

Eq. (C9) predicts both increase and decay in $y(t)$ with time t following exponential dependences, exhibiting maximum R_c at t_c. From (C9) one has

$$\frac{dR}{dt} = \frac{d^2y(t)}{dt^2} = y_{\max}\lambda^2\exp(2ct)\exp\left\{-\frac{\lambda}{c}(\exp ct - 1)\right\} = 0, \quad \text{(C10)}$$

which gives

$$t_c = \frac{1}{c}\ln\left(1 - \frac{c}{\lambda}\right). \quad \text{(C11)}$$

Substitution of t_c from Eq. (C11) into Eq. (C9) gives the maximum R_c in the form

$$R_c = y_{\max}\lambda\exp\left(1 - \frac{c}{\lambda}\right). \quad \text{(C12)}$$

According to the above equations t_c and R_c depend on the parameters c and λ. Note that a maximum value of R is not achieved when $c \ll \lambda$. Then $t_c = 0$ and $R_c = 2.718y_{\max}\lambda$ (cf. Eqs. (C11) and (C12)).

According to power-law relation (3), the items generation rate R may be given by

$$R = \frac{dy(t)}{dt} = y_{\max}\frac{q}{\Theta^q}t^{q-1}. \quad \text{(C13)}$$

Obviously, the rate R is also expected to follow power-law dependence. Similarly, for $ct \ll 1$, Gompertz function (3) gives

$$R = y_{\max}\lambda\exp\{(c-\lambda)t\}. \quad \text{(C14)}$$

Depending on the value of $c > \lambda$, Eq. (C14) predicts an exponential increase in R with t. For $q = 1$ and $c = \lambda$, (C13) and (C14) reduce to the form

$$R = y_{\max}\lambda, \quad \text{(C15)}$$

with $\lambda = 1/\Theta$.

In summary, although Verhulst and Gompertz functions (2) and (3) predict the appearance of maximum rate R_c, their parameters β, and c and λ do not give any insight into the processes involved in the growth of items. In contrast to them, Avrami–Weibull relation (1) not only explains the growth of items with time t better than the Verhulst and Gompertz functions as well as maximum values of R_c through its parameters Θ and q but these parameters have physical meaning.

Acknowledgements

The author is grateful to Dr Kazimierz Wójcik for his assistance with the collection of the data analyzed in this work and for preparing Figures 8 and B1. The author also expresses his gratitude to Prof. Bożena Gładyszewska for the original data used in this study.

References

[1] D.D.S. Price, Little Science, Big Science. Columbia University Press, New York & London, 1963.
[2] L. Egghe, I.K. Ravichandra Rao, Classification of growth models based on growth rates and its applications, Scientometrics 25 (1992) 5-46.
[3] B.M., Gupta, L. Sharma, C.R. Karisiddappa, Modeling the growth of papers in a scientific speciality, Scientometrics 33(2), (1995) 187-201.
[4] B.M., Gupta, S. Kumar, S.L. Sangam, C.R. Karisiddappa, Modeling the growth of social science literature, Scientometrics 53(1), (2002) 161-164.
[5] I.K. Ravichandra Rao, D. Srivastava, Growth of journals, articles and authors in malaria research, Journal of Informetrics 4(1), (2010) 249-256.
[6] C.-Y. Wong, K.-L. Goh, Growth behavior of publications and patents: A comparative study on selected Asian economies, Journal of Informetrics 4(2), (2010) 460-474.
[7] K. Sangwal, Progressive nucleation mechanism and its application to the growth of journals, articles and authors in scientific fields, Journal of Informetrics 5(4), (2011) 529-536.
[8] K. Sangwal, Application of progressive nucleation mechanism for the citation behavior of individual papers of different authors,. Scientometrics 92(2), (2012) 643-655.
[9] Q.L. Burrell, The nth-citation distribution and obsolescence, Scientometrics 53(3), (2002) 309-323.
[10] Q.L. Burrell, Hirsch's *h*-index: A stochastic model, Journal of Informetrics 1(1), (2007) 16-25.
[11] Q.L. Burrell, On the *h*-index, the size of the Hirsch core and Jin's A-index, Journal of Informetrics 1(2), (2007) 170-177.
[12] Q.L. Burrell, The individual author's publication–citation process: theory and practice, Scientometrics 98(1), (2014) 725-742.
[13] W. Glänzel, On the possibility and reliability of predictions based on stochastic citation processes, Scientometrics 40(3), (1997) 481-492.
[14] W. Glänzel, A. Schubert, Predictive aspects of a stochastic model for citation processes, Information Processing and Management 31(1), (1995) 69-80.
[15] X. Zheng, Predicting publication productivity for researchers: A piecewise Poisson model, Journal of Informetrics 14(3), (2020) 101065.
[16] S.Nadarajan, S. Kotz, Models for citation behavior, Scientometrics 72(2), (2007) 291-305.
[17] M.V. Simkin, V.P. Roychowdhury, A mathematical theory of citing, Journal of American Society for Information Science and Technology 58(11), (2007) 1661-1673.
[18] J.E. Hirsch, An index to quantify an individual's scientific research output, Proceedings of the National Academy of Sciences of the USA 102(46), (2005), 16569-16572.
[19] A. Fernandez, C. Salmeron, P.S. Fernandez, A. Martinez, Application of a frequency distribution model to describe the thermal inactivation of two strains of Becillus cereus, Trends in Food Science and Technology 10(4-5), (1999) 158-162.
[20] J.-C. Augustin, A. Brouillaud-Delattre, L. Rosso, V. Carlier, Significance of Inoculum size in the lag time of Listeria monocytogenes, Applied and Environmental Microbiology 66(4), (2000) 1706-1710.
[21] M. Valero, S. Leontidis, P.S. Fernandez, A. Martinez, M.C. Salmeron, Growth of Bacillus cereus in natural and acidified carrot substrates over the temperature range 5-30°C, Food Microbiology 17(6), (2000) 605-612.
[22] H. Fujizawa, A. Kai, S. Morozumi, A new logistic model for bacterial growth, Journal of the Food Hygienic Society of Japan 44(3), (2003) 155-160.
[23] H. Fujizawa, A. Kai, S. Morozumi, A new logistic model for Escherichia coli growth at constant and dynamic temperatures. Food Microbiology 21(5), (2004) 501-509.
[24] M.G. Corradini, M. Peleg, A Weibullian model for microbial injury and mortality, International Journal of Food Microbiology 119(3), (2007) 319-328.
[25] M. Peleg, M.G. Corradini, M.D. Normand, The logistic (Verhulst) model for sigmoidal microbial growth curves revisited, Food Research International, 40(7), (2007) 808-818.
[26] G.T., Yates, T. Smotzer, On the phase lag and initial decline of microbial growth curves, Journal of Theoretical Biology 244(3), (2007) 511-517.
[27] G.M.F., Aragao, M.G., Corradini, M.D., Nonmand, M. Peleg, Evaluation of the Weibull and log normal distribution functions as survival models of Escherichia coli under isothermal and non-isothermal conditions, International Journal of Food Microbiology 119(3), (2007) 243-257.
[28] M.Y. Li, X.M. Sun, G.M. Zhao, X.Q. Huang, J.W. Zhang, W. Tian, Q.H. Zhang, Comparison of mathematical models of lactic acid bacteria growth in vacuum-packaged raw beef stored at different temperatures, Journal of Food Science 78(4), (2012) M600-M604.
[29] J. Kowalik, A. Lobacz, Development of a predictive model for describing the growth of *Yersinia enterocolitica* in Camembert-type cheese, International Journal of Food Science and Technology 50(3), (2015) 811-818.
[30] A. Lobacz, J. Kowalik, A predictive model for *Listeria monocytogenes* in UHT diary products with various fat content during storage, Journal of Food Safety 35(1), (2015) 119-277.
[31] S. Sakanoue, Extended logistic model for growth of single-species population, Ecological Modelling 205(1-2), (2007) 159-168.
[32] H. Krug, G. Taubert, Practical use of the logistic law in experimental tumor-growth, Archiv für Geschwulstforsch 55(4), (1985) 235-244.
[33] U. Foryś, A. Marciniak-Czochra, Logistic equations in tumour growth modelling, International Journal of Applied Mathematical Computation Science 13(3), (2003) 317-325.
[34] B. Gładyszewska, Ocena wpływu przedsiewnej laserowej biostymulacji nasion pomidorów na proces ich kiełkowania (Evaluation of presowing laser biostymulation of tomato seeds on the process of their germination), PhD thesis, Agriculture Academy, Lublin (1998).
[35] W. Kloek, P. Walstra, T. van Vliet. Crystallization kinetics of fully hydrogenated palm oil in sunflower oil mixtures, Journal of American Oil Chemistry Society 77(4), (2000) 389-398.
[36] S. Padar, S.A.K. Jeelani, E.J. Windhab, Crystallization kinetics of cocoa fat systems: Experiments and modeling,

Journal of American Oil Chemists' Society 85(12), (2009) 1115-1126.
[37] K. Sangwal, K. Sato, Nucleation and crystallization kinetics of fats. In: A.G. Marangoni (Editor), Structure-function analysis of edible fats, AOCS Press, Urbana, 2012, Chapter 2, pp. 25-78.
[38] L.M. Cunha, F.A.R. Oliveira, J.C. Oliveira, Optimal experimental design for estimating the kinetic parameters of processes described by the Weibull probability distribution function, Journal of Food Engineering 37(1), (1998) 175-191.
[39] K. Sangwal, Growth dynamics of citations of cumulative papers of individual authors according to progressive nucleation mechanism: concept of citation acceleration, Information Processing and Management 49(4), (2013) 757-772.
[40] K. Sangwal, On the growth dynamics of citations of articles by some Nobel Prize winners, Journal of Informetrics 9 (2015) 466-476.
[41] M.F. Brilhante, M.I. Gomes, D. Pestana, Extensions of Verhulst model in population dynamics and extremes. Chaotic Modeling and Simulation (CMSIM), Issue 4, (2012) 575-591.
[42] T. Chatterjee, B.K. Chatterjee, D. Majumdar, P. Chakrabarti, Antibacterial effect of silver nanoparticles and the modeling of bacterial growth kinetics using a modified Gompertz model, Biochimica et Biophysica Acta 185(2), (2015) 299-306.
[43] A. El-Gohary, A. Alshamrani, A.N. Al-Otaibi, The generalized Gompertz distribution, Applied Mathematical Modelling 37(1), (2013) 13-24.
[44] A.A. Jafari, A. Tahmasebi, M. Alizadeh, The beta-Gompertz distribution, Revista Colombiana de Estadistica, 37(1), (2014) 139-156.
[45] K.Y. Chuang, Y.S. Ho, Bibliometric profile of top-cited single-author articles in the Science Citation Index Expanded, Journal of Informetrics 8 (2014) 951-962.
[46] G. Leubner-Metzger, Functions and regulation of β-1,3-glucanases during seed germination, dormancy release and after-ripening, Seed Science Research 13(1), (2003) 17–34.
[47] K. Weitbrecht, K, Müller, G. Leubner-Metzger, First off the mark: Early seed germination, Journal of Experimental Botany 62(10), (2011) 3289–3309.
[48] M.A. Lopez-Manchado, J. Biagiotti, L. Torre, J.M. Kenny, Effects of reinforcing fibers on the crystallzation of polypropylene, Polymer Engineering Science 40(10), (2000) 2194-2204.
[49] D.F. Eggers, N.W. Gregory, G.D. Halsey, B.S. Rabinovitch, Physical Chemistry. Wiley, New York, 1964.
[50] J.W. Atkinson, An Introduction to Motivation. Van Nostrand, Princeton, 1964.
[51] P. Vinkler, A quasi-quantitative citation model, Scientometrics 12 (1987) 47–72.
[52] P. Vinkler, Comparative investigation of frequency and strength of motives towards referencing: The reference threshold model, Scientometrics 43(1) (1998) 107-127.
[53] K. Sangwal, Progressive nucleation mechanism for the growth behavior of items and its application to cumulative papers and citations of individual authors, Scientometrics 92(2), (2012) 575-591.
[54] K. Sangwal, Czochralski method of crystal growth in the scientific literature: An informetric study, Acta Physica Polonica B 124(2), (2013) 173-180.
[55] W. Weibull, A statistical distribution function of wide applicability, Journal of Applied Mechanics 18(3), (1951) 293–297.
[56] Y.X. Liu, R. Rousseau, Citation analysis and the development of science: A case study using articles by some Nobel Prize winners, Journal of American Society for Information Science and Technogy 65(2), (2014) 281-289.
[57] K. Sangwal, On the growth of citations of publication output of individual authors, Journal of Informetrics 5(4), (2011) 554-564.
[58] K. Sangwal, On the growth behavior of yearly citations of cumulative papers of individual authors, Journal of Scientometric Research 2(1), (2013) 30-36.
[59] Y.-S. Ho, M. Kahn, A bibliometric study of highly cited reviews in the Science Citation Index Expanded^{TM}, Journal of American Society for Information Science and Technology 65(2), (2014) 372-385.
[60] E.S. Aversa, Citation patterns of highly cited papers and their relationship to literature aging: A study of the working literature, Scientometrics 7(3-6), (1985) 383-389.
[61] A. Avramescu, Actuality and obsolescence of scientific literature, Journal of American Society for Information Science 30(4), (1979) 296-303.
[62] L. Egghe, On the influence of growth on obsolescence, Scientometrics, 27(2), (1993) 195-214.
[63] L. Egghe, I.K. Ravichandra Rao, R. Rousseau, On the influence of production on utilization functions: Obsolescence or increased use? Scientometrics 34(2), (1995) 285-315.
[64] U. Gupta, Obsolescence of physics literature: Exponential decrease of the density of citations to Physical Review articles with age, Journal of American Society for Information Science 41(4), (1990) 282-287.
[65] A.L. Horvat, Handbook of Electrolyte Solutions: Physical Properties, Estimation and Correlation Methods. Ellis Horwood, Chichester (1985).
[66] D. Kashchiev, Nucleation: Basic Theory with Applications. Butterworth-Heinemann. Oxford, 2000.
[67] J.W. Mullin, Crystallization, 4th Edition. Butterworth-Heinemann, Oxford, 2001.
[68] A.G. Marangoni, On the use and misuse of the Avrami equation in the characterization of the kinetics of fat crystallization, Journal of American Oil Chemists' Society 75(10), (1998) 1465-1467.
[69] S. Bonzi, H.W. Snyder, Motivations for citation: A comparison of self citation and citation to others. Scientometrics 21 (1991) 245–254.
[70] M.H.C. Ho, J.S. Liu, The motivations for knowledge transfer across borders: the diffusion of data envelopment analysis (DEA) methodology, Scientometrics 94 (2013) 397–421 .
[71] D. Lyu, X. Ruan, J. Xie, Y. Cheng, The classification of citing motivations: a meta-synthesis, Scientometrics 126 (2021) 3243–3264.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 279–286
Received: 29 August 2021
Accepted: 30 September 2021

Comparison of classical machine learning algorithms in the task of handwritten digits classification

Porównanie klasycznych algorytmów uczenia maszynowego w zadaniu klasyfikacji liczb pisanych odręcznie

Oleksandr Voloshchenko*, Małgorzata Plechawska-Wójcik

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The purpose of this paper is to compare classical machine learning algorithms for handwritten number classification. The following algorithms were chosen for comparison: Logistic Regression, SVM, Decision Tree, Random Forest and k-NN. MNIST handwritten digit database is used in the task of training and testing the above algorithms. The dataset consists of 70,000 images of numbers from 0 to 9. The algorithms are compared considering such criteria as the learning speed, prediction construction speed, host machine load, and classification accuracy. Each algorithm went through the training and testing phases 100 times, with the desired metrics retained at each iteration. The results were averaged to reach the reliable outcomes.

Keywords: machine learning; classification; MNIST; classical algorithms

Streszczenie

Celem niniejszej pracy jest porównanie klasycznych algorytmów uczenia maszynowego do klasyfikacji liczb pisanych odręcznie. Do porównania wybrano następujące algorytmy: Logistic Regression, SVM, Decision Tree, Random Forest oraz k-NN. Do szkolenia i testowania powyższych algorytmów wykorzystano zbiór danych MNIST. Zbiór danych składa się z 70 000 obrazów cyfr od 0 do 9. Algorytmy porównywane są z uwzględnieniem takich kryteriów jak szybkość uczenia, szybkość budowania predykcji, obciążenie maszyny głównej oraz dokładność klasyfikacji. Każdy algorytm przeszedł przez fazy szkolenia i testowania 100 razy, z zachowaniem pożądanych metryk przy każdej iteracji. Wyniki zostały uśrednione w celu uzyskania wiarygodnych rezultatów.

Słowa kluczowe: uczenie maszynowe; klasyfikacja; MNIST; algorytmy klasyczne

*Corresponding author

Email address: oleksandr.voloshchenko@pollub.edu.pl (O. Voloshchenko)

1. Introduction

Machine learning is one of the most popular technologies these days, evolving incredibly fast and having a huge impact on the world today. The definition of machine learning dates back to 1959, when Arthur Samuel, an American pioneer in machine learning and artificial intelligence, said that machine learning is a field of learning that allows computers to learn without explicit programming [1]. In turn, in 1997, American professor Thomas Mitchell gave a more modern definition. He said that machine learning is when a program is told to learn from experience E with respect to some class of tasks T that measures performance P, if its performance on tasks T, which is measured as P, improves with each experience E.

This paper considers one of the main tasks of machine learning – the classification, and more specifically, the task of classifying handwritten digits.

Classification is one of the problems of machine learning, which refers to the supervised learning approach. Supervised learning is an approach in machine learning in which there is a set of data and expected results for that data, and the goal is to find a dependency rule between input and output parameters. In turn, image classification is a fairly commonly used task. For example, in classifying crops using satellite images, for agricultural purposes [2], determining benign and malignant tumors in medicine [3], gesture languages [4], in manufacturing, and in other modern tasks [5-8].

One of the problems of classification is the problem of classifying handwritten numbers [9-15]. The oldest papers and their conference discussions were held in 1994-1995. One of the examples of such works is *Comparison of classifier methods: a case study in handwritten digit recognition* [10], written by a large group of authoritative scientists in the field of machine learning. Despite the fact that the first mentions are quite old – the problem continues to be studied even now. For example, the paper [14] gives examples of using such recognition on documents, photos, or, for example, text analysis after touch screen input. The article compares three common machine learning algorithms (SVM, k-NN, NN) for text classification after optical character recognition (OCR). Or, for example, the paper [13] explores the capabilities of SVM, k-NN and neural network tools. The recognition time, error rate, number of misclassified images, and computation time were studied. In general, the following algorithms dominate in the literature among classical algorithms, those that do not rely on neural networks: Logistic Regression,

Support Vector Machine (SVM), k-Nearest Neighbours (k-NN) classifier, Decision Tree and Random Forest.

In this paper, the handwritten digit classification problem is solved using classical machine learning algorithms described above. These algorithms were chosen because they are among the most popular classification algorithms and are fairly common in the literature on such topics.

2. Purpose of the experiment

The purpose of this paper is to compare classical machine learning algorithms in the handwritten number classification problem. The experiment covers the comparison of several machine learning algorithms, such as logistic regression, support vector machine (SVM), k-nearest neighbor (k-NN) classifier, decision tree and random forest. The MNIST dataset was used in the analysis. The thesis defined for this paper is as follows: all the investigated algorithms will show different results in the handwritten digit classification problem.

2.1. Accuracy part

The primary purpose of this article is to compare the above algorithms in terms of overall accuracy and individual number recognition accuracy. This part of the work involves comparing algorithms in terms of the accuracy of handwritten number classification classes and the accuracy of classification in the case of each individual class.

The following research questions are defined for this part of the experiment:

- Which algorithm more accurately identifies classes of handwritten digits?
- Which numbers are better or worse defined for each algorithm?

2.2. Performance part

In addition to accuracy during the experiment, additional algorithm parameters, such as performance metrics, can also be explored. The following research questions are defined for the performance part of the experiment:

- Which algorithm learns faster?
- Which algorithm takes longer to learn?
- Which algorithm makes predictions faster?
- Which algorithm makes the prediction slower?
- Which algorithm requires more computer resources to learn?

3. Experiment description

To compare the algorithms, it is necessary to provide a plan for testing all of the mentioned algorithms under the study. All selected algorithms will be compared in the handwritten number classification task. MNIST handwritten number dataset was chosen as a ready-made dataset. This set consists of 70,000 pictures of numbers from 0 to 9, 28 by 28 pixels (a total of 784 pixels per picture). The parameters by which each of the algorithms will be evaluated are defined. These parameters were chosen so that the values obtained would help to answer the questions presented in section 2. In the case of the accuracy part (section 2.1), the following metrics were determined:

- accuracy – accuracy of algorithms when classifying handwritten numbers, which is determined as the ratio of successful classifications to the total number of tests;
- accuracy within numbers classes – accuracy of algorithms in determining each individual class, where classes are numbers from 0 to 9. Metric shows the percentage of correctly recognized numbers of a particular class (from 0 to 9) out of the total amount of numbers of that class in the test data set.

In the case of the performance part of the article (section 2.2), the following metrics were defined for the study:

- learning time – amount of time it takes for the algorithm to learn on the training data set;
- prediction time – amount of time needed for the algorithm to build the prediction for the whole test dataset;
- CPU load – percentage of CPU load during the processing of the training data set;
- RAM load – amount of used MiB RAM while processing the training dataset;

All of the above parameters must be measured during the execution of the overall plan of the experiment. This plan is as follows:

1. Randomly divide 70,000 handwritten number pictures into training and test datasets.
2. Start measuring the training parameters.
3. Train the learning algorithm on the training dataset.
4. End of training parameters measurement.
5. Beginning of measurement of the test parameters.
6. Testing of the trained algorithm on the test dataset.
7. End of the testing parameters measurement.
8. Saving of all measured parameters.
9. Repeat steps 1-8 N-number of times.
10. Results analysis.

Following this plan, steps 2 and 4 obtain the values of the parameters related to the training of the model, namely learning time, CPU load and RAM load. In steps 5 and 7 the data of the parameters calculated during model testing are obtained, namely prediction time, accuracy and accuracy within numbers classes. In step 8, the parameters obtained during iteration are saved for further analysis. To improve the accuracy of the obtained results, each algorithm went through the steps of the above plan 100 times, and the final results were averaged.

At each iteration, the set of pictures described above was randomly divided into a training and a test dataset according to the recommendations of its creators. The training dataset at each iteration contained 60,000 pictures (85.7% of the total dataset), while the test dataset contained 10,000 pictures (14.3% of the total dataset).

For the experiment was used the python language and the library scikit-learn, which contains ready-made implementations of the studied algorithms. The numerical values of the pixels of the pictures were taken as the parameters of the models. This means that each input

algorithm had 784 numeric parameters on the basis of which it must classify the number in the picture.

The computer used for the experiment is an Ubuntu 18.04 system, a 2.8GHz Intel Core i7-7700HQ processor and 16GB of DDR4 RAM. During the experiment the algorithms were run with a CPU limit of 1 thread.

4. Algorithms

4.1. Logistic Regression

Logistic regression is a statistical model that uses a logistic function to model the relationship between an output variable and input parameters. Binary logistic regression assumes answers at the model output as 0 or 1. Logistic regression can be used to estimate the probability of occurrence of an event for a particular test.

The logistic regression problem uses a linear regression function for estimating prediction. However, the linear regression equation does not fit the definition of logistic regression, since the model's responses may differ from 0 and 1. Some kind of transformation, also called a logit-transformation, must be performed to solve the problem. This transformation looks like this (1):

$$P = \frac{1}{1 + e^{-y}} \tag{1}$$

where P is the probability of occurrence of the event of interest, e is the Euler number, and y is the standard regression equation.

During the experiment, the *LogisticRegression* class from the *linear_model* module of the used library was used as a logistic regression in the experiment. Since the algorithm needs to classify many classes, in this case the algorithm with a multinomial parameter was used. There were also used parameters for L2 regularization and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimizing algorithm, which fit multiclass classification, are quite robust, and together should converge faster. When it comes to the number of iterations for convergence for the optimizing algorithm, a value of 100 iterations was chosen.

4.2. Decision Tree

Decision trees are a method for data analysis and predictive analytics. It is clear from the name that the tool is a hierarchical tree structure in which nodes are rules and leaves are decision points.

The rules are generated automatically by training the algorithm on a test dataset. These nodes test whether an example matches a given rule on a particular attribute. As a result of the check, the set of examples in a node is split into two subsets, one of which contains examples that satisfy the rule and one that does not. The sets then descend the tree to the next nodes, where further conditions are checked. This process is repeated until some stopping condition is satisfied. The last node performs no more checks, such a node is called a tree leaf. The leaf defines a solution for each example it contains. However, unlike a node, a leaf does not contain a rule, but a subset of objects that satisfy all branch rules.

The attribute that is used for partitioning in a node must partition the set so that the resulting subsets contain objects with the same class labels, or are as close to this result as possible. In the experiment, the Gini index, which is calculated by formula (2), is used to solve this problem:

$$Gini(Q) = 1 - \sum_{i=1}^{n} p_i^2 \tag{2}$$

where Q – the final data set, n – the number of classes in that set, p_i – probability of the i-th class. This index takes values from 0 to 1. If the index takes the value 0, then all elements of the dataset Q belong to one class, and 1 if all classes are equally likely. Obviously, the smaller the index value, the better the partition will be.

After the rules are formed on the branches of the tree on the basis of the training dataset – the branch pruning stage is performed. This approach determines the accuracy and error of the full tree. After evaluation, leaves and nodes are removed from the tree, the removal of which will not lead to a significant loss of accuracy and increase the error.

During the experiment, the *DecisionTreeClassifier* class from the *tree* module was chosen as the Decision Tree algorithm. The Gini index was used as a criterion for evaluating data partitioning in the tree, and for the partitioning itself all the parameters of the input objects were taken into account. The classification tree was split until the outermost nodes (leaves) had objects of the same class, or until the number of objects in the leaf was equal to 1. The number of leaves was not limited.

4.3. Random Forest

This model is an improvement of the previous one – the decision tree. The improvement is made by the fact that the random forest consists of an ensemble of decision trees [16].

During training, each tree is trained on random data sets, each of which is taken from the main training set. These sets are generated using a statistical method called bootstrapping. Although each tree may be highly variable with respect to a particular training dataset, training trees on different sample sets reduces the overall variability of the forest without sacrificing accuracy.

The random forest result is derived by averaging the predictions obtained from each tree in the ensemble of decision trees. This bootstrapping approach for each tree, and after averaging, is called bagging.

During the experiment, the *RandomForestClassifier* class from the *ensemble* module of the library was used as Random Forest algorithm. The used forest had 100 tree units, which were built using bootstrap samples. The trees used the same parameters as the Decision Tree algorithm above. This means that each tree used a Gini index to evaluate the quality of the separation. All input object parameters were used in the node for splitting. The tree was not limited in growth and grew as long as the leaves did not consist of objects of the same class, or as long as the leaf did not count 1 element.

4.4. Support Vector Machine

Support vector machine is a supervised machine learning algorithm. It is used for both classification and regression tasks. In this algorithm, each sample of the training dataset is represented as a point in n-dimensional space, where n is the number of features of an element in the dataset. The task of the algorithm is to construct a hyperplane that would optimally divide by classes the elements in this space. This hyperplane is constructed so that the distance from it to the elements of the classes was the largest. In other words, the principle of the algorithm is to increase the gap between the dividing hyperplane and the vectors of classes that are closest to it. These vectors are called support vectors. Obviously, the best hyperplane is the one with the largest gap.

During the experiment, the *SVC* class from the *svm* module of the used library was used as Support Vector Machine. Multi-class classification in the algorithm is supported by the one-vs-one scheme. As a kernel, the Radial Basis Function was used with an adjusting parameter with value 1 and with value gamma, which was calculated using the (3) formula:

$$\gamma = \frac{1}{n\_features \times \sigma^2} \tag{3}$$

where $n\_features$ is the number of model parameters, which in the case of this experiment is the number 784, and σ^2 – dispersion of test parameters values. The average value of the γ parameter during the experiment was 2.06.

4.5. k – Nearest Neighbors

The algorithm works according to the following principle: the object under study gets the class that most of its neighbors have in space. During training, the algorithm remembers the feature vectors of incoming observations and their corresponding class labels. At the classification stage, k nearest neighbors that already have a class are determined for the example under study.

In the simplest case, a class is defined by simply choosing the most frequent class among k examples, but this approach is not always successful. For example, in cases where the frequency of occurrence of all classes is the same. It is also important that not all neighbors are equally significant for the studied example. The usual case is simple unweighted voting. In this strategy, all k neighbors have the same weight regardless of their distance to the examined instance.

However, it would be reasonable to assume that the further away the sample is, the less influence it has. To do this, the model includes a weighting of the parameters depending on their distance to the sample under study. This method is called weighted voting.

Also, an important aspect of the algorithm is the choice of coefficient k. If k is small enough, there is an overfitting phenomenon, and the decision is made on the basis of a rather small number of examples and is generally of low significance. When k is too large, too many examples from different classes are involved in the classification process, which poorly reflects the local features of the data.

During the experiment, the *KNeighborsClassifier* class from the *neighbors* module of the used library was used as the k-NN algorithm. To determine the class of an object, five of its nearest neighbors were studied. Each object in the space had the same weight.

5. Results

During the experiment described in section 3, each algorithm under study went through the training and testing stages 100 times. At each stage the parameters described in section 3 were recorded. All results were averaged and displayed in this section.

5.1. Accuracy results

When evaluating the overall accuracy of the algorithms on the test data set, the results were obtained as shown in Figure 1.

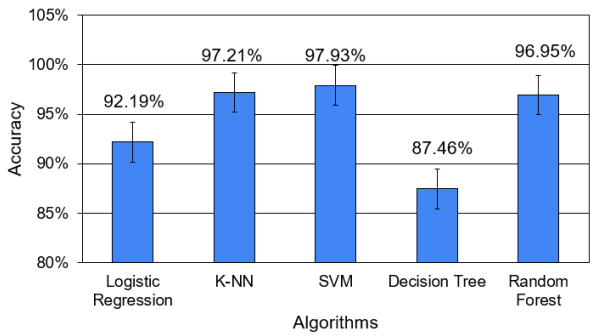


Figure 1: Accuracy of the algorithms

As can be seen from the graph, the best results were shown by the SVM, k-NN and Random Forest algorithms, the Logistic Regression algorithm comes next and the Decision Tree algorithm showed the worst performance relative to other algorithms in this task.

The next accuracy metric is accuracy within numbers classes. Along with the usual bar charts, confusion matrices are also used to display the results, which will show what errors the algorithm made during training. For each algorithm, the matrices were obtained on one of the 100 iterations of the overall experiment plan.

Figure 2 presents the results for the Logistic Regression algorithm.

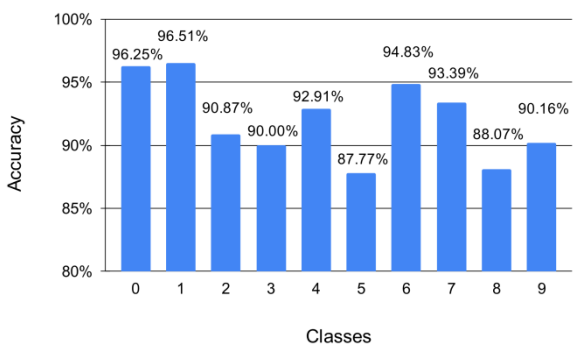


Figure 2: Accuracy of determining each individual number by Logistic Regression

The algorithm is best at determining the numbers 0 and 1, in turn determining the numbers 5 and 8 is relatively worse than other numbers. Figure 3 shows the confusion matrix of the Logistic Regression algorithm. From the matrix, it can be seen that the number 5 was mostly mistaken for the numbers 3, 6, and 8. In the case of number 8, false predictions were made in favor of numbers 1, 3, and 5.

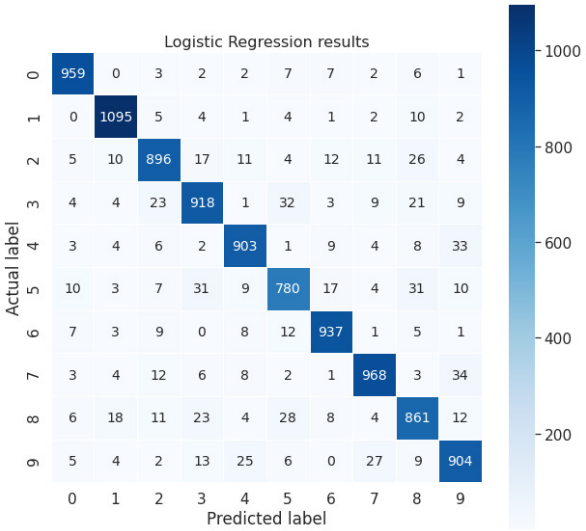


Figure 3: Logistic Regression confusion matrix

In the case of the k-NN algorithm the following results were obtained, shown in Figure 4.

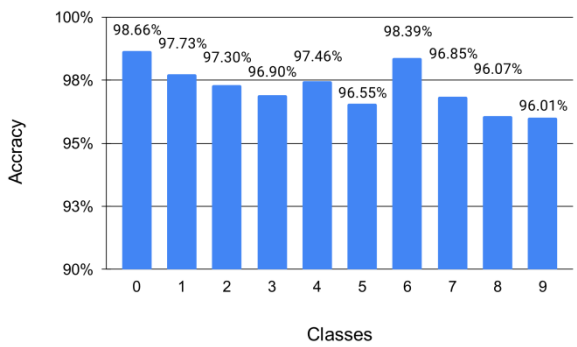


Figure 4: Accuracy of determining each individual number by k-NN

The algorithm performed well in the task of classifying numbers in general, but showed great accuracy for the numbers 0 and 6, but did a little worse for the numbers 8 and 9. The confusion matrix for the k-NN algorithm is shown in Figure 5. The matrix shows that in the case of number 8, the largest number of false predictions are assigned to numbers 1, 3, and 5. In the case of number 9 most of the errors fell on number 7 and a few on number 4.

The results of the SVM algorithm are shown in Figure 6. The algorithm did best with the numbers 0, 1 and 6, the algorithm did relatively worst with the classification of the number 9. The confusion matrix of the algorithm results is shown in Figure 7. The matrix shows that in the case of number 9, most of the false assumptions were made in favor of numbers 4 and 7.

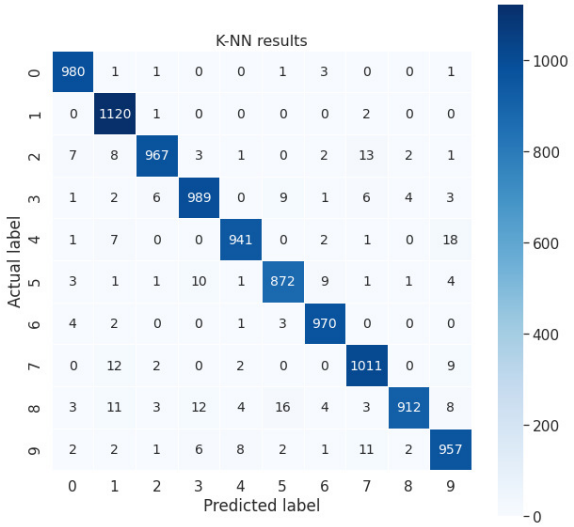


Figure 5: k-NN confusion matrix

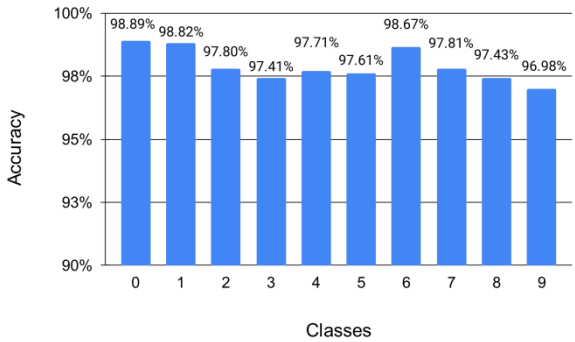


Figure 6: Accuracy of determining each individual number by SVM

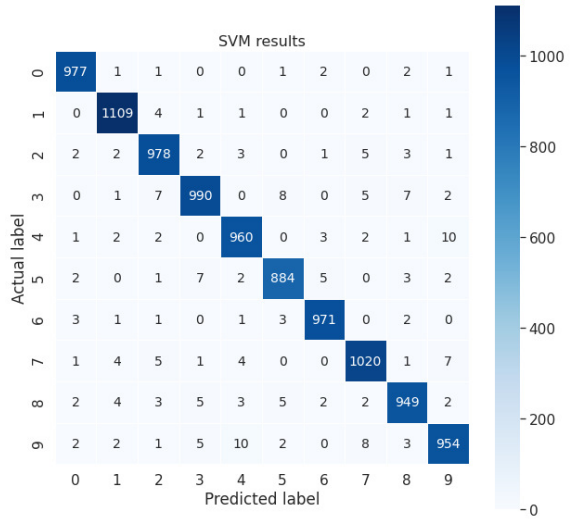


Figure 7: SVM confusion matrix

The Decision Tree algorithm got the results shown in Figure 8. The algorithm showed the best classification results for numbers 0 and 1. In the case of numbers 3, 5, 8, and 9, the algorithm showed the worst result, which, in general, is the lowest result among all the algorithms studied. The confusion matrix for the Decision Tree algorithm is shown in Figure 9. From the data in the matrix, it can be said that in the case of number 3,

a lot of false predictions were made in favor of classes 2, 5, 8, and 9. In the case of number 5, false predictions were made in favor of numbers 3, 6, 8, and 9. The number 8 was taken by the algorithm as the numbers 2, 3, 4, 5, and 9 in most false cases. In the case of number 9, most false predictions were made in favor of numbers 3, 4, 5, 7, and 8.

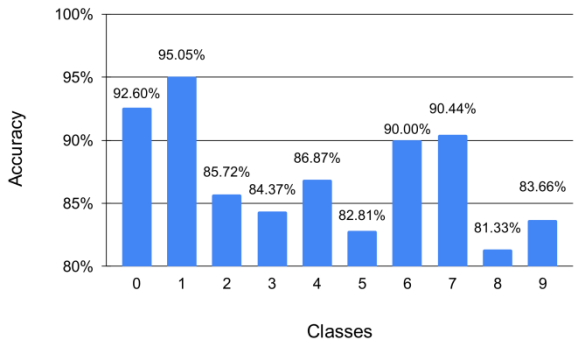


Figure 8: Accuracy of determining each individual number by Decision Tree

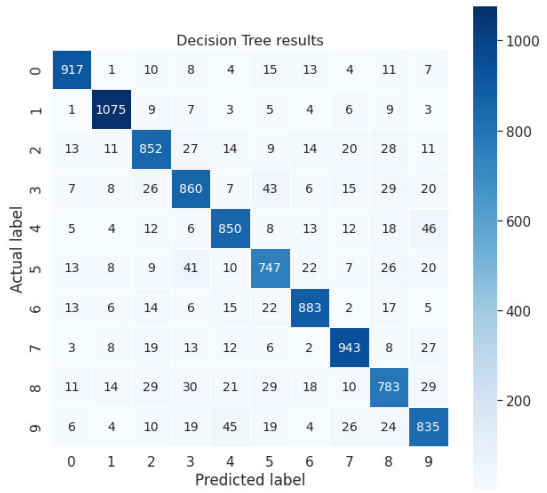


Figure 9: Decision Tree confusion matrix

In the case of the Random Forest algorithm the results are shown in Figure 10.

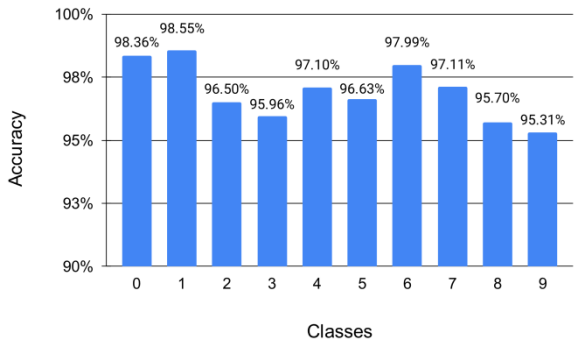


Figure 10: Accuracy of determining each individual number by Random Forest

The algorithm showed the best results in numbers 0, 1 and 6, but in the case of classes 3, 8 and 9 the results are relatively worse. The confusion matrix in the case of this algorithm is shown in Figure 11. From the matrix it can be seen that the algorithm confused the number 3 with the numbers 2 and 8. In the case of number 8, false predictions were in favor of numbers 3, 5 and 9. In the case of number 9, numbers 3, 4, and a little bit of 7 caused problems for the algorithm.

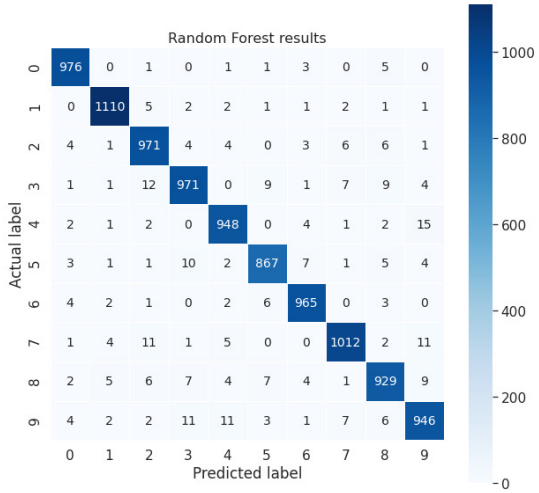


Figure 11: Random Forest confusion matrix

For a more convenient comparison of the accuracy of the class classification is built Figure 12, which can be used to compare the accuracy of determining the individual classes of numbers by algorithms.

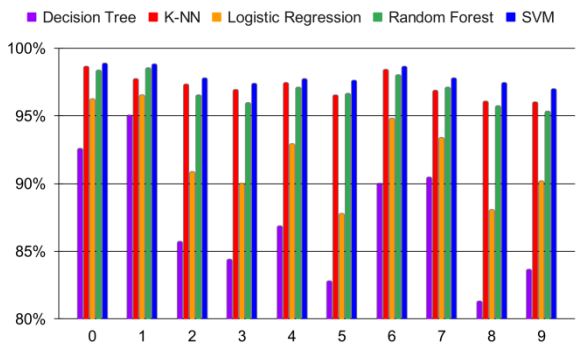


Figure 12: Accuracy of classes classification by algorithms

5.2. Performance results

In terms of learning time, the algorithms showed the following results shown in Figure 13.

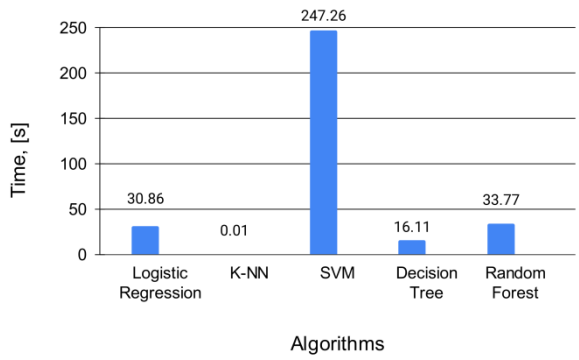


Figure 13: Learning time of algorithms

As can be seen, the fastest algorithm in terms of learning was k-NN. Such a fast result is due to the specifics of the algorithm during training, the main calculations and time costs occur during the construction of predictions, which will be shown further in this section of the article. The next fastest algorithms are Decision Tree, Logistic Regression and Random Forest. The slowest algorithm in the training phase was SVM.

The next evaluation criterion is the speed of building the predictions for the test dataset with the results shown in Figure 14. As can be seen, the fastest algorithms during the prediction stage for the test dataset were Decision Tree, Logistic Regression and Random Forest. The slowest algorithm at this stage was the SVM algorithm.

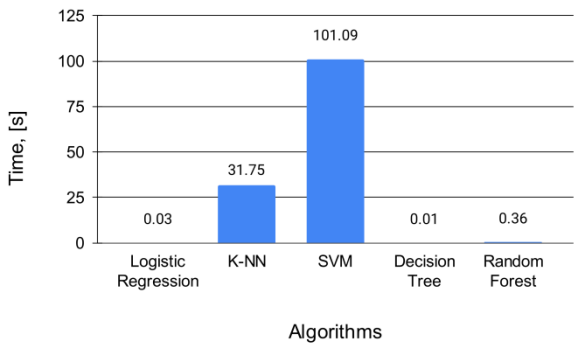


Figure 14: Algorithms prediction time

The next metric to measure was CPU load. The results are shown in Figure 15. SVM, Decision Tree, and Random Forest had the highest CPU load. The k-NN algorithm showed the lowest load.

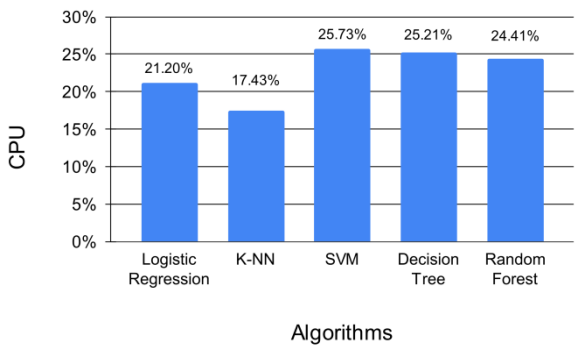


Figure 15: CPU load during learning

On the RAM usage side, the following results were obtained, shown in Figure 16. The highest level of memory usage is observed for the SVM and Logistic Regression algorithms. The Random Forest and Decision Tree algorithms are relatively average and the lowest memory usage is observed for the k-NN algorithm.

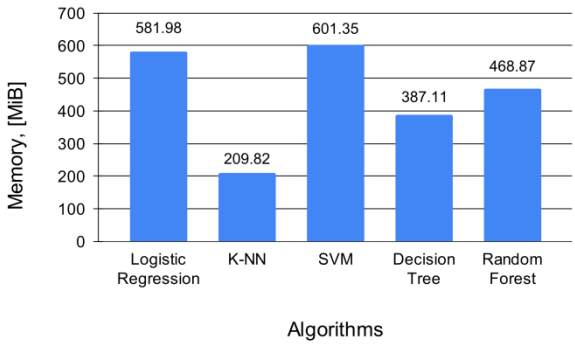


Figure 16: Memory usage during learning

6. Conclusions

As already described in section 2, the aim of the experiment was to compare classical algorithms in the problem of classifying handwritten numbers. First of all, as described in section 2.2 of this paper, we analyzed the accuracy metrics of the algorithms.

The accuracy analysis showed that the most accurate algorithms in this problem were SVM, k-NN and Random Forest algorithms with 97.93%, 97.21% and 96.95% accuracy. The results for this metric can also be seen in more detail in Figure 1.

An accuracy metric within number classes, the overall results of which are shown in Figure 12, showed that the algorithms classified numbers 0, 1, and 6 well, but had some problems classifying numbers 3, 5, 8, and 9. Based on the confusion matrices for each of the algorithms (figures 3, 5, 7, 9, and 11), we can find that in most of the incorrect cases the numbers 3, 5, 8, and 9 were confused with each other. These results are generally plausible, because the spelling of these numbers is quite similar.

The second part of the research questions (section 2.2) concerned the study of the performance metrics, which were also measured according to the experimental plan in section 3. The conclusions of the results of these metrics are described below.

The k-NN algorithm was the fastest on the training dataset with a result of 0.01 second. This speed is caused by the specifics of the algorithm, the main calculations of which take place at the prediction stage. The next fastest was the Decision Tree algorithm with a result of 16.11 seconds.

The slowest algorithm was SVM, which took an average of 247.26 seconds to train.

In terms of prediction construction time, the Decision Tree, Logistic Regression, and Random Forest algorithms were the fastest to process the test data set with corresponding results of 0.01, 0.03 and 0.36 seconds. The slowest in this aspect was the SVM algorithm with a result of 101.09 seconds.

The CPU load and memory consumption metrics show that the k-NN algorithm was the least stressful on the machine during training with a CPU load of 17.43% and used memory of 209.8 MiB. The SVM algorithm, in turn, had the highest load on the host machine among all

the algorithms studied with a CPU load of 25.73% and used memory of 601.35 MiB.

The experiment confirmed the thesis described in section 2: all the algorithms under study showed different results in the task of classifying handwritten numbers. In the future, this experiment could be conducted using deep learning tools and compare different types of neural networks in this task.

References

[1] A.L. Samuel, Some studies in machine learning using the game of checkers, IBM Journal of Research and Development 44 (2000) 206-226.

[2] J.M. Peña-Barragán, P.A. Gutiérrez, C. Martínez, J. Six, R.E. Plant, F. López-Granados, Object-Based Image Classification of Summer Crops with Machine Learning Methods, Remote Sensing 6 (2014) 5019-5041.

[3] P. Mohapatra, B. Panda, S. Swain, Enhancing histopathological breast cancer image classification using deep learning, The International Journal of Innovative Technology and Exploring Engineering 8 (2019) 2024-2032.

[4] N.H. Aung, Y.K. Thu, S.S. Maung, Feature Based Myanmar Fingerspelling Image Classification Using SIFT, SURF and BRIEF, Proceedings of the 17th International Conference on Computer Applications (ICCA 2019) (2019) 245-253.

[5] I.H. Sarker, A.S. Kayes, P. Watters, Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage, Journal of Big Data 6 (2019) 1-28.

[6] R. Razavi, A. Gharipour, M. Gharipour, Depression screening using mobile phone usage metadata: a machine learning approach, Journal of the American Medical Informatics Association 27 (2020) 522-530.

[7] M. Pennacchiotti, A.-M. Popescu, A Machine Learning Approach to Twitter User Classification, Proceedings of the International AAAI Conference on Web and Social Media 5 (2021) 281-288.

[8] Y. Nieto, V. Gacía-Díaz, C. Montenegro, C.C. González, R.G. Crespo, Usage of machine learning for strategic decision making at higher educational institutions, IEEE Access 7 (2019) 75007-75017.

[9] L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, L.D. Jackel, Y. LeCun, U.A. Muller, E. Sackinger, P. Simard, V. Vapnik, Comparison of classifier methods: a case study in handwritten digit recognition, Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5) 2 (1994) 77-82.

[10] Y. LeCun, L.D. Jackel, L. Bottou, A. Brunot, C.Cortes, J. Denker, H. Drucker, I. Guyon, U.A. Muller, E. Sackinger, P. Simard, Comparison of learning algorithms for handwritten digit recognition, International conference on artificial neural networks 60 (1995) 53-60.

[11] B. El Kessab, C. Daoui, B. Bouikhalene, R. Salouan, A Comparative Study between the Support Vectors Machines and the K-Nearest Neighbors in the Handwritten Latin Numerals Recognition, International Journal of Signal Processing, Image Processing and Pattern Recognition 8 (2015) 325-336.

[12] K. Zhao, Handwritten digit recognition and classification using machine learning, M.Sc. in Computing (Data Analytics), Technological University Dublin (2018).

[13] C. Kaensar, A comparative study on handwriting digit recognition classifier using neural network, support vector machine and k-nearest neighbor, The 9th International Conference on Computing and InformationTechnology (IC2IT2013) (2013) 155-163.

[14] N.A. Hamid, N.N. Sjarif, Handwritten recognition using SVM, KNN and neural network, arXiv preprint arXiv:1702.00723 (2017).

[15] T.A. Assegie, P.S. Nair, Handwritten digits recognition with decision tree classification: a machine learning approach, International Journal of Electrical and Computer Engineering (IJECE) 9 (2019) 4446-4451.

[16] L. Breiman, Random forests, UC Berkeley TR567 (1999).

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 287–294
Received: 01 September 2021
Accepted: 04 October 2021

The comparative analysis of Java frameworks: Spring Boot, Micronaut and Quarkus

Analiza porównawcza szkieletów programistycznych języka Java: Spring Boot, Micronaut oraz Quarkus

Maciej Jeleń*, Mariusz Dzieńkowski

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The aim of the work is a comparative analysis of three frameworks designed for building web applications for the Java programming language: Spring Boot 2.4.4, Micronaut 2.5.4 and Quarkus 1.13.4.Final. Test applications were prepared, equipped with the same functionality as used in the experiment consisting in measuring the server response times to a request of POST, GET, PUT and DELETE performing operations on the database. For each test application, the scenario aimed at measuring the time of handling requests under various load conditions was repeated five times. During each repetition of the scenario, the load which was the average number of requests sent per second by virtual users was increased. In parallel with performance tests, the reliability of the test applications was measured. Reliability was defined as the percentage of requests sent to the server that ended in a failure. The comparative analysis also took into consideration the volume of the code of the test applications based on the selected frameworks. The performed analyses showed that in terms of most of the criteria considered in this work Micronaut proved to be the best framework.

Keywords: web application; frameworks of the Java programming language; performance analysis; Spring Boot; Micronaut; Quarkus

Streszczenie

Przedmiotem tej pracy jest analiza porównawcza trzech szkieletów programistycznych do budowy aplikacji internetowych dla języka Java: Spring Boot 2.4.4, Micronaut 2.5.4 oraz Quarkus 1.13.4.Final. Przygotowano aplikacje testowe, wyposażone w tą samą funkcjonalność, które wykorzystano w eksperymencie, polegającym na pomiarze czasów odpowiedzi serwera na żądania typu POST, GET, PUT i DELETE – realizujące operacje na bazie danych. Dla każdej aplikacji testowej, powtórzono pięciokrotnie scenariusz, który miał na celu zmierzyć czas obsługi żądań w różnych warunkach obciążeniowych. Podczas każdego powtórzenia zwiększano wielkość obciążenia, które oznaczało średnią liczbę wysyłanych żądań na sekundę przez wirtualnych użytkowników. Równolegle z badaniami wydajności wykonano pomiary niezawodności aplikacji testowych. Niezawodność zdefiniowano jako odsetek żądań wysyłanych do serwera, które zakończyły się niepowodzeniem. W porównaniach wzięto również pod uwagę objętość kodu aplikacji testowych opartych na wybranych szkieletach. Z przeprowadzonych analiz wynikło, że pod względem większości rozpatrywanych w ramach tej pracy kryteriów najlepszym szkieletem programistycznym okazał się Micronaut.

Słowa kluczowe: aplikacja internetowa; szkielety programistyczne języka Java; analiza wydajności; Spring Boot; Micronaut; Quarkus

*Corresponding author

Email address: maciej.jelen@pollub.edu.pl (M. Jeleń)

1. Wstęp

W czasach, gdy internet nie był jeszcze tak rozpowszechniony, jak to jest obecnie, większość aplikacji komputerowych była dostępna w formie desktopowej, co oznaczało, że każdy klient musiał zainstalować aplikację oddzielnie na swoim komputerze. Niosło to ze sobą pewne konsekwencje, takie jak na przykład konieczność uaktualniania danego oprogramowania u każdego klienta, co zmniejszało produktywność oraz zabierało więcej zasobów. W 1995 roku pojawiła się pierwsza wersja języka Java, umożliwiająca tworzenie apletów, które wkrótce mogły być uruchamiane w przeglądarkach internetowych.

W celu przyspieszenia tworzenia aplikacji internetowych w Javie, powstały szkielety programistyczne dla tego języka – zwane często z języka angielskiego frameworkami. Ich stosowanie ułatwia tworzenie aplikacji internetowych oraz redukuje ilość powtarzalnego kodu, jaki programista musiał dodać przed faktycznym utworzeniem głównej części oprogramowania. Wynikało to z tego, że przystępując do implementacji logiki biznesowej, należało zaimplementować takie elementy jak konfiguracje, połączenia z bazami danych czy innymi serwisami internetowymi za pośrednictwem interfejsu API (ang. Application Programming Interface – Interfejs Programowania Aplikacji).

Jednym z najbardziej popularnych obecnie szkieletów programistycznych opartych na języku Java jest Spring Framework, który jest następcą technologii Java EE. Na bazie tych szkieletów powstawały i ciągle powstają nowe rozwiązania, wykorzystujące zalety swoich poprzedników oraz naprawiając ich wady. Przykładem

nowych frameworków są rozpatrywane w ramach tej pracy Micronaut oraz Quarkus. Pierwszy szkielet został utworzony przez grupę programistów, którzy wcześniej przyczynili się do powstania frameworka Grails, natomiast drugi jest efektem prac inżynierów firmy Red Hat.

Obecnie istnieje bardzo duża liczba języków programowania, które mają swoje frameworki - najlepszym przykładem jest język Java. Szeroki wybór oraz różnorodność powodują, że trudno jest się zdecydować na najlepsze, najbardziej optymalne rozwiązanie dostosowane do konkretnego przedsięwzięcia programistycznego. W związku z tym powstaje wiele publikacji, w których podejmowany jest ten problem, i które mają ułatwić i przyśpieszyć proces decyzyjny dotyczący wyboru właściwej technologii. W ramach tej pracy, przeanalizowano trzy szkielety programistyczne dla języka Java: Spring Boot, Micronaut oraz Quarkus, które mają uniwersalne zastosowania aplikacyjne.

2. Przegląd literatury

W przypadku aplikacji internetowych, mających architekturę wielowarstwową, które są uniwersalne i skierowane do szerokiej grupy odbiorców, bardzo ważna jest wydajność i jakość wykonania, a także łatwość ich implementacji. Przedmiotem pracy [1] było porównanie dwóch szkieletów programistycznych: Spring Framework, którego pochodną jest Spring Boot oraz Play Framework. W tym celu przygotowano dwie aplikacje internetowe oparte na badanych szkieletach oraz przeanalizowano wydajność za pomocą narzędzia JMeter. Miarą użytą do porównań był czas wyrażony w milisekundach, w jakiej dany serwer odpowiadał na żądania 100 użytkowników wysyłanych w ciągu jednej sekundy. Wysyłane żądania polegały głównie na przypisaniu oceny studentowi i dodatkowo obejmowały operacje zalogowania się do i wylogowania się z systemu. Na podstawie uzyskanych wyników, wysunięto wnioski, że przy małym obciążeniu szkielet Play Framework był wydajniejszy od Spring Framework. Natomiast w warunkach dużego obciążenia, podczas wykonywania operacji, lepsze wyniki uzyskiwał Spring Framework.

Podobną tematykę podjęli autorzy pracy [2], w której zawarli wyniki analizy trzech szkieletów programistycznych (Enterprise Java Beans 2, Spring Framework, Grails) w środowisku maszyny wirtualnej Java. Jako aplikacji testowej, użyto systemu do automatyzacji procesów wstępnego testowania. Również w tym przypadku porównano czasy odpowiedzi na żądania każdej aplikacji opartej na danym szkielecie. Analizy przeprowadzano przy różnych obciążeniach – podając przez okres 1, 5, 10 i 60 sekund zestaw danych od 20 do 700 użytkowników na sekundę. Do badań wykorzystano bibliotekę JMeter. Podczas testów wysyłano stałą liczbę pakietów dla każdego przypadku. Porównanie tych trzech technologii pod kątem wydajności doprowadziło autorów do wniosku, że szkielet Grails wykorzystujący język Groovy jest platformą wydajniejszą dla realizowanego systemu zarówno dla małych jak i dużych obciążeń.

Quarkus na platformie GraalVM, która jest alternatywą dla wirtualnej maszyny Java, oprócz szkieletów Micronaut oraz Spring Boot na platformie OpenJDK 13, był przedmiotem kolejnej analizy wykonanej w ramach pracy [3]. Autorzy przeprowadzili test, który polegał na pomiarze czasu odpowiedzi serwera przy próbie odczytu danych - w postaci kolekcji obiektów metodą GET pochodzącą z protokołu HTTP. Uzyskane wyniki wydajności wykazały, że najlepiej w testach wypadł szkielet Quarkus wykorzystujący platformę GraalVM, zaś w przypadku platformy OpenJDK 13 lepsze rezultaty uzyskał Micronaut. Szczegółowe wyniki mówią o tym, że Quakus uruchamiany na natywnym obrazie GraalVM jest o prawie 82% szybszy niż ta sama aplikacja napisana w Spring Boot i wykorzystująca OpenJDK 13. Natomiast gdy obie aplikacje działają na platformie OpenJDK 13, wówczas Quarkus jest o 18% szybszy od Spring Boot. Porównując szkielety Quarkus i Micronaut działające na OpenJDK 13, różnica w szybkości między nimi wynosi 17% na korzyść tego drugiego.

W artykule [4] porównywano wydajność dwóch aplikacji opartych na szkieletach Spring Boot oraz Microsoft .NET, wykorzystujących dwa podobne do siebie języki programowania, którymi są Java oraz C#. Wydajność w tych badaniach była określana jako czas odpowiedzi serwera na zadane żądania dla aplikacji wykonanej w Spring Boot oraz Microsoft .NET Core. Dodatkowo w pracy, mierzona była także niezawodność aplikacji, wyrażana w procentach i definiowana jako liczba żądań wysyłanych do serwera, które nie zostały poprawnie obsłużone. Testy były wykonywane pod obciążeniem, które zmieniało się co 30 sekund. Najpierw żądania do serwera wysyłało 1000 użytkowników, następnie 2000, 4000, 8000, 16000, 32000 i w końcu 64000 użytkowników, w każdej sekundzie. W każdym z tych testów, dla żądań reprezentowanych przez metody protokołu HTTP: GET, POST, PUT oraz DELETE, przy małej liczbie użytkowników szkielety miały podobną wydajność. Natomiast przy większych obciążeniach, wyraźnie lepsze wyniki osiągała aplikacja wykorzystująca język C# i platformę programistyczną .NET Core. Biorąc pod uwagę wyniki niezawodności aplikacji opartych na porównywanych szkieletach, można stwierdzić że kształtowały się one na podobnych poziomach. Należy przy tym zaznaczyć, że procent zakończonych niepowodzeniem żądań rósł wraz ze wzrostem obciążenia serwera.

W kolejnym artykule [5] badano wydajność takich samych aplikacji opartych na jednym z czterech szkieletów programistycznych języka Java: Spring Boot, Micronaut, Quarkus oraz Javalin. Dokonano porównań, w których wzięto pod uwagę wydajność, traktowaną jako czas uruchomienia aplikacji w środowisku produkcyjnym i programistycznym, a także zużycie pamięci oraz obciążenie procesora. Wykonano także test wydajności żądań pobrania (metodą HTTP GET) oraz zapisu (metodą HTTP POST) danych. Do serwera wysyłano najpierw 8000 żądań pobierających 1000 rekordów z bazy danych oraz 8000 żądań typu POST, które dodawały 200 rekordów do bazy. Na podstawie uzyska-

nych wyników okazało się, że najwydajniejszym szkieletem jest Javalin, który wyróżniał się najniższym spośród innych zużyciem pamięci, najmniejszym obciążeniem procesora oraz najkrótszym czasem odpowiedzi na żądania.

W każdym z przytoczonych artykułów realizowano badania, które koncentrowały się głównie na analizie wydajności pod obciążeniem aplikacji internetowych opartych na jednym z framworków języka Java. W ramach tej pracy oprócz kwestii wydajnościowych, podjęto problem niezawodności systemu rozumianego jako liczba żądań, na które serwer poprawnie nie odpowiedział. Aspekt ten może mieć kluczowe znaczenie przy podejmowaniu decyzji wyboru technologii dla tworzonego oprogramowania.

3. Wykorzystane technologie i narzędzia

3.1. Java 11

Język Java jest obecnie jednym z najpopularniejszych języków programowania. W rankingu TIOBE [6] z lipca 2021 roku, zajmuje on wysokie trzecie miejsce, zaraz za językiem C oraz językiem Python. Java zaczerpnęła podstawowe koncepcje z takich języków jak Smalltalk i C++. Na bazie Smalltalk powstała maszyna wirtualna Javy i narzędzie do zarządzania pamięcią (ang. garbage collector). Z kolei z języka C++ Java przejęła dużą część składni. Głównymi koncepcjami tego języka jest obiektowość, dziedziczenie oraz niezależność architektury, co pozwala na uruchomienie kodu napisanego w tym języku na wszystkich dostępnych systemach operacyjnych takich jak Microsoft Windows, różnych dystrybucjach Linuxa np. Ubuntu, Mint, Fedora oraz na systemach firmy Apple. Wersja 11 języka Java jest ostatnią dostępną, stabilną i długoterminową wersją oznaczaną skrótem LTS (ang. Long Term Support) [7].

3.2. REST

REST (ang. Representional State Transfer) jest stylem architektury do budowy aplikacji internetowych, w którym nacisk jest kładziony na używanie dostępu do zasobów za pomocą metod protokołu HTTP z użyciem odpowiedniego URL (ang. Uniform Resource Locator). Architektura ta nakłada pewne restrykcje, takie jak nieużywanie podkreśleń („_”), używanie wyłącznie małych liter oraz słów w liczbie mnogiej [8]. W tym przypadku ciało żądania do serwera wysyła się najczęściej w formacie JSON. Wcześniej, do budowy aplikacji internetowych opartych na usługach sieciowych, wykorzystywany był protokół SOAP (ang. Simple Object Access Protocol). Obecnie SOAP jest wypierany przez REST, przede wszystkim ze względu na prostotę tego drugiego rozwiązania. Do głównych cech architektury REST należą [9]:

- jednolity interfejs komunikacyjny,
- podział na aplikacje klient-serwer,
- bezstanowość,
- przechowywanie danych w pamięci podręcznej.

3.3. Narzędzie Gatling

Gatling jest darmowym narzędziem do wykonywania testów wydajnościowych, udostępnionym na licencji Apache License 2.0 w 2011 roku [10]. Narzędzie to oparte jest głównie na języku Scala z użyciem składni DSL (ang. Domain Specific Language), nieblokującym serwerze Netty oraz szkielecie programistycznym Akka. Gatling został zaprojektowany z myślą o łatwości użycia, wysokiej wydajności i łatwości utrzymania [11]. Z jednej maszyny JVM można utworzyć kilka tysięcy współistniejących wirtualnych użytkowników i nie ma potrzeby tworzenia rozproszonej sieci maszyn do przeprowadzania testów [12].

Narzędzie to pozwala określić warunki przeprowadzanych testów pod obciążeniem takich jak: stała lub zmienna liczba użytkowników w każdej sekundzie, czas trwania testu, ustalenie wielkości wzrostu liczby użytkowników w zadanym interwale czasowym, włączenie opcji uśpienia testu w momencie oczekiwania na odpowiedź serwera, gdy potrzebna jest sekwencja żądań do serwera.

Główne właściwości narzędzia Gatling są wymienione poniżej [13]:

- obsługuje HTTP i JMS (Java Message Service),
- skrypty są pisane w języku Scala z wykorzystaniem łatwego w użyciu DSL,
- składnia Scala DSL dobrze współpracuje z IDE (auto-uzupełnianie, refaktoring, kontrola wersji, debugowanie testów) i pozwala na szybkie pisanie i łatwe utrzymywanie istniejących scenariuszy,
- dane testowe do scenariuszy można przekazywać z plików w formacie CSV, TSV, JSON, SSV oraz zbiorów danych Redis,
- udostępnia asercje i sprawdza, czy mogą one być wykonane na otrzymanej odpowiedzi,
- posiada wbudowany rejestrator scenariuszy, który przechwytuje komunikację między przeglądarką, a serwerem,
- generuje bogate i przyjazne, graficzne, łatwo konfigurowalne raporty w formie pliku HTML,
- posiada rozszerzenia do prostej integracji z narzędziami takimi jak Maven, Jenkins i SBT.

Do odwzorowania typowego zachowania użytkowników, testerzy definiują scenariusze w postaci skryptów przekazywanych następnie do Gatlinga. Opisem testu obciążeniowego jest symulacja, która określa, jaki scenariusz będzie realizowany i w jaki sposób dodawani (wstrzykiwani) będą nowi użytkownicy. Wprowadzenie danych do scenariusza może być realizowane z różnych źródeł za pomocą komponentu zwanego feederem, który mapuje zmienne z wartościami i dostarcza je do sesji użytkownika wirtualnego [14].

4. Porównywane szkielety programistyczne

Dla Javy powstało bardzo wiele szkieletów programistycznych, wśród których jedne są bardziej, inne mniej popularne. Ich zadaniem jest m.in. ułatwienie pracy programistom, zwiększenie bezpieczeństwa aplikacji poprzez zastosowanie SSL, a także zaoferowanie pod-

stawowej autoryzacji hasłem i loginem oraz automatyzacja wielu procesów związanych szczególnie z początkową fazą realizacji projektów, takich jak konfiguracja bazy danych.

W ramach pracy została przeprowadzona analiza porównawcza trzech platform programistycznych języka Java: Spring Boot, Micronaut oraz Quarkus. Wybór padł na te szkielety, ze względu na obserwowany w ostatnim czasie wzrost zainteresowania dwoma ostatnimi technologiami, co można potwierdzić coraz większą ilością pojawiających się w sieci artykułów na ich temat, w których są one rekomendowane jako dobra alternatywa dla Spring Boota.

Tabela 1 zawiera wyniki popularności, dla trzech wybranych frameworków. Analiza została przeprowadzona przy pomocy trzech narzędzi: wyszukiwarki Google, serwisu społecznościowego dla programistów Stack Overflow oraz hostingowego serwisu internetowego przeznaczonego dla projektów programistycznych GitHub. W serwisach tych przeprowadzono wyszukiwanie wg słów kluczowych będących nazwami badanych szkieletów. Przedstawione w tabeli wyniki reprezentują odpowiednio liczbę artykułów znalezionych przy użyciu wyszukiwarki Google, liczbę istniejących wątków w serwisie StackOverflow oraz liczbę repozytoriów w systemie GitHub.

Tabela 1: Popularność szkieletów programistycznych języka Java

Szkielet	Rok wydania	Google	Stack Overflow	GitHub
Spring Boot	2004	17 700 000	107 000	249 383
Micronaut	2018	365 000	1 100	4 061
Quarkus	2019	958 000	1 700	8 049

Wyniki te pokazują, że najbardziej popularnym szkieletem jest Spring Boot. Framework ten deklasuje dwie pozostałe technologie. Z kolei szkielet Quarkus ma wyniki popularności około dwa razy wyższe niż Micronaut. Z dużą popularnością danego szkieletu wiąże się wielkość społeczności skupionej wokół technologii, co z kolei przekłada się na łatwiejsze rozwiązywanie potencjalnych problemów, szczególnie w przypadku kiedy oficjalna dokumentacja nie pomaga w ich rozwiązaniu.

4.1. Spring Boot

Spring Boot jest frameworkiem opracowanym z myślą o tworzeniu mikroserwisów [15]. Szkielet ten automatyzuje konfigurację, która w Spring Framework była kopiowana z aplikacji do aplikacji. Szybki start aplikacji sprawia, że framework ten jest wygodny do tworzenia mikroserwisów. W tym celu utworzono stronę start.spring.io [16], miejsce w którym można wygenerować szkielet projektu gotowy do implementacji oraz uruchomienia, z wszystkimi niezbędnymi zależnościami takimi jak komunikacja z bazą danych (Spring Data JPA), czy komunikacja poprzez HTTP przy pomocy architektury REST. Serwis ten pozwala także na wybór narzędzia budującego projekt (Maven lub Gradle) oraz na wybór wersji języka Java, którego będzie można używać w projekcie.

4.2. Micronaut

Micronaut, jest także szkieletem opartym na maszynie wirtualnej Javy, co pozwala na jego integrację z wszystkimi językami opartymi na tej platformie, takimi jak Java, Scala czy Kotlin. Podobnie jak Spring Boot, kładzie on nacisk na szybkie implementowanie aplikacji w architekturze mikroserwisowej. Natomiast w odróżnieniu od Spring Boot, twórcy Micronauta zwrócili szczególną uwagę na niskie zużycie pamięci [17]. Kolejną różnicą jest natywne wsparcie Micronauta dla nowopowstałej platformy GraalVM [18], co nie zostało jeszcze zaimplementowane w Spring Boot, a co ma wpływ na uzyskanie wysokiej wydajności i dostępności tworzonych aplikacji. Podobnie jak Spring Boot, Micronaut posiada mechanizm kontenera aplikacji, ale zarządzanie nim jest zaimplementowane w inny sposób. Spring Boot, korzysta z zalet refleksji języka Java, natomiast Micronaut wykorzystuje procesor adnotacji, co pozwala na szybsze uruchomienie aplikacji oraz wykrywanie błędów już na etapie kompilacji, zamiast w momencie uruchomienia aplikacji, jak to ma miejsce w przypadku szkieletu Spring Boot.

4.3. Quarkus

Quarkus to framework, który jest preferowany do natywnych aplikacji wykorzystujących narzędzie do konteneryzacji o nazwie Kubernetes [19]. Szkielet ten został utworzony ze względu na zwiększające się zapotrzebowanie na aplikacje oparte na mikroserwisach i na aplikacje chmurowe. Quarkus, inaczej niż Spring Boot oraz Micronaut, oferuje w pełni skonfigurowany projekt dostępny na głównej stronie serwisu [20] służący do szybkiego tworzenia szablonu aplikacji. W wygenerowanym projekcie szkielet posiada pliki konfiguracyjne, wśród których jeden o nazwie Dockerfile.jvm zawiera konfigurację aplikacji niezbędną do jej uruchomienia. Quarkus korzysta z narzędzi niedostępnych w Spring Boot i Micronaut, z czego mogą wyniknąć pewne trudności podczas implementacji. Zaletą tego szkieletu jest możliwość budowania lekkich aplikacji pod względem rozmiaru i wykorzystania pamięci [3], uproszczony kod dla typowych zastosowań i możliwość szybkiego przeładowania konfiguracji, dzięki czemu zmiany aplikowane są natychmiast po odświeżeniu uruchomionej aplikacji [21].

5. Metoda badań

5.1. Opis eksperymentu

Analizę porównawczą szkieletów programistycznych Spring Boot, Micronaut i Quarkus przeprowadzono na podstawie trzech scenariuszy badawczych, w ramach których wykonano:

1. pomiar czasów odpowiedzi na żądania POST, GET, PUT i DELETE pod różnym obciążeniem uzależnionymi od liczby wysyłanych żądań przez wirtualnych użytkowników,
2. pomiar niezawodności działania badanych aplikacji testowych,

3. pomiar objętości kodu aplikacji zaimplementowanych na bazie danego szkieletu, polegający na zliczeniu liczby linii kodu.

W pierwszej części eksperymentu zrealizowano scenariusze pierwszy i drugi. Badanie wydajności polegało na wysyłaniu żądań do serwera metodami POST, GET, PUT i DELETE z protokołu HTTP. Testy wydajnościowe i niezawodności przeprowadzono za pomocą biblioteki Gatling. Dokonano pomiarów czasów obsługi żądań, które następnie zostały uśrednione. Po przeprowadzeniu testów, Gatling wygenerował obszerny raport zawierający wyniki i statystyki w postaci atrakcyjnych wykresów i tabel.

Druga część eksperymentu dotyczyła trzeciego scenariusza, w którym dokonano pomiaru objętości kodu aplikacji. Czynność ta została w bardzo prosty sposób zrealizowana za pomocą wtyczki Statistics dostępnej w zintegrowanym środowisku programistycznym IntelliJ IDEA Ultimate 2020.3. Otrzymane wyniki po analizie kodu trzech aplikacji zaprezentowano na wykresie na rysunku 7.

5.2. Środowisko testowe

W tabeli 2 przedstawiono środowisko testowe, za pomocą którego przeprowadzono badania. Oprócz parametrów komputera, znajdują się tutaj wersje analizowanych szkieletów programistycznych oraz wersje narzędzi użytych do realizacji testów.

Tabela 2: Środowisko testowe

Sprzęt	
Procesor	Intel® Core™ i5-9300H
Pamięć RAM	16GB
Karta sieciowa	Killer E2500 Gigabit Ethernet Controller
System operacyjny	Windows 10 Pro
Oprogramowanie	
Spring Boot	2.4.4
Micronaut	2.5.4
Quarkus	1.13.4.Final
Gatling	3.5.1
Statistic	4.1.7

5.3. Pomiar czasów obsługi żądań POST

W pierwszym scenariuszu badawczym, czas odpowiedzi serwera na żądanie był mierzony od momentu jego wysłania przez aplikację kliencką, do prawidłowej i zakończonej powodzeniem odpowiedzi serwera, sygnalizowanej zwróceniem statusu HTTP 200. Dla każdej aplikacji utworzonej za pomocą poszczególnych szkieletów pięć razy powtórzono scenariusz dla różnych obciążeń tzn. dla 500, 1000, 2000, 5000 oraz 10000 użytkowników, którzy w ciągu minuty trwania testu wysyłali po 60 żądań, czyli średnio jedno żądanie na sekundę.

Obiektami badań były trzy aplikacje testowe, z których każda została zbudowana za pomocą innego szkieletu programistycznego języka Java. Aplikacje te realizują cztery podstawowe operacje na bazie danych.

Kontrolery trzech aplikacji testowych korzystają ze wspólnej klasy projektu `UserFolderRepository`, której metodę o nazwie `save` zaprezentowano na listingu 1.

Listing 1: Kod wspólnego repozytorium dla każdej aplikacji testowej

```
@RequiredArgsConstructor
public abstract class UserFolderRepository {
  private final DSLContext dslContext;

  protected void save(CreateUserFolderRequest request) {
    final var address = request.getAddress();
    dslContext.insertInto(USER_FOLDER)
        .columns(USER_FOLDER.LOGIN,
            USER_FOLDER.NAME,
            USER_FOLDER.SECOND_NAME,
            USER_FOLDER.SURNAME,
            USER_FOLDER.EMAIL,
            USER_FOLDER.AGE,
            USER_FOLDER.ADDRESS_LINE1,
            USER_FOLDER.ADDRESS_LINE2,
            USER_FOLDER.CITY,
            USER_FOLDER.COUNTRY_SUBDIVISION,
            USER_FOLDER.POSTAL_CODE,
            USER_FOLDER.COUNTRY
            )
        .values(request.getLogin(),
            request.getName(),
            request.getSecondName(),
            request.getSurname(),
            request.getEmailAddress(),
            request.getAge(),
            address.getLine1(),
            address.getLine2(),
            address.getCity(),
            address.getCountrySubdivision(),
            address.getPostalCode(),
            address.getCountry())
    .execute();
  }
}
```

W pomiarach czasów, kluczową rolę odegrała biblioteka Gatling, za pomocą której utworzono oddzielną aplikację kliencką, w ramach której utworzono i wykonano po 100 symulacji dla każdej aplikacji testowej (po pięć powtórzeń dla czterech typów operacji CRUD i dla pięciu rodzajów obciążenia).

Listing 2 pokazuje fragment kodu zawierający zmienną httpProtocol, za pomocą której komponowane jest żądanie POST, zawierające dane klienta przesyłane do bazy.

Listing 2: Zapis scenariusza badawczego Gatlingu

```
val httpProtocol: HttpProtocolBuilder = http
  .baseUrl("http://localhost:8082")
  .header("Accept", "application/json")
  .header("Content-Type", "application/json")

setUp(
    scenario("Test Spring Boot app performance for 1000 users")
    .exec(
      http("Create User")
        .post("/spring-boot/users")
        .body(StringBody("{\n  \"login\": \"qwerty\",\n  \"name\": \"Jan\",
          \n  \"age\": \"18\",\n  \"secondName\": \"Tadeusz\",
          \n..\"surname\": \"Kowalski\",
          \n  \"emailAddress\": \"jan.kowalski@email.com\",
          \n  \"address\": {\n  \"line1\": \"ul. Debowa 32\",
          \n  \"line2\": \"Lublin, 21211\",\n  \"city\": \"Lublin\",
          \n  \"countrySubdivision\": \"Lubelskie\",
          \n  \"postalCode\": \"21000\",\n  \"country\": \"Polska\"\n  }\n}"))
    )
    .inject(constantUsersPerSec(1000) during 1.minutes)
    .protocols(httpProtocol),
)
```

Metoda `setUp` służy do konfiguracji symulacji. Definicja scenariusza rozpoczyna się od słowa `scenario`, której parametrem jest jego nazwa, natomiast parametrem części `http` jest nazwa danego żądania. Słowo `post` określa typ metody, która zostanie wykonana, z parametrem który zostanie połączony z właściwością `baseUri` zmiennej `httpProtocol`. Część `body` definiuje ciało żądania, a metoda `inject` umożliwia ustawienie liczby wirtualnych użytkowników - która może być stała lub zmienna oraz pozwala na określenie czasu trwania testu.

5.4. Pomiar niezawodności aplikacji

Wyzwaniem dla każdej aplikacji serwerowej jest osiągnięcie jak najwyższej niezawodności określanej jako najmniejsza liczba żądań zakończonych niepowodzeniem czyli takich, w przypadku których serwer zwracał status inny niż HTTP 200. Niezawodność jest wyrażana procentowo. Pomiar tego wskaźnika został przeprowadzony jednocześnie podczas badań wydajności w scenariuszu 1.

5.5. Pomiar objętości kodu

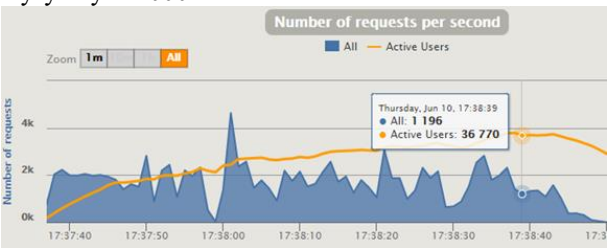
W scenariuszu 3 przeprowadzono oddzielnie pomiar liczby linii kodu trzech porównywanych aplikacji testowych, które realizowały te same zadania. Aplikacje były punktem końcowym serwera służącym do zapisu danych o użytkownikach w bazie wraz z ich adresem. Części wspólne kodu takie jak klasa repozytorium i klasa odwzorowująca ciało żądania zostały pominięte. Wiersze zawierające białe znaki, takie jak znak nowej linii czy nawiasy klamrowe, jeśli występowały samodzielnie w danej linii, także nie były zliczane.

6. Wyniki badań

6.1. Pomiar czasów odpowiedzi serwera na żądania POST

Na Rysunkach 1 i 2 pokazano fragment raportu w postaci wykresu i tabeli generowanego przez narzędzie Gatling, po wykonaniu pojedynczego badania wydajności przeprowadzonego według scenariusza 1 dla aplikacji zbudowanej za pomocą Spring Boota pod obciążeniem 2000 żądań na sekundę. Takie badania zostały powtórzone 15-krotnie (dla trzech aplikacji i pięciu rodzajów obciążeń).

Wykres z Rysunku 1 pokazuje, przypadek kiedy średnio było wysyłanych 2000 żądań przez aktywnych użytkowników, czyli po jednym żądaniu na sekundę. Jak widać, zdarzały się takie sytuacje, kiedy przez kilka sekund użytkownicy nie wysłali żadnego żądania, a w kolejnej chwili wysłali ich jednocześnie ponad 2000. Oznacza to, że w ciągu 60 sekund, w czasie trwania testu, 2000 użytkowników wysyłało po 2000*60 żądań. Liczbę wysyłanych żądań w ciągu całego testu obrazuje na rysunku 1 niebieska krzywa. Przyglądając się jej przebiegowi, widać, że czasem liczba wysyłanych żądań spadała prawie do zera, a momentami przekraczała 4000. Z kolei pomarańczowa linia oznacza liczbę aktywnych użytkowników, wśród których w danej chwili tylko część wysyła żądania. Liczba wysyłanych żądań zmieniała się w trakcie trwania testu (60 sekund), ale średnio w ciągu każdej sekundy było ich wysyłanych 2000.



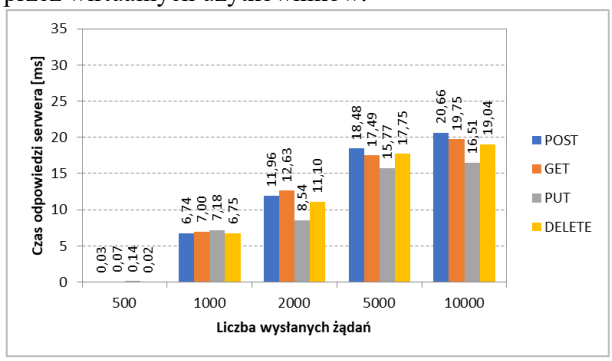
Rysunek 1: Fragment raportu otrzymany z narzędzia Gatling przy obciążaniu 2000 żądań/s dla aplikacji wykonanej za pomocą Spring Boot.

Na Rysunku 2 widoczny jest inny fragment raportu w postaci tabeli z wynikami testów. Znajdują się tutaj informacje o całkowitej liczbie wysłanych żądań, liczbie żądań poprawnie obsłużonych oraz tych, których obsługa zakończyła się niepowodzeniem. Ponadto tabela zawiera szereg danych statystycznych dotyczących czasów obsługi żądań takich jak średnia, mediana, czas minimalny i maksymalny oraz percentyle. Ostatnia metryka wskazuje wartości, poniżej których znajduje się pewien procent danych w zbiorze danych. Na przykład 95-ty percentyl to wartość, dla której 95% przypadków było lepszych tzn. odpowiedź z systemu trwała krócej, a 5% było gorszych czyli trwały dłużej.

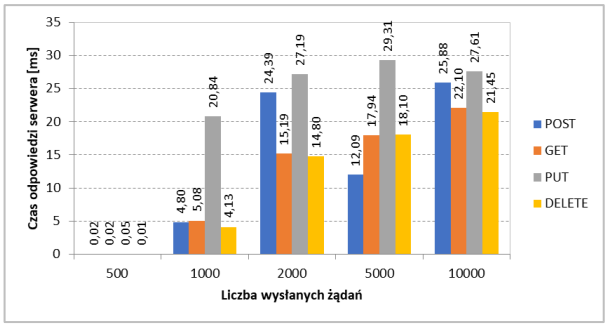
Requests	Executions					Response Time (ms)					
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max
Global Information	60000	36776	23224	39%	600	48	10335	16713	31004	33500	38317
Create User	60000	36776	23224	39%	600	48	10343	16717	31006	33500	38317

Rysunek 2: Fragment tabeli z raportu w postaci tabeli wygenerowany przez narzędzie Gatling.

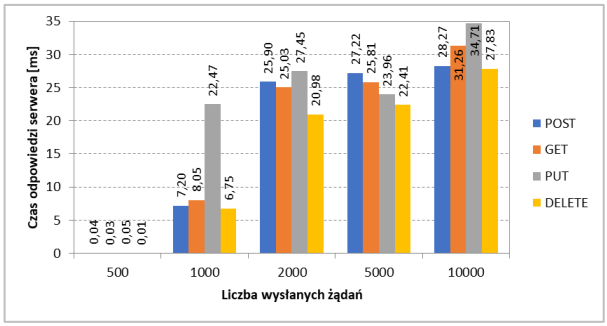
Na Rysunkach 3, 4 i 5 przedstawiono zagregowane wyniki dla scenariusza 1, w ramach którego przeprowadzono testy punktu końcowego podczas którego wykorzystano metody POST, GET, PUT i DELETE przy różnych obciążenia, począwszy od 500, następnie 1000, 2000, 5000 oraz 10000 żądań na sekundę wysyłanych przez wirtualnych użytkowników.



Rysunek 3: Średnie czasy odpowiedzi serwera na cztery typy żądań przy ustalonym obciążeniu dla aplikacji testowej opartej na szkielecie Spring Boot.



Rysunek 4: Średnie czasy odpowiedzi serwera na cztery typy żądań przy ustalonym obciążeniu dla aplikacji testowej opartej na szkielecie Micronaut.



Rysunek 5: Średnie czasy odpowiedzi serwera na cztery typy żądań przy ustalonym obciążeniu dla aplikacji testowej opartej na szkielecie Quarkus.

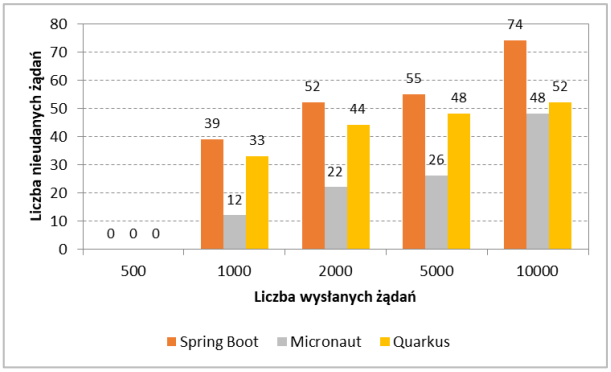
Powyższe rysunki pokazują średnie czasy obsługi poszczególnych rodzajów żądań przy różnych wielkościach obciążenia.

Z wykresów tych widać, że przy niskim obciążeniu (500 żądań/s), choć różnice w wydajności aplikacji opartych na poszczególnych szkieletach są bardzo małe, najlepszy okazał się szkielet Micronaut. Dla obciążenia 1000 żądań/s w przypadku operacji POST, GET i DELETE także najlepsze wyniki osiągnął Micronaut. Wyjątkiem jest tu operacja aktualizująca dane (żądanie PUT), która wymagała znacznie dłuższego czasu obsługi w aplikacjach opartych na szkieletach Micronaut i Quarkus. Przyczyną tego stanu rzeczy może być to, że w sytuacji gdy odsetek nieudanych żądań jest niewielki, żądania te ustawiane są ponownie w kolejce i w ten sposób dodatkowo obciążają serwer.

Dla wyższych obciążeń tzn. 2000, 5000 i 10000 żądań/s najlepsze czasy obsługi czterech rodzajów żądań osiągnął szkielet Spring Boot. W przypadku obciążenia 5000 żądań/s dobre wyniki osiągnął także Micronaut, choć były one o 11% gorsze od wyników szkieletu Spring Boot.

6.2. Pomiar niezawodności aplikacji

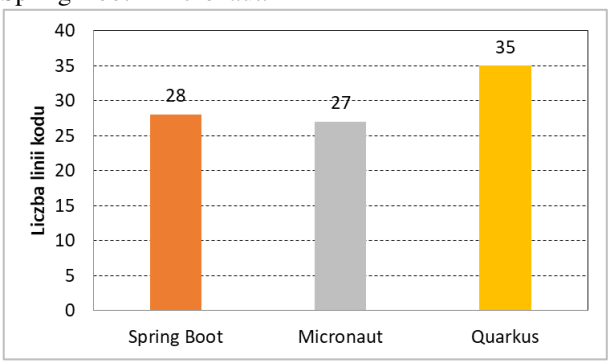
Na Rysunku 6 zaprezentowano niezawodność wyrażoną jako liczba nieobsłużonych żądań. Wykres pokazuje, że dla obciążenia 500 żądań/s trzy aplikacje testowe obsługiwały wszystkie żądania. Dla obciążeń 1000, 2000, 5000 i 1000 żądań/s liczba tych nieobsłużonych była największa w przypadku aplikacji utworzonej na bazie Spring Boot. Trochę lepsze wyniki uzyskała aplikacja wykonana za pomocą szkieletu Quarkus. Natomiast najlepszy wynik, oznaczający najmniejszą liczbę nieobsłużonych żądań, osiągnęła aplikacja oparta na platformie programistycznej Micronaut.



Rysunek 6: Wyniki niezawodności testowanych aplikacji.

6.3. Pomiar objętości kodu

Liczba linii kodu wymaganego do utworzenia aplikacji była brana pod uwagę przy porównaniach badanych szkieletów. Z Rysunku 7 wynika, że najmniejszą liczbę linii kodu niezbędną do osiągnięcia tego samego efektu miała aplikacja utworzona za pomocą szkieletu Micronaut, a najwięcej aplikacja oparta na szkielecie Quarkus. Wynika to m.in. z tego, że w Micronaut pousuwano zbędne według jego twórców adnotacje, które nadal znajdują się w szkielecie Spring Boot. Jako przykład można tu podać adnotację @RequestMapping, którą włączono w Micronaut bezpośrednio do adnotacji @Controller. Poza tym w Micronaut nie ma konieczności użycia adnotacji do wykonania auto-konfiguracji, ponieważ szkielet rozwiązuje to samoistnie - domyślnie ustawiając odpowiednią flagę. Natomiast szkielet Quarkus, jest najbardziej konfigurowalny, stąd potrzebne było dodanie adnotacji określających typ danych przyjmowanych oraz zwracanych - w tym przypadku JSON. Robi się to za pomocą pary adnotacji @Produces oraz @Consumes. Dodatkowo w aplikacji zbudowanej na frameworku Quarkus konieczne było utworzenie klasy implementującej abstrakcyjne repozytorium, co czyniło jej implementację trudniejszą od aplikacji opartej na Spring Boot i Micronaut.



Rysunek 7: Liczba linii kodu dla każdego szkieletu.

7. Wnioski

Decyzja dotycząca wyboru optymalnej technologii dostosowanej do rozpoczynanego przedsięwzięcia programistycznego jest bardzo istotna, ponieważ niesie ze

sobą trudne do przewidzenia konsekwencje w przyszłości. Niniejsza praca, skupiająca się na wydajności i niezawodności trzech wybranych platform języka Java, miała na celu ułatwienie podjęcia tej decyzji. W analizie porównawczej szkieletów Spring Boot, Micronaut i Quarkus wzięto pod uwagę cztery kryteria: popularność, wydajność, niezawodność działania oraz objętość kodu.

Biorąc pod uwagę kryterium popularności, framework Spring Boot okazał się liderem według trzech użytych do analiz narzędzi. Popularność tego szkieletu wiąże się przede wszystkim z jego długim istnieniem na rynku trwającym nieprzerwanie od 2004 roku. Pozostałe dwa szkielety: Quarkus i Micronaut powstały stosunkowo niedawno – odpowiednio 2 i 3 lata temu.

Porównując otrzymane wyniki wydajności, można stwierdzić, że szkielet Micronaut osiągnął najlepsze rezultaty w niskich warunkach obciążeniowych. Przy większych obciążeniach najlepiej wypadł Spring Boot, a najgorsze wyniki osiągnął Quarkus.

Pod względem niezawodności jednoznacznie najlepszą platformą programistyczną okazał się Micronaut. W każdych warunkach obciążeniowych aplikacja zbudowana na bazie tego szkieletu była bardziej niezawodna niż aplikacje zbudowane na szkieletach Spring Boot i Quarkus. Biorąc pod uwagę wydajność, Micronaut był lepszy dla mniejszych obciążeń, natomiast dla większych przegrywał z szkieletem Spring Boot. W ostatniej kategorii dotyczącej objętości kodu również Micronaut okazał się najlepszym wyborem.

Z przeprowadzonej analizy można wysnuć generalny wniosek, że najlepszym z trzech badanych szkieletów programistycznych okazał się Micronaut. Dokonana analiza ma jednak swoje ograniczenia, ponieważ dotyczyła bardzo prostej aplikacji testowej, która wykonywała cztery proste czynności – obsługę żądań zapisu, odczytu, aktualizacji i usunięcia danych z bazy. W celu dokonania dokładniejszej oceny technologii należy poszerzyć spektrum badań i uwzględnić jeszcze inne aspekty.

Literatura

[1] E. K. Smyk, Overview of technologies and methods designed to build Java Enterprise web applications. Comparison of Spring and Play Frameworks based on proprietary application (praca magisterska), Politechnika Warszawska, 2014.

[2] P. Dutta, V. Gupta, S. Rana, Performance Comparison on Java Technologies – A Practical Approach, Centre for development of Advanced Computing, Third International Conference on Computer Science, Engineering & Applications (2013) 349-357, https://doi.org/10.5121/CSIT.2013.3536.

[3] M. Šipek, D. Muharemagić, B. Mihaljević, A. Radovan, Enhancing Performance of Cloud-based Software Applications with GraalVM and Quarkus, 43rd International Convention on Information, Communication and Electronic Technology (MIPRO) (2020) 1746-1751, DOI: 10.23919/MIPRO48935.2020.9245290.

[4] H. K. Dhalla, Performance Comparison of RESTful Applications Implemented in Spring Boot Java and MS.NET Core, Journal of Physis: Conference Series 1933 (2021), https://doi.org/10.1088/1742-6596/1933/1/012041.

[5] M. Pucek, M. Błaszczyk, P. Kopniak, Porównanie lekkich szkieletów dla języka Java poprzez analizę autorskich aplikacji internetowych, Journal of Computer Sciences Institute 19 (2021) 159-164.

[6] TIOBE Index, https://www.tiobe.com/tiobe-index/, [02.07.2021].

[7] Oracle Java SE Support Roadmap, https://www.oracle.com/java/technologies/java-se-support-roadmap.html, [06.07.2021].

[8] M. Masse, REST API Design Rulebook, O'Reilly Media, 2012.

[9] B. Miłosierny, M. Dzieńkowski, Analiza porównawcza szkieletów do budowy aplikacji internetowych w ekosystemie Node.js, Journal of Computer Sciences Institute 18 (2021) 42-48.

[10] M. Herber, Gatling. Testy wydajnościowe w innej formie, https://testerzy.pl/baza-wiedzy/gatling-testy-wydajnosci-w-innej-formie-czesc-1, [02.07.2021].

[11] Xie A., Performance Testing Tutorial: Automation, Gatling, and Jenkins, https://www.educative.io/blog/performance-testing-tutorial-gatling-jenkins, [21.07.2021].

[12] Lee G., Gatling Load Testing: How-To, Distributed Tests & Examples, https://www.loadview-testing.com/blog/gatling-load-testing-how-to-distributed-tests-examples/, [21.07.2021].

[13] A. Ludwikowski, Gatling vs JMeter – czego użyć do testowania wydajności, https://softwaremill.com/gatling-vs-jmeter-testy-wydajnosci/, [02.07.2021].

[14] Gatling, https://gatling.io/docs/gatling/reference/current/general/concepts/, [21.07.2021].

[15] B. Nius, Jak Spring Boot ułatwia tworzenie aplikacji w Javie? https://global4net.com/ecommerce/jak-spring-boot-ulatwia-tworzenie-aplikacji-w-javie/, [05.12.2019].

[16] Spring Initializr, https://start.spring.io/, [02.07.2021].

[17] P. Bykowski, Micronaut – framework dedykowany dla mikroserwisów, https://bykowski.pl/micronaut-framework-dedykowany-dla-mikroserwisow/, [17.10.2019].

[18] Micronaut, https://micronaut.io/, [03.07.2021].

[19] Quarkus – Supersonic Subatomic Java, https://quarkus.io/, [05.07.2021].

[20] Quarkus – start coding with code.quarkus.io, https://code.quarkus.io/, [06.07.2021].

[21] Oficjalna dokumentacja szkieletu Quarkus, https://quarkus.io/, [28.02.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 295–302
Received: 01 September 2021
Accepted: 19 September 2021

Usability analysis taking into consideration the aspects of accessibility of selected university websites

Analiza użyteczności z uwzględnieniem aspektów dostępności wybranych serwisów internetowych uczelni wyższych

Karol Kałan*, Damian Karpiuk*, Mariusz Dzieńkowski

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The goal of this paper was to evaluate selected web services of universities in terms of user experience, with particular emphasis on usability and accessibility. The research was conducted using eye-tracking and questionnaire methods. Ten people participated in this study. The objects of the study were three university websites: the Catholic University of Lublin (KUL), the Cracow University of Technology (PK) and the West Pomeranian University of Technology (ZUT). The eye-tracking data were subjected to qualitative and quantitative analyses, while the data from questionnaires were subjected to quantitative analysis. The results of individual analyses are presented in the form of heat maps, scan paths, charts and tables.

Keywords: user experience; usability; accessibility; eye tracking

Streszczenie

Celem pracy była ocena wybranych serwisów internetowych uczelni wyższych pod względem doświadczenia użytkownika, ze szczególnym uwzględnieniem użyteczności i dostępności. Badania wykonano metodą eyetrackingową oraz kwestionariuszową. Uczestniczyło w nich dziesięć osób. Obiektem badań były trzy serwisy internetowe uczelni wyższych: Katolickiego Uniwersytetu Lubelskiego (KUL), Politechniki Krakowskiej (PK) oraz Zachodniopomorskiego Uniwersytetu Technicznego (ZUT). Dane eyetrackingowe zostały poddane analizie jakościowej i ilościowej, natomiast dane z ankiet analizie ilościowej. Wyniki poszczególnych analiz przedstawiono w formie map cieplnych, ścieżek skanowania, wykresów oraz tabel.

Słowa kluczowe: user experience; użyteczność; dostępność; eye tracking

*Corresponding author

Email address: karol.kalan@pollub.edu.pl (K. Kałan), damian.karpiuk@pollub.edu.pl (D. Karpiuk)

1. Wstęp

Organizacja jaką jest uczelnia wyższa, powinna udostępniać informacje jak najszerszemu gronu odbiorców, a szczególnie kandydatom na studia, studentom oraz interesariuszom zewnętrznym. Strony internetowe, Biuletyn Informacji Publicznej, serwisy społecznościowe są pewnego rodzaju interfejsem między uczelnią a użytkownikami, stanowiąc swego rodzaju wizytówkę, dostarczając niezbędnych treści oraz umożliwiając różne formy kontaktu. Wśród użytkowników mogą być także osoby niepełnosprawne i dlatego trzeba zapewnić im możliwość bezproblemowego dotarcia do informacji zawartych w serwisach www. Uczelnia państwowa jako instytucja publiczna jest do tego zobligowana prawem.

O tym, że interfejs oprogramowania jest użyteczny, świadczy intuicyjna nawigacja, zrozumiała dla użytkownika komunikacja oraz zastosowanie ułatwień umożliwiających skuteczne poszukiwanie informacji. Jakob Nielsen, specjalista w dziedzinie projektowania interfejsów graficznych, wyróżnia pięć cech użyteczności, które są związane z produktywnością. Zaliczają się do nich: łatwość nauki, efektywność, zapamiętywalność, stopa błędów oraz satysfakcja [1]. Elementami użyteczności są różne narzędzia, w które wyposażone są witryny internetowe, wspomagające dostępność serwisu dla użytkowników, w szczególności niepełnosprawnych. Może to być łatwy dostęp do wyszukiwarki czy przycisk zatrzymujący czyli element witryny internetowej, który wyświetla na przemian banery graficzne (ang. slider). Te i inne elementy mają za zadanie uczynić serwis dostępnym czyli przystosowanym dla osób z różnymi typami niepełnosprawności, które mogą z niego korzystać bez większych przeszkód.

Wśród rodzajów niepełnosprawności człowieka wyróżnia się: niepełnosprawność wzrokową (ślepota, monochromatyzm, jaskra, zaćma, itd.), słuchową, poznawczą (problemy uczenia się, zapamiętywania, utrzymywania uwagi etc.), ruchową (trudności lub niemożność korzystania z kończyn). Istnieją technologie wspomagające i adaptacyjne, które w pewnym zakresie przeciwdziałają ograniczeniom. W przypadku zaburzenia widzenia w kontekście serwisów www mogą to być takie techniki jak powiększanie czcionki, zmiana kontrastu, czytniki ekranu. W przypadku osób niesłyszących lub słabosłyszących stosuje się napisy na materiałach audiowizualnych lub transkrypcję w niewerbalnym języku migowym. Użytkownicy strony, którzy mają problemy poznawcze powinni mieć możliwość łatwego znalezienia odpowiednio opisanych znaczników wszystkich funkcji danej witryny. Dla osób z niedowładem kończyn

górnych strony www mogłyby zawierać dodatkowe rozwiązania takie jak odpowiednio powiększone elementy nawigacyjne, umożliwienie pełnego korzystania ze strony wyłącznie za pomocą klawiatury zwłaszcza osobom, które nie są w stanie korzystać z myszy.

Uczelnie wyższe posiadające własne serwisy internetowe jako podmioty publiczne są zobowiązane do stosowania ustawy z dnia 4 kwietnia 2019 roku, w której zawarte są informacje mówiące o wymogach dotyczących kwestii dostępności. Ustawa jasno informuje, jakie wytyczne powinny być przestrzegane dla dostępności witryn internetowych oraz aplikacji mobilnych. Pierwszą z nich jest postrzegalność mówiąca o umożliwieniu korzystania użytkownikowi z serwisu za pomocą zmysłu słuchu, dotyku lub wzroku. Kolejną zasadą jest zrozumiałość, która musi gwarantować zrozumienie informacji tak, aby treści przedstawione były zrozumiałe dla każdego odbiorcy. Ostatnią zasadą jest kompatybilność odpowiadająca za umożliwienie integracji zawartości jak największej ilości urządzeń oraz programów wspomagających osoby niepełnosprawne.

Wśród wielu metod badania UX, do najpopularniejszych należą: wywiady pogłębione, testy użyteczności, sortowanie kart czy eyetracking. Wywiady pogłębione polegają na przeprowadzeniu rozmów z grupami docelowymi w celu zdobycia informacji np. o tym co można poprawić na stronie internetowej, aby była bardziej przyjazna w użytkowaniu. Często z wywiadami pogłębionymi są łączone testy użyteczności, które pozwalają określić czy dana witryna jest przystępna w użytkowaniu. Kolejną metodą badań jest tzw. sortowanie kart. Jest to zespołowa metoda klasyfikacji informacji, polegająca na wspólnym uporządkowaniu informacji, zwykle umieszczonych na samoprzylepnych karteczkach. Umożliwia stworzenie architektury informacji, struktury zadań czy też układu menu i może być stosowana w grupie lub poprzez serię indywidualnych sortowań z następującym potem uogólnieniem wyników [2]. Interesującą i coraz częściej stosowaną metodą badań UX jest eyetracking. Badania eyetrackerem przynoszą wiele cennych informacji, które są zobiektywizowane. Wyniki z eyetrackera, takie jak np. mapy cieplne, pozwalają określić, na jakich elementach użytkownik skupiał najwięcej uwagi oraz jakie części strony były przez niego pomijane.

Strony internetowe instytucji publicznych powinny być jak najbardziej użyteczne dla wszystkich użytkowników. Niniejsza praca skupia się na kwestii użyteczności z uwypukleniem problematyki dostępności. Osoby z niepełnosprawnościami powinny mieć pełną możliwość dostępu do zamieszczanych treści. Witryny uczelni wyższych poddane badaniom w ramach niniejszej pracy zostały tak dobrane, aby zawierały narzędzia, które czynią je dostępnymi. Szczegółowy zakres tej dostępności jest określony w specjalnych dokumentach umieszczonych w każdym z serwisów, w tzw. deklaracjach dostępności. Badania w pracy dotyczą głównie ergonomii, rozmieszczenia elementów dostępności, a także łatwości nawigacji po serwisie oraz łatwości dostępu do wbudowanej wyszukiwarki.

2. Przegląd literatury

Kompleksowe opisy aspektów związanych z użytecznością takich jak skuteczność, wydajność i satysfakcja oraz powiązanych z nimi metryk zawarte są w pracy [3]. Efektywność, kojarzona z produktywnością, jest funkcją pracy wykonanej w jednostce czasu. Główne miary efektywności to czas wykonywania zdania (często ważony na liczbę błędów) oraz średni czas ukończenia zadania. Skuteczność określa czy i w jakim stopniu zadanie zostało poprawnie wykonane przez użytkownika. Powszechną miarą dla tego aspektu jest procent wykonania zadań także ważony dla błędów. Natomiast zadowolenie jest określane na podstawie subiektywnych miar i jest ono oceniane po zakończeniu danego zadania lub na koniec testu użyteczności. Zadowolenie odnosi się do takich pojęć jak przydatność, użyteczność, zrozumiałość i estetyka [4].

Główne miary stosowane w badaniach okulograficznych dotyczą dwóch stanów: fiksacji i sakad. Sakadami określa się szybkie ruchy oczu występujące między fiksacjami – momentami, w których oczy są stosunkowo nieruchome. Chociaż istnieje kilka innych rodzajów ruchów oczu, sakadyczne ruchy, składające się z sakad i fiksacji, są najczęściej spotykane w badaniach doświadczenia użytkowników. Fiksacja jest główną miarą wykorzystywana w obszarach HCI i UX, ponieważ dostarcza użytecznych informacji na temat uwagi wzrokowej, z tego względu, że w większości przypadków fiksacja zbiega się z uwagą [5]. Istnieje również wiele metryk pochodnych, które wywodzą się z tych podstawowych miar np.: miary związane z uwagą, ścieżkami skanowania oraz zmianą wielkości źrenicy czy częstością mrugnięć. Na przykład miara rozszerzenia źrenic została powiązana z wysiłkiem umysłowym i uwagą [6].

Większość badań użyteczności z wykorzystaniem eyetrackingu miało na celu wypracowanie w szerokim ujęciu, szczegółowych i jak najlepszych rekomendacji projektowych. Pozyskiwane były głównie dane dotyczące nawigacji, wyszukiwania i innych interakcji z aplikacjami online. Miały one wpływ na architekturę interfejsu użytkownika oprogramowania, projekt i zawartość ekranów i menu, lokalizację i typ elementów wizualnych oraz wybór stylu wizualizacji interfejsu użytkownika [4].

Zalecenia dotyczące wielkości ikon dla dobrze widzących i niedowidzących użytkowników zostały przedstawione w pracy [7], w ramach której osoby badane wyszukiwały ikony o różnych rozmiarach. W analizach pod uwagę brano sześć fiksacji i czas trwania ścieżek skanowania. We wnioskach autorzy proponują używanie różnych minimalnych rozmiarów ikon, które są uzależnione od jasności tła wyświetlacza.

Autorzy pracy [8] oceniali portal internetowy przy pomocy techniki śledzenia wzroku, umożliwiając użytkownikom swobodne poruszanie się po stronach internetowych i między nimi. Okazało się, że wizualne aplety wymagające dużej zauważalności należy umieszczać w lewej kolumnie na stronie portalu. Natomiast w przypadku odsyłaczy „Dostosuj” i „Więcej” znajdujących

się w nagłówkach tych aplikacji, korzystnie byłoby je również dodać pod treścią portalowych apletów.

W ostatnim czasie badania eyetrackingowe są coraz chętniej oraz szerzej wykorzystywane przy ocenianiu użyteczności i ergonomii stron internetowych. Wiele firm ma w swojej ofercie badania ergonomii, które są wspierane przez analizę eyetrackingową. Wynika to z faktu, że wyżej wymieniona technika daje obiektywne dane liczbowe, oraz przydatne materiały graficzne.

3. Cel i zakres pracy

Celem pracy jest wieloaspektowa ocena użyteczności serwisów internetowych wybranych uczelni wyższych, ze szczególnym uwzględnieniem aspektów dostępności. W badaniach skupiono się na elementach użyteczności i dostępności stron www i zrealizowano je dwuetapowo.

Najpierw w laboratorium zostały zrealizowane badania eyetrackingowe. Po nich uczestnicy wypełniali ankietę, która dotyczyła satysfakcji użytkowników podczas interakcji z wybranymi serwisami.

Zakres pracy obejmował wybór metod badawczych oraz obiektów do badań – serwisów internetowych, opracowanie zadań dla użytkowników, zaprojektowanie i przeprowadzenie eksperymentu, analizę i uogólnienie wyników oraz sformułowanie wniosków.

4. Metoda badawcza

W pracy zastosowano dwie metody oceny jakości interfejsów: metodę okulograficzna oraz kwestionariuszową. Przygotowany eksperyment składał się z dwóch części: pierwszej z wykorzystaniem eyetrackera oraz drugiej, podczas którego użytkownicy zdalnie wypełniali ankietę.

Etapy eksperymentu realizowanego metodą eyetrackingową:

1. Określenie problematyki badań, celu, pytań i hipotez badawczych
2. Selekcja obiektów do badań – serwisów www uczelni
3. Opracowanie zadań dla użytkowników do realizacji podczas eksperymentu
4. Zaprojektowanie eksperymentu w iMotions
5. Badanie pilotażowe
6. Dodawanie kolejnych uczestników badań
7. Nagrywanie respondentów
8. Obróbka zgromadzonych danych i ich analiza
9. Uogólnienie wyników - ocena serwisów, wnioski

Analiza danych po badaniach eyetrackingowych miała charakter jakościowy i ilościowy. W analizie jakościowej wzięto pod uwagę mapy ciepła oraz ścieżki skanowania. Natomiast w analizie ilościowej w celach porównawczych interfejsów wykorzystano następujące metryki:

- czas ukończenia zadań (znalezienia celu),
- liczba fiksacji do momentu osiągnięcia celu,
- liczba niepoprawnie zrealizowanych zadań (nieznalezienie obiektu na stronie).

4.1. Obiekty badań

Proces wyboru serwisów internetowych do badań był bardzo długi i wieloetapowy. Na początku wybrano 15 serwisów, ale po głębszej analizie zdecydowano się na trzy. Autorom zależało na tym, żeby wszystkie strony zawierały określone funkcjonalności, żeby zadania dla użytkowników były powtarzalne. Witryny zostały wybrane w taki sposób, aby na stronie głównej była możliwość wyboru rozmiaru czcionki, kontrastu, przejścia na stronę biblioteki oraz na stronę dla osób z niepełnosprawnościami.

Tabela 1: Badane obiekty – wyselekcjonowane serwisy uczelni

Lp.	Uczelnia	Adres strony
1	Politechnika Krakowska (PK)	https://www.pk.edu.pl
2	Zachodniopomorski Uniwersytet w Szczecinie (ZUT)	https://www.zut.edu.pl/uczelnia/aktualnosci.html
3	Katolicki Uniwersytet Lubelski (KUL)	https://www.kul.pl/

4.2. Grupa badawcza

W badaniach wzięło udział 10 osób. Byli to studenci ostatniego roku studiów kierunku Informatyka z Politechniki Lubelskiej. Uczestnicy badań wcześniej nie używali wybranych i analizowanych w pracy serwisów internetowych.

4.3. Stanowisko badawcze

Eksperyment został przeprowadzony w laboratorium Katedry Informatyki Politechniki Lubelskiej, w którym zapewnione były optymalne warunki oświetleniowe. Uczestnicy mieli pewną swobodę ruchów, podczas badań siedzieli na fotelu, który umożliwiał regulację wysokości i odległości od eyetrackera. Badania były prowadzone przy udziale moderatora.

Rysunek 1: Stanowisko badawcze.

Parametry techniczne eyetrackera stacjonarnego Gazepoint GP3 HD [9]:

- częstotliwość próbkowania 60Hz lub 150Hz śledzenie obuoczne)
- kalibracja 5- lub 9- punktowa
- detekcja stanów oka: fiksacji, sakad, zmian średnicy źrenicy
- technika śledzenia: jasna źrenica
- zakres swobody poruszania głową: 35x22cm
- zakres odległości oczu od eyetrackera: ~65cm
- dokładność w idealnych warunkach: 0,5-1° obuocznie

Oprogramowanie: iMotions 9.0 umożliwia [10]:

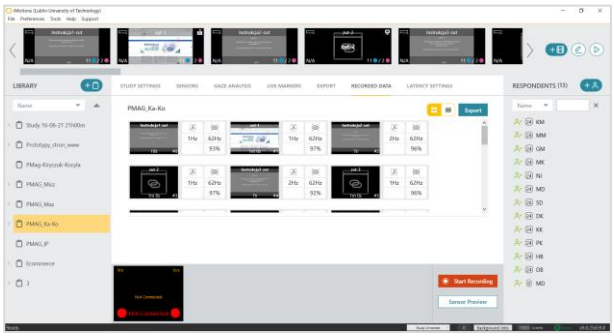
- projektowanie eksperymentu
- kalibracja: 5- lub 9- punktowa
- nagrywanie sesji uczestników badań
- odtwarzanie i edycję nagrań
- wizualizację wyników: mapy cieplne, mapy uwagowe, ścieżki fiksacji, obszary zainteresowań, roje pszczół
- eksport surowych danych
- generowanie statystyk

Parametry komputera: laptop ThinkPad T540p

- procesor: Intel Core i7-4710MQ (2,50GHz)
- pamięć RAM: 16GB
- system operacyjny: Windows 10 (64-bit)
- karta graficzna: 1G NVIDIA GeForce N14M-GS 730M
- ekran: 15,6” 1920x1080
- dysk 512GB SSD SATA III

4.4. Eksperyment

Eksperyment został zaprojektowany i przeprowadzony za pomocą zaawansowanej platformy iMotions 9.0 (rys. 2). Każdemu uczestnikowi były wyświetlane instrukcje do wykonania oraz bodźce w postaci stron uczelni. Prezentowane były strony trzech serwisów, na których uczestnicy mieli do wykonania po cztery takie same zadania (tabela 2).



Rysunek 2: Projekt eksperymentu.

Tabela 2: Zadania do wykonania dla uczestników

Lp.	Treść zadania
1	Znajdź narzędzie do zmiany kontrastu strony
2	Znajdź opcję w menu/ikonę umożliwiającą dotarcie do strony biblioteki
3	Wskaż w pozycję w menu umożliwiającą dostęp do informacji dla osób niepełnosprawnych
4	Wskaż opcję, która umożliwi przejście na stronę Wydziału Informatyki/Filozofii

Przebieg sesji badawczej:

1. Poinformowanie uczestnika o celu badania oraz poinstruowanie jak ma się zachowywać podczas badań
2. Wyrażenie zgody na udział w eksperymencie
3. Wypełnienie ankiety dotyczącej danych metrycznych
4. Przeprowadzenie kalibracji
5. Nagrywanie uczestników podczas realizacji zadań
6. Wypełnienie kwestionariusza dotyczącego satysfakcji
7. Poinformowanie użytkownika o wynikach badań

Druga część badań, która następowała bezpośrednio po badaniach eyetrackingowych polegała na wypełnieniu przez uczestników ankiety składającej się z 10 pytań (tabela 3), na które musieli udzielić odpowiedzi w siedmiostopniowej skali Likerta.

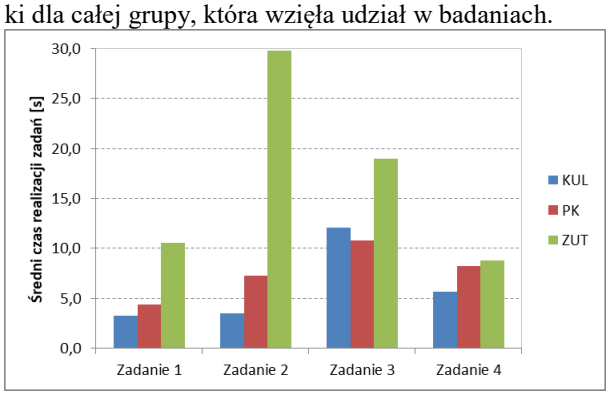
Tabela 3: Treści pytań ankietowych

Nr pytania	Treść pytania
1	Czy strona jest przejrzysta?
2	Czy zgadzasz się ze stwierdzeniem, że interfejs jest prosty w użyciu i nie ma zbędnych rzeczy które utrudniają wyszukiwanie informacji?
3	Czy uważasz, że strona jest dobrze przystosowana dla osób z niepełnosprawnościami?
4	Czy jesteśmy w stanie w łatwy sposób wyszukać informacje na stronie?
5	W jakim stopniu zgadzasz się ze stwierdzeniem, że dany serwis jest wygodny w użytkowaniu?
6	Czy czujesz, że na stronie znajduje się wiele zbędnych informacji?
7	Czy strona nie przytłacza nadmiarem informacji w jednym miejscu?
8	Czy w Twojej ocenie strona jest atrakcyjna?
9	Czy zgadzasz się ze stwierdzeniem, że menu główne strony jest przejrzyste i wygodne w użyciu?
10	Czy przeznaczenie elementów i przycisków strony jest odpowiednio opisane oraz odpowiada faktycznym informacjom, które niosą ze sobą odnośniki?

5. Wyniki badań

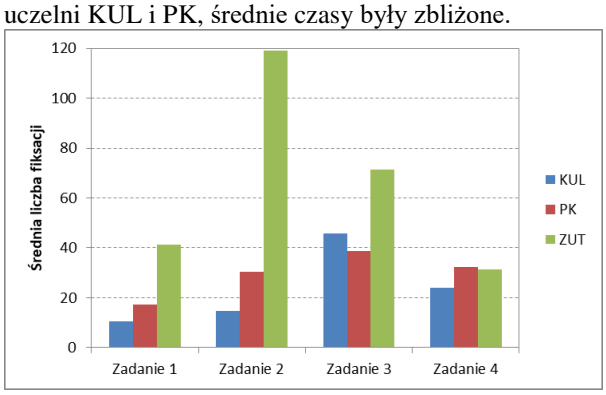
5.1. Wyniki badań eyetrackingowych – analiza ilościowa

Analiza danych eyetrackingowych została przeprowadzona przez dwóch ekspertów, a następnie ich wyniki zostały uśrednione. Analizę wykonywano poklatkowo na materiale wideo z wszystkich sesji badawczych. Dokonano pomiaru czasu realizacji zadań (czas do pierwszej fiksacji na wyszukiwanym obiekcie), liczby fiksacji do momentu odnalezienia celu oraz określono liczbę nieprawidłowo wykonanych zadań. Na rysunku 3, 4 i 5 znajdują się wykresy prezentujące średnie wyniki dla całej grupy, która wzięła udział w badaniach.



Rysunek 3: Średnie czasy realizacji zadań.

Czas do pierwszej fiksacji na obiekcie zainteresowania lub czas realizacji zadania jest miarą eyetrackingową, która określa m.in. poziom trudności odnalezienia danego elementu. W przypadku gdy czas ten jest krótki, to może oznaczać, że poziom trudności odnalezienia obiektu jest niski. Natomiast jeżeli czas odnalezienia elementu jest długi, wówczas poziom trudności jest wysoki. Wykres z rysunku 3 pokazuje, że we wszystkich zadaniach respondenci mieli problem z obsługą serwisu ZUT. Wskazują na to średnie czasy przy realizacji wszystkich zadań. W przypadku stron uczelni KUL i PK, średnie czasy były zbliżone.



Rysunek 4: Średnie liczby fiksacji do momentu znalezienia celu.

Kolejną miarą zastosowaną w analizie ilościowej jest liczba fiksacji. Miara ta może być w różny sposób interpretowana. Na przykład duża liczba fiksacji, może wynikać z tego, że dany obszar jest bardziej skomplikowany i wymaga od użytkownika bardziej wytężonego myślenia [11, 12]. Rysunek 4 pokazuje średnie liczby fiksacji podczas realizacji poszczególnych zadań dla trzech serwisów uczelni. Dla trzech pierwszych zadań wyraźnie większą średnią liczbę fiksacji zanotowano w serwisie ZUT. W zadaniu 4 rezultaty się wyrównały. Generalnie najmniej fiksacji występowało podczas eksploracji serwisu KUL (oprócz zadania nr 3).

Rysunek 5: Liczba nieprawidłowo wykonanych zadań.

Po przeprowadzeniu eksperymentu, nagrania były analizowane przez dwóch ekspertów. Oprócz pomiarów czasu znalezienia obiektów oraz zliczenia fiksacji, eksperci oceniali czy dane zadanie zostało prawidłowo wykonane tzn. czy cel został osiągnięty przez respondenta. Z wykresu wynika, że oceny ekspertów na ogół się pokrywały. Także w tym przypadku widać, że uczestnicy badań najwięcej problemów mieli z realizacją zadań w serwisie ZUT, chociaż liczba nieprawidłowych realizacji zadań nie była duża i wynosiła w granicach od 1 do 3.

5.2. Wyniki badań eyetrackingowych – analiza jakościowa

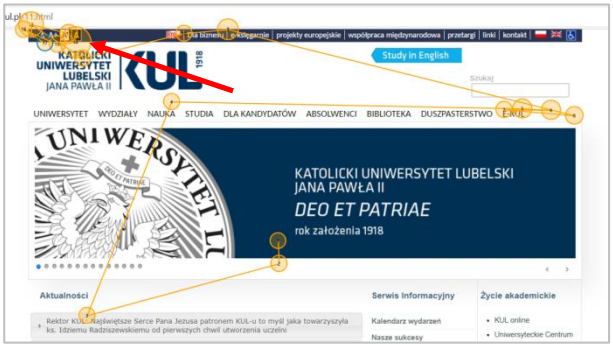
Analiza jakościowa oparta była na mapach ciepła i ścieżkach skanowania wybranych przykładów poprawnie i niepoprawnie zrealizowanych scenariuszy badawczych (zadań) przez osoby badane. Obszary oznaczone kolorem czerwonym, żółtym i zielonym pokazują gdzie kierowana była uwaga respondentów. Kolor czerwony świadczy o dużej intensywności uwagi w danym miejscu wszystkich uczestników badań. Poniżej znajdują się trzy mapy (rys. 6, 7, 8), które obrazują gdzie znajdowały się cele wyszukiwane przez badanych podczas realizacji zadania nr 1 czyli szukania ikony do zmiany kontrastu strony.

Rysunek 6: Wynik realizacji zadania 1 w postaci mapy ciepła dla serwisu uczelni ZUT.

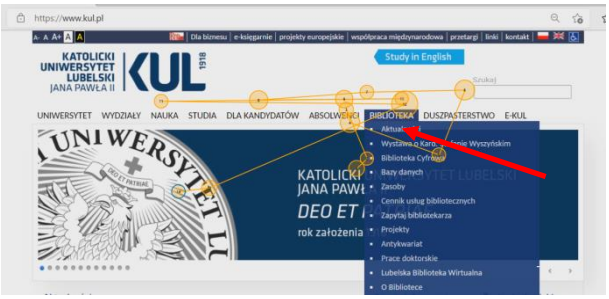
Rysunek 7: Wynik realizacji zadania 1 w postaci mapy ciepła dla serwisu uczelni PK.

Rysunek 8: Wynik realizacji zadania 1 w postaci mapy ciepła dla serwisu uczelni KUL.

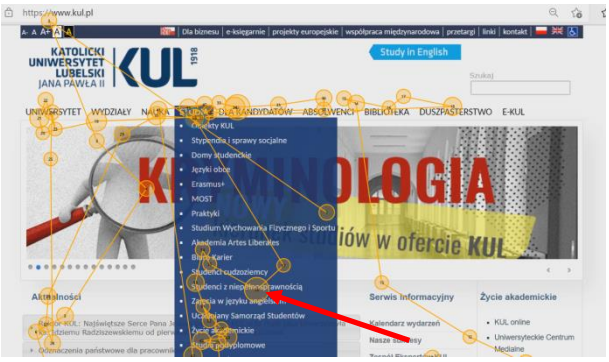
Poniżej zostały przedstawione cztery przykładowe poprawne realizacje zadania 1, 2, 3 i 4 dla serwisu uczelni KUL z wskazanymi strzałką, docelowymi elementami wyszukiwanymi przez respondentów (rys. 9, 10, 11 i 12).



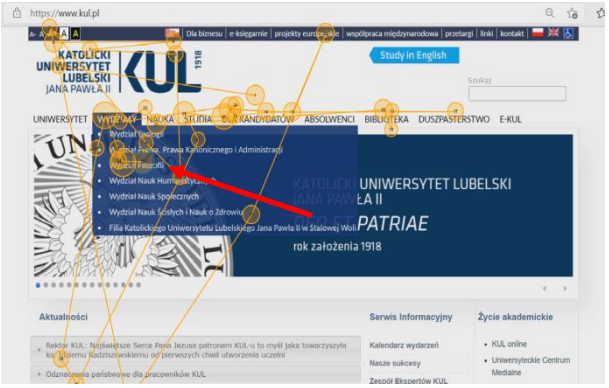
Rysunek 9: Ścieżka skanowania jednego z uczestników badań realizującego zadanie 1 (uczelnia KUL, ikona zmiany kontrastu).



Rysunek 10: Ścieżka skanowania jednego z uczestników badań realizującego zadanie 2 (KUL, opcja w menu Biblioteka).

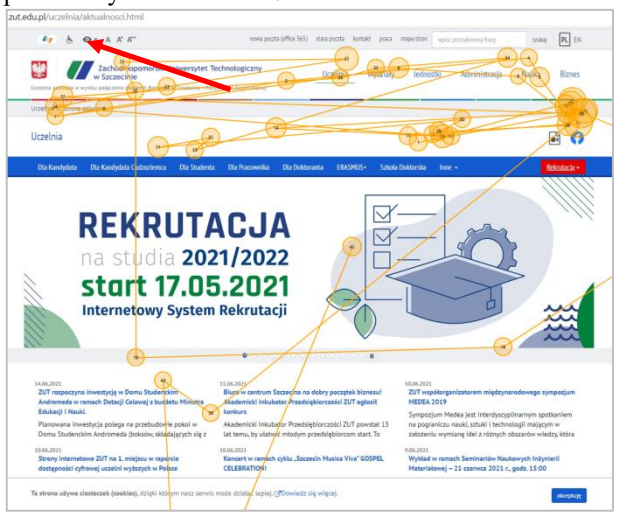


Rysunek 11: Ścieżka skanowania jednego z uczestników badań realizującego zadanie 3 (KUL, opcja w menu z informacjami dla osób niepełnosprawnych).

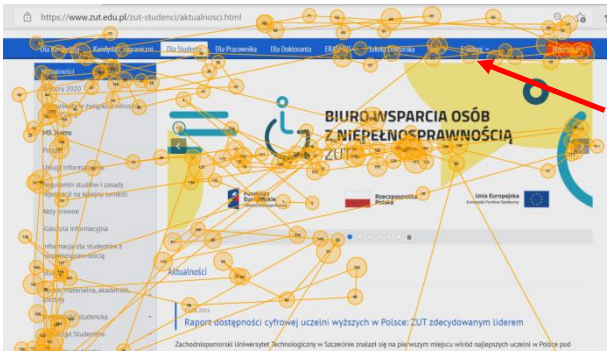


Rysunek 12: Ścieżka skanowania jednego z uczestników badań realizującego zadanie 4 (KUL, opcja w menu Wydział Informatyki).

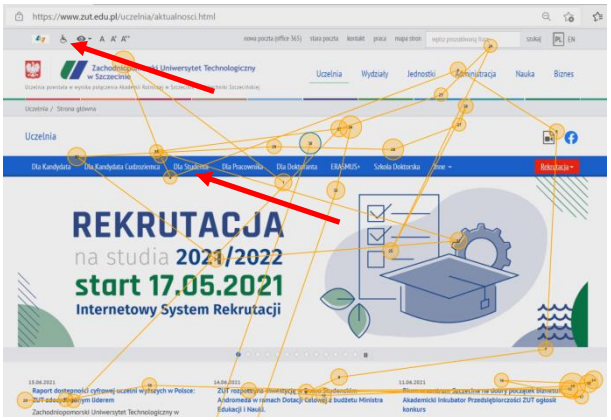
Kolejne cztery rysunki 13, 14, 15 i 16 przedstawiają ścieżki skanowania na tle bodźców – stron www dla dwóch uczelni (PK i ZUT) ukazujące z kolei niepoprawne wykonanie zadań.



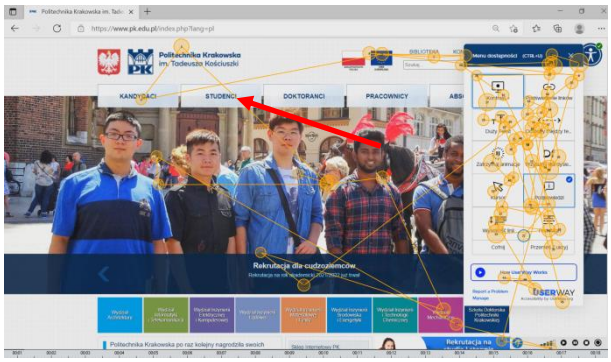
Rysunek 13: Efekt realizacji zadania 1 – narzędzie zmiany kontrastu nie zostało odnalezione (strona ZUT).



Rysunek 14: Efekt realizacji zadania 2 – opcja dostępu do biblioteki nie została odnaleziona (widoczne intensywne przeszukiwanie całego obszaru strony trwające 42 sekundy).



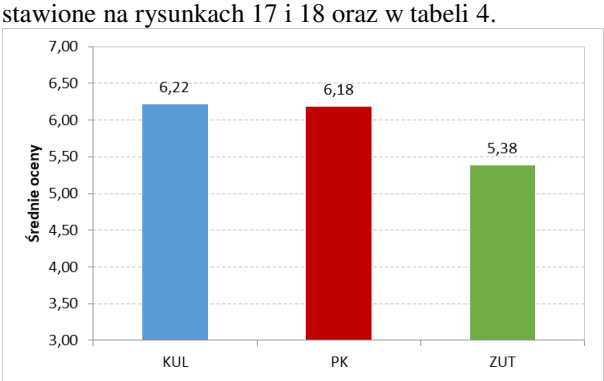
Rysunek 15: Efekt realizacji zadania 3 – ikona dostępu do informacji dla osób niepełnosprawnych nie została znaleziona.



Rysunek 16: Efekt realizacji zadania 3 – ikona dostępu do informacji dla osób niepełnosprawnych nie została znaleziona (osobę badaną zmyliło narzędzie dostępności).

5.3. Wyniki badań ankietowych

Wyniki z badań kwestionariuszowych zostały przedstawione na rysunkach 17 i 18 oraz w tabeli 4.



Rysunek 17: Średnie oceny z wszystkich pytań dla analizowanych serwisów uczelni.

Rysunek 18: Ocena serwisów uczelni pod względem ich przystosowania dla osób z niepełnosprawnościami.

Tabela 4: Średnie wyniki ocen na pytania z ankiety

Nr zadania	KUL	PK	ZUT
1	6,67	6,67	5,50
2	6,00	5,83	5,17
3	6,00	6,33	3,50
4	6,50	7,00	6,33
5	6,17	6,00	6,00
6	3,83	5,33	6,00
7	6,83	5,33	4,67
8	5,33	5,50	5,00
9	7,50	6,67	5,67
10	7,67	7,17	6,17

Wykres z rysunku 17 przedstawia uśrednione oceny poszczególnych serwisów wystawione przez uczestników badań na podstawie pytań wchodzących w skład przygotowanej ankiety. Najwyższą średnią ocenę otrzymał serwis KUL (6,22), a najniższą witryna ZUT (5,38). Ważnym elementem ankiety było pytanie dotyczące przystosowania serwisów do potrzeb osób niepełnosprawnych. Odpowiedzią na to pytanie są wyniki zwizualizowane na rysunku 18, które pokazują, że serwis PK osiągnął najwyższą średnią ocenę (6,33), niewiele niższą serwis KUL (6,00) a najniższą strony ZUT (3,50). Średnie wyniki na pozostałe pytania zostały przedstawione w tabeli 4. Czcionką koloru czerwonego zostały oznaczone rezultaty najlepsze, zaś czcionką zieloną wyniki najgorsze. Dla pięciu pytań (2, 5, 6, 9, 10), które dotyczyły takich aspektów jak prostota stron, wygoda użytkowania, zawartość zbędnych informacji, przejrzystość i wygoda menu oraz opis narzędzi zgodny z przeznaczeniem - najwyższe oceny uzyskał serwis KUL. W przypadku pytania nr 1, dotyczącego przejrzystości stron, takie same średnie oceny uzyskały witryny uczelni KUL oraz PK. Natomiast najlepsze oceny dla pytania trzeciego i czwartego osiągnął serwis PK, a dla pytania siódmego serwis ZUT. Pytania te dotyczyły odpowiednio kwestii wyszukiwania informacji, wygody korzystania z serwisu oraz nadmiarowości informacji.

6. Wnioski

W ramach pracy została zaproponowana porównawcza metoda badania serwisów internetowych opierająca się na badaniach eyetrackingowych oraz kwestionariuszowych. Badania skupiały się na pewnych elementach użyteczności, ze zwróceniem uwagi na narzędzia dla osób niepełnosprawnych. Zarówno w części eyetrackingowej oraz ankietowej badań brało udział 10 osób. Każda osoba miała do wykonania po cztery zadania na trzech serwisach uczelni. W sumie dało to 120 nagrań wideo aktywności ocznej. Analiza tych rejestracji, przeprowadzona przez dwie niezależne osoby, była więc dla nich dużym wyzwaniem. Pomimo tego na uzyskane wyniki trzeba patrzeć z pewnym dystansem ze względu na fakt, że w badaniach uczestniczyła niewielka grupa respondentów, a zestaw zadań do wykonania był ograniczony do czterech zadań, podczas realizacji których uczestnicy wchodzili w interakcję wyłącznie ze stronami głównymi serwisów.

Wyniki dwóch ilościowych analiz, zarówno eyetrackingowej jak i ankietowej, pokazują, że najlepsze wyniki w tych badaniach uzyskał serwis uczelni KUL, nieco gorsze PK, a najsłabsze ZUT. W zakresie analizy jakościowej zebrano szeroki materiał badawczy w postaci map cieplnych i ścieżek skanowania. Pierwszy sposób wizualizacji wyników pokazywał ogólne tendencje grupy podczas realizacji zadań. Drugi sposób wizualizacji drogi po jakiej poruszał się punkt skupienia uwagi uczestników badań, pokazywał indywidualne sposoby dotarcia do celu. Ścieżki skanowania naniesione na wyświetlany bodziec, ukazywały to, co respondentom przeszkadzało w realizacji zadania. Najczęstsze przypadki błędnego wyszukiwania były wynikiem niezro-

zumienia treści zadania, zbyt małej wielkości szukanego obiektu lub zbyt dużego nagromadzenia obiektów w danym obszarze strony. Badania zawarte w tej pracy nie uwzględniały bardziej pogłębionych analiz, które w przyszłości mogą zostać przeprowadzone na podstawie uzyskanych wyników. Badania rozpoczęte w tej pracy można kontynuować na podstawie analizy kierunków przeszukiwania oraz pomiarów odległości pomiędzy kolejnymi fiksacjami.

Literatura

[1] M. Nilsson, UX method development from Usability testing with Eye tracking for E-commerce, http://urn.kb.se/resolve?urn=urn:nbn:se:mau:diva-21842, [14.08.2021].

[2] M. Miłosz, Ergonomia systemów informatycznych, Politechnika Lubelska, 2014.

[3] M. Macleod, R. Bowden, N. Vevan, I. Curson, The MUSIC Performance Measurement Method, Behaviour and Information Technology 16 (1997) 279-293.

[4] J. H. Goldberg, A. M. Wichansky, Eye tracking in usability evaluation: A practitioner's guide, [w] J. Hyönä, R. Radach, & H. Deubel (Ed.), The mind's eye: Cognitive and applied aspects of eye movement research, Amsterdam: Elsevier (2003) 493-516.

[5] A. Bojko, Eye tracking the user experience: A practical guide to research, Brooklyn, NY: Rosenfeld Media, 2013.

[6] F. Onorati, R. Barbieri, M. Mauri, V. Russo, L. Mainardi, Characterization of affective states by pupillary dynamics and autonomic correlates, https://www.frontiersin.org/articles/10.3389/fneng.2013.00009/full, [14.08.2021].

[7] J. A. Jacko, A. B. Barreto, J. Y. M., Chu, H. S. Bautsch, G. J. Marmet, I. U. Scott, R. H. Rosa, Low vision: The role of visual acuity in the efficiency of cursor movement, Proceedings of ACM Conference on Assistive Technologies (2000) 1-8.

[8] J. H. Goldberg, M. J. Stimson, M. Lewenstein, N. Scott, A. M. Wichansky, Eye Tracking in Web Search Tasks: Design Implications. Proceedings of Eye tracking research & applications (2002) 51-58.

[9] Gazepoint, https://www.gazept.com/product/gp3hd/, [14.08.2021].

[10] iMotions, https://imotions.com/, [14.08.2021].

[11] X. P. Kotval, J. H. Goldberg, Eye Movements and Interface Components Grouping: An Evaluation Method, Proceedings of the 42nd Annual Meeting of the Human Factors and Ergonomics Society (HFES) 42 (1998) 486-490.

[12] A. Bojko, Using Eye Tracking to Compare Web Page Designs: A Case Study, Journal of Usability Studies 3(1) (2006) 112-120.

[13] T. Tullis, B. Albert, Measuring the user experience: collecting, analyzing, and presenting usability metrics, Morgan Kaufmann Publishers, Elsevier, 2008.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 303–308
Received: 09 September 2021
Accepted: 19 September 2021

A comparison of conventional and deep learning methods of image classification

Porównanie metod klasycznego i głębokiego uczenia maszynowego w klasyfikacji obrazów

Maryna Dovbnych*, Małgorzata Plechawska–Wójcik

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20–618 Lublin, Poland

Abstract

The aim of the research is to compare traditional and deep learning methods in image classification tasks. The conducted research experiment covers the analysis of five different models of neural networks: two models of multi–layer perceptron architecture: MLP with two hidden layers, MLP with three hidden layers; and three models of convolutional architecture: the three VGG blocks model, AlexNet and GoogLeNet. The models were tested on two different datasets: CIFAR–10 and MNIST and have been applied to the task of image classification. They were tested for classification performance, training speed, and the effect of the complexity of the dataset on the training outcome.

Keywords: image classification; machine learning; deep learning; neural networks

Streszczenie

Celem badań jest porównanie metod klasycznego i głębokiego uczenia w zadaniach klasyfikacji obrazów. Przeprowadzony eksperyment badawczy obejmuje analizę pięciu różnych modeli sieci neuronowych: dwóch modeli wielowarstwowej architektury perceptronowej: MLP z dwiema warstwami ukrytymi, MLP z trzema warstwami ukrytymi; oraz trzy modele architektury konwolucyjnej: model z trzema VGG blokami, AlexNet i GoogLeNet. Modele przetrenowano na dwóch różnych zbiorach danych: CIFAR–10 i MNIST i zastosowano w zadaniu klasyfikacji obrazów. Zostały one zbadane pod kątem wydajności klasyfikacji, szybkości trenowania i wpływu złożoności zbioru danych na wynik trenowania.

Słowa kluczowe: klasyfikacja obrazów; uczenie maszynowe; uczenie głębokie; sieci neuronowe

*Corresponding author

Email address: maryna.dovbnych@pollub.edu.pl (M. Dovbnych)

1. Introduction

Nowadays, image classification methods play an important role in a wide variety of areas of life. Image classification is the process of extracting classes of information from a multiband bitmap, in other words, the problem of image classification is receiving an initial image and determine its class (cat, dog, etc.) or a group of probable classes that best characterizing the image. This paper presents a comparison of conventional and deep learning methods of image classification.

Multilayer Perceptron (MLP) is the most popular type of artificial neural networks. It is a class of feed-forward artificial neural network. This type of network typically consists of one input layer, several hidden layers and one output layer. Each node in MLP is a neuron with a nonlinear activation function (except of input nodes). Although this type of network ignores the spatial information of the image, a lightweight MLP with 2–3 layers can easily cope with simple data sets like MNIST [1]. The MNIST is a voluminous database of handwritten numeral samples [2]. In the paper [3] MLP network with a single hidden layer was able to reach 43.4% of accuracy. The multilayer perceptron based architecture was once commonly used for computer vision, and is now increasingly being replaced by the Convolutional Neural Network (CNN) [3, 4] and other machine learning methods. For example, the paper [5] compares MLP with other machine learning methods such as decision tree, logistic regression and support vector machine for solving image classification problems.

Artificial networks based on CNN architecture are considered to be universal [6], because they are used for a wide range of tasks, from botany [7, 8, 9, 10] and geography [11] to medical diagnostics [12, 13, 14, 15]. CNN–based models take into account the dimensional information of an image, which gives this type of architecture an advantage over networks with an architecture like MLP for image classification tasks. Another difference between MLP and CNN architectures is that layers in CNN not fully connected like in MLP. Convolutional neural network through the use of a special convolution operation allows to simultaneously reduce the amount of information stored in memory, due to which it copes better with higher–resolution pictures, and to highlight the reference features of the image, such as edges, contours or edges. At the next level of processing, from these edges and faces, you can recognize repeatable fragments of textures, which can then fold into fragments of the image. There are many types of convolutional neural network architectures and their modifications that have been developed to make the trained

model perform better [16, 13]. In the paper [17] proposed methods of the automatic designing CNN architectures using the Genetic image classification algorithm. Not only architectures are being modified, but also ways of solving problems. For example, the paper [18] shows how image segmentation techniques have evolved.

The process of learning a machine itself consists in preparing the appropriate data containing the necessary rules and a description of the object's properties, as well as selecting the optimal parameters for the model which is trained. These factors increase the impact of the selected training data set on training efficiency [19, 10]. The data set is usually divided into several parts: training data, which is used to train the model, validation data, which is used by machine learning engineers during the design phase to tune the hyperparameters of the model, and test data is used to evaluate performance of the already trained model. Sometimes, validation data is used as test data. The number of images in the data set, their size, as well as the number of images in each of the categories by which we will classify them affect the training efficiency. As mentioned above, if the model is be trained successfully, it must be well parametrized and optimized. In the paper [20], the optimization problems faced by a machine learning specialist are described. According to the paper [21] choosing the correct activation function also plays a critical role in model training. The wrong selection of parameters can lead to overfitting or underfitting [22].

Underfitting is a situation when in a parametric family of functions it is not possible to find a function that describes the data well. The most common reason for underfitting is when the complexity of the data structure is higher than the complexity of the model that the researcher came up with. The solution to this problem is to complicate the model and find a better description of the effects that are in the data.

Overfitting is the opposite of underfitting when the model is too complex and universal. The error probability of the trained algorithm on the objects of the test sample turns out to be significantly higher than the average error on the training sample. There are techniques to avoid overfitting the model. For example, increasing the size of the training sample can help, if collecting more data is not possible, then various transformations (rotation, reflection, scaling, etc.) can be performed on an already existing set of images. Techniques such as cross validation, L1/L2 regularization also can help to avoid the problem of overfitting. One of the most effective techniques to prevent the appearance of the overfitting effect is to add dropout layers to the neural network architecture. By using dropout layers model ignore a subset of our network units with a given probability and reduce interdependent learning among units that could lead to overfitting. However, using dropout layers, it will take more epochs for our model to converge.

To predict how the trained model will behave in practice, the performance of the model is evaluated. Different performance metrics are used to evaluate the performance of different algorithms. Metrics such as Confusion Matrix, Accuracy, Precision, Recall, Specificity and F1 Score are commonly used for classification tasks. All of the above metrics use number of true positives, true negatives, false positive and false negative predictions. A true positive is when the model correctly predicted a positive class, and a true negative is when the model correctly predicts a negative class. False positive and false negative, respectively, are cases where the model incorrectly predicted a positive or negative class. Correctly selected metrics are the key to an accurate assessment of model performance.

To carry out this research work, a machine learning framework or library is needed. To solve the problems of image classification in this work, an open source Tensorflow library from Google was chosen. Tensorflow offers many out–of–the–box solutions that make learning model faster and easier. The API of Tensorflow library layer provides a simpler interface to commonly used layers in deep learning models. An example of the classification performance and qualitative analysis using the Tensorflow library can be seen in the paper [23]. A systematic overview of using TensorFlow for image classification can be found in the paper [24].

In this work, it is conducted an experiment that relies on classification performance and qualitative analysis of conventional and deep learning methods of image classification. The thesis of this study is “CNN obtains better performance in the task of image classification than MLP”. Detailed research hypotheses are:

1. CNN based architecture give better accuracy than MLP;
2. models with MLP architecture give lower classification accuracy than CNN–based models when classifying color images;
3. CNN type networks train faster than MLP.

2. Research implementation

The research covered two tests. In the first one, it is checked whether CNN–type architectures give a higher classification accuracy than a multilayer perceptron, and also whether the choice of a black–and–white dataset affects the classification accuracy in the case of using a multilayer perceptron. The second test examines and compares the training speed for the neural network architectures studied in this article.

All tests were carried out on an MSI GL63 8SC laptop with the following specifications:

- CPU: Intel Core i7–8750H;
- CPU Clock Rate: 2.2 GHz / 4.1 GHz;
- GPU: NVIDIA GeForce GTX 1650 GDDR5;
- GPU memory: 4 GB;
- RAM: 16 GB.

Two datasets were chosen for training and evaluating the models: MNIST Database – volume set (60000 train and 10000 test images) of black and white handwritten numbers samples from 0 to 9 (ten classes) size of 28x28 and CIFAR–10 data set [25] consists of color

images in 10 classes size of 32x32. There are 50000 training images and 10000 test images.

The evaluation performance of the model is carried out on the basis of the Accuracy and F1 score metrics, as well as the value of the loss function. Accuracy is way to measure how often the algorithm classifies a data correctly. Accuracy is the number of correctly predicted data points out of all the data points. F1 score is a metric for determining how accurate a test is. It is calculated using the test's precision and recall, with precision equaling the number of true positive results divided by the total number of positive results, including those that were incorrectly identified, and recall equaling the number of true positive results divided by the total number of samples that should have been identified as positive. The F1 score is calculated by taking the harmonic mean of precision and recall.

For the experiment, two models of the MLP type and three convolutional models were chosen: MLP with two hidden layers, MLP with 3 hidden layers, Three VGG blocks model, AlexNet and GoogLeNet.

The MLP is a feedforward neural network having a source neuron input layer, at least one hidden layer (two and three hidden layers in these cases) of computational neurons, and a computational neuron output layer. The input layer receives signals from the environment and redistributes them to all neurons in the hidden layer.

The basic idea behind VGG architectures is to use more layers with smaller filters. There are VGG–16 and VGG–19 versions with 16 and 19 layers respectively. In this experiment, a model with three VGG layers is implemented.

The AlexNet architecture consists of five convolutional layers, between which pooling layers and normalization layers are located, and three fully connected layers complete the neural network.

The GoogLeNet is a deep architecture with 22 layers. The goal was to develop a neural network with the highest computational efficiency. To do this, Google came up with the so–called Inception module — the entire architecture consists of many such modules, following one after another. The idea behind the main Inception module is that it is itself a small local area network. All his work consists in the parallel application of several filters to the original image. The filter data is combined to create an output that goes to the next layer.

2.1. Test 1 implementation

Each model was trained in a loop 20 times to maximum accuracy for MNIST and CIFAR–10 data sets, and the training results (Accuracy, Loss, F1 score) were recorded in a csv format file for further analysis. Chart of accuracy and loss and confusion matrix chart were saved on disk. SGD optimizers and data generators were used for all models. At the beginning of each loop step, a random seeds were set, the training data set was randomly splitted into training (80%) and validation (20%) subsets, a new instance of SGD optimizer, data generator and model were created and compiled. After training, the model was saved to disk, and then its object was deleted, and the session was cleaned.

2.2. Test 2 implementation

All models were trained with the same settings as in section 2.1. 10 times up to 20 epochs on MNIST data set. Accuracy and loss of first, fifth, tenth, fifteenth and twentieth epochs, as well as accuracy, loss and F1 score for test data set were recorded in the csv file at each loop step.

3. Results of first test

During the analysis of the results, the mean value of the accuracy, loss function and F1 score were calculated for each model. Mean loss for each model without division into a data set are presented in the figure 1.

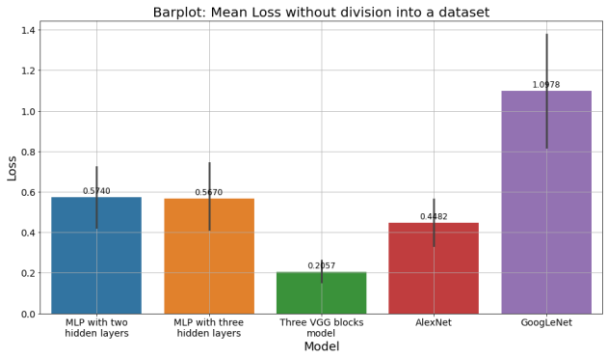


Figure 1: Mean loss for each model without division into a data set

Mean accuracy for each model without division into a data set are presented in the figure 2.

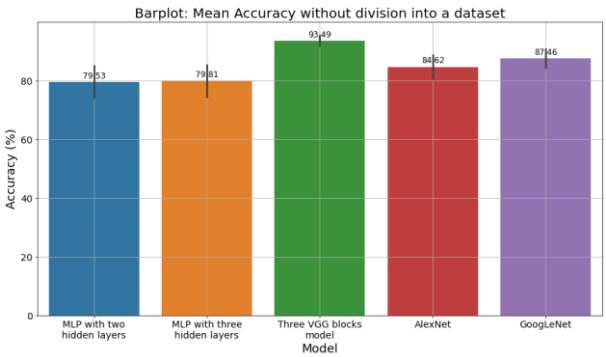


Figure 2: Mean accuracy for each model without division into a data set

Mean F1–score value for each model without division into a data set are presented in the figure 3.

Mean loss for each model on MNIST data set are shown in the figure 4.

Mean accuracy for each model on MNIST data set are presented in the figure 5.

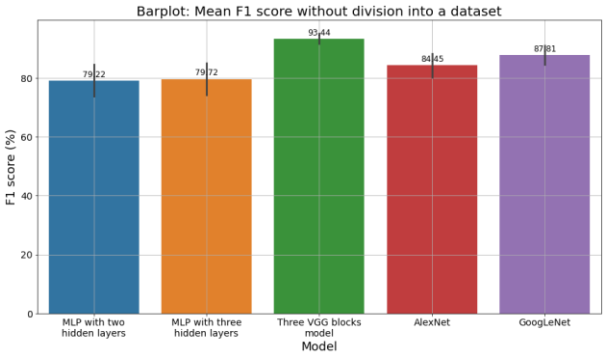


Figure 3: Mean F1–score for each model without division into a data set

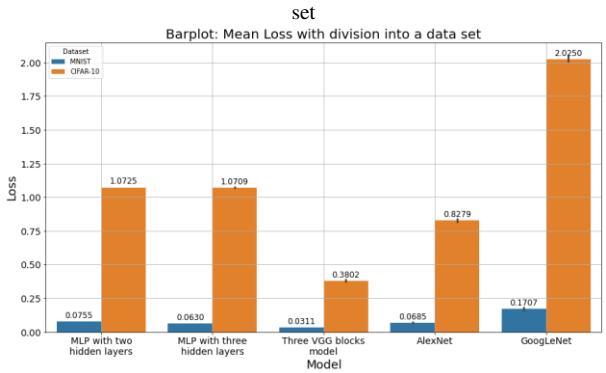


Figure 4: Mean loss for each model with division into a data set

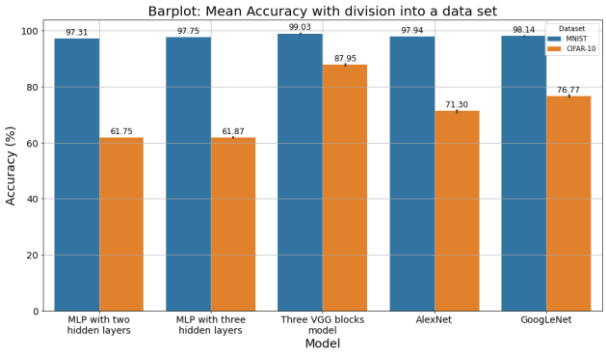


Figure 5: Mean accuracy for each model with division into a data set

Mean F1–score for each model on MNIST data set are shown in the figure 6.

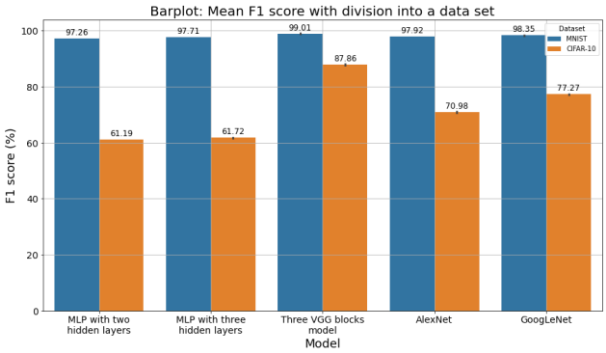


Figure 6: Mean F1–score for each model with division into a data set

As can be seen from the results the MLP–type architectures presented in this paper do a good job with a simple black–and–white MNIST dataset, but their classification accuracy and F1–score for the color CIFAR–10 dataset is much lower than that of the CNN–type architectures. However, the loss function value for both MLP architectures is lower than the value for GoogLeNet, but higher than for another two CNN–type architectures presented in this paper (Three VGG blocks and AlexNet). The three VGG blocks architecture showed the best results in terms of accuracy, loss function and F1 score. AlexNet architecture shows third best results in terms of accuracy and F1 score and second best result in term of loss function.

4. Results of second test

During the analysis of the results, the mean value of the final accuracy, loss function and F1–score were calculated. Also the mean value of the loss function, validation loss function, accuracy and validation accuracy for first, fifth, tenth, fifteenth and twentieth epoch were calculated. Figure 7 shows the change of the classification accuracy for each model during training.

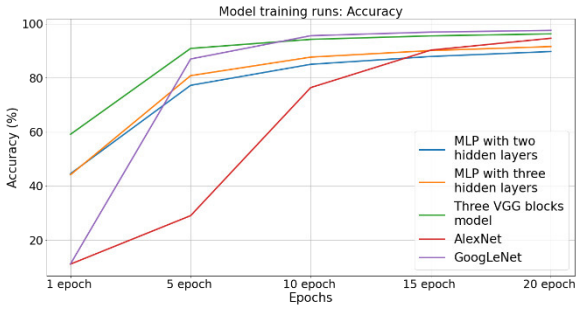


Figure 7: Change of the classification accuracy for each model

Figure 8 shows the change of the validation accuracy values for each model.

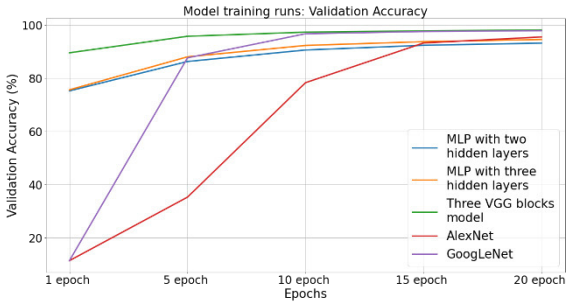


Figure 8: Change of the validation accuracy for each model

Figure 9 shows the change of the loss function values for each model during training.

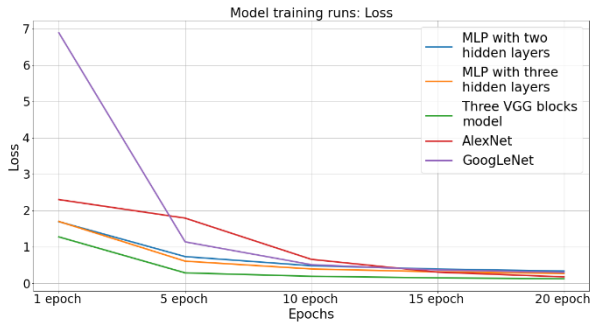


Figure 9: Change of the loss function values for each model

Figure 10 presents the change of the validation loss function values for each model.

Figure 10: Change of the validation loss function values for each model

The results show that initially AlexNet trains the slowest of all and, in terms of classification accuracy, catches up with MLP–type models only closer to the fifteenth epoch. GoogLeNet in the first epoch has worse results than MPL–type models, but quickly overtakes them after the fifth epoch. The three VGG blocks model has the best training speed and retains it throughout each epoch.

5. Conclusions

The aim of the study was to compare the performance of convolutional and traditional neural network architectures. During the research for this thesis, familiarization with machine learning and deep learning issues was required. Convolutional networks are the most widely used neural networks used for image classification tasks, and it was concluded during this study that CNN–type networks are the best choice for this purpose due to the accuracy of the classification and the smaller loss function, than MLP–type architectures. In terms of training speed, three VGG blocks network showed the best results while AlexNet showed the worst. These results are due to the filter size influencing the training speed. As mentioned in section 2, the VGG models use small filters and that's why three block VGG model have best results in training speed test. For the same reason, it cannot be argued that convolutional neural networks train faster than MLP–type networks. It also was confirmed that MLP networks give lower classification accuracy than CNN–based models when classifying color images, but give higher accuracy for simple black and white images comparing to them classifying color images. The obtained results partially confirm the main thesis put on beginning of work.

References

[1] MNIST handwritten digit database, http://yann.lecun.com/exdb/mnist [13.02.2021]

[2] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient–based learning applied to document recognition, Proceedings of the IEEE 86(11) (1998) 2278–2324.

[3] S. B. Driss, M. Soua, R. Kachouri, M. Akil, A comparison study between MLP and convolutional neural network models for character recognition, in SPIE Conference on RealTime Image and Video Processing, Anaheim, United States, 10–11 April (2017) 1022306.

[4] N. Sharma, V. Jain, A. Mishra, An analysis of convolutional neural networks for image classification, Procedia computer science 132 (2018) 377–384.

[5] J. M. Peña, P. A. Gutiérrez, C. Hervás–Martínez, J. Six, R. E. Plant, F. López–Granados, Object–based image classification of summer crops with machine learning methods, Remote Sensing 6(6) (2014) 5019–5041.

[6] D. X. Zhou, Universality of deep convolutional neural networks, Applied and computational harmonic analysis 48(2) (2020) 787–794.

[7] I. M. Dheir, A. S. A. Mettleq, A. A. Elsharif, S. S. Abu–Naser, Classifying Nuts Types Using Convolutional Neural Network, International Journal of Academic Information Systems Research 3(12) (2020) 12–18.

[8] Y. Li, J. Nie, X. Chao, Do we really need deep CNN for plant diseases identification?, Computers and Electronics in Agriculture 178 (2020) 105803.

[9] P. Sharma, Y. P. S. Berwal, W. Ghai, Performance analysis of deep learning CNN models for disease detection in plants using image segmentation, Information Processing in Agriculture 7(4) (2019) 566–574.

[10] J. G. A. Barbedo, Impact of data set size and variety on the effectiveness of deep learning and transfer learning for plant disease classification, Computers and electronics in agriculture 153 (2018) 46–53.

[11] P. T. T. Ngo, M. Panahi, K. Khosravi, O. Ghorbanzadeh, N. Karimnejad, A. Cerda, S. Lee, Evaluation of deep learning algorithms for national scale landslide susceptibility mapping of Iran, Geoscience Frontiers 12(2) (2020) 505–519.

[12] I. Banerjee, Y. Ling, M. C. Chen, S. A. Hasan, C. P. Langlotz, N. Moradzadeh, B. Chapman, T. Amrhein, D. Mong, D. L. Rubin, O. Farri, M. P. Lungren, Comparative effectiveness of convolutional neural network (CNN) and recurrent neural network (RNN) architectures for radiology text report classification, Artificial intelligence in medicine 97 (2019) 79–88.

[13] C. L. Chowdhary, M. Mittal, P. A. Pattanaik, Z. Marszalek, An efficient segmentation and classification system in medical images using intuitionist possibilistic fuzzy C–mean clustering and fuzzy SVM algorithm, Sensors 20(14) (2020) 3903.

[14] T. Nakaura, T. Higaki, K. Awai, O. Ikeda, Y. Yamashita, A primer for understanding radiology articles about machine learning and deep learning, Diagnostic and Interventional Imaging 101(12) (2020) 763–844.

[15] X. Yang, Y. Ye, X. Li, R. Y. Lau, X. Zhang, X. Huang, Hyperspectral image classification with deep learning models., IEEE Transactions on Geoscience and Remote Sensing 56(9) (2018) 5408–5423.

[16] A. F. Agarap, An architecture combining convolutional neural network (CNN) and support vector machine (SVM) for image classification, arXiv:1712.03541v2 (2019).

[17] Y. Sun, B. Xue, M. Zhang, G. G. Yen, J. Lv, Automatically Designing CNN Architectures Using the

Genetic Algorithm for Image Classification, IEEE Transactions on Cybernetics 50(9) (2020) 3840–3854.

[18] F. Sultana, A. Sufian, P. Dutta, Evolution of image segmentation using deep convolutional neural network: A survey, Knowledge–Based Systems 201–202 (2020) 106062.

[19] O. Sbai, C. Couprie, M. Aubry, Impact of base data set design on few–shot image classification, in Computer Vision–ECCV 2020: 16th European Conference, Glasgow, United Kingdom, August 23–28 (2020) 597–613.

[20] C. Gambella, B. Ghaddar, J. Naoum–Sawaya, Optimization problems for machine learning: a survey, European Journal of Operational Research 290(3) (2020) 807–828.

[21] Y. Wang, Y. Li, Y. Song, X. Rong, The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition, Applied Sciences 10(5) (2020) 1897.

[22] D. Bashir, G. D. Montanez, S. Sehra, P. S. Segura, J. Lauw, An Information–Theoretic Perspective on Overfitting and Underfitting, in AI 2020: Advances in Artificial Intelligence: 33rd Australasian Joint Conference, Canberra, Australia, November 29–30 (2020) 347–358.

[23] T. Kiran, Computer Vision Accuracy Analysis with Deep Learning Model Using TensorFlow, International Journal of Innovative Research in Computer Science & Technology (IJIRCST) 8(4) (2020) 2347–5552.

[24] T. P. P. Padilha, L. E. A. de Lucena, A Systematic Review About Use of TensorFlow for Image Classification and Word Embedding in the Brazilian Context, Academic Journal on Computing, Engineering and Applied Mathematics 1(2) (2020) 24–27.

[25] The CIFAR–10 dataset, https://www.cs.toronto.edu/~kriz/cifar.html [13.02.2021]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 309–315
Received: 07 September 2021
Accepted: 26 September 2021

Comparative analysis of connection performance with databases via JDBC interface and ORM programming frameworks

Analiza porównawcza wydajności połączeń z bazami danych poprzez interfejs JDBC i szkielety programistyczne ORM

Mateusz Żuchnik*, Piotr Kopniak

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The research topic of this paper was a comparative analysis of the performance of database connections using different communication methods based on the Java programming language. The investigated tools mediating communication with databases included JDBC drivers and Object-Relational Mapping (ORM) programming frameworks. The research was conducted based on 8 different criteria, in order to select the most effective method and tool for working with relational databases, when developing applications in Java. The different criteria were given weights, which were determined through a survey of Java developers and computer science students. Hibernate turned out to be the best tool without taking into account the weights obtained, and with taking into account the weights the JDBC tool.

Keywords: database connections; Java; performance; ORM framework

Streszczenie

Tematem badań niniejszego artykułu była analiza porównawcza wydajności połączeń z bazami danych za pomocą różnych metod komunikacji w oparciu o język programistyczny Java. W skład badanych narzędzi pośredniczących w komunikacji z bazami danych weszły: sterowniki JDBC i szkielety programistyczne ORM (ang. Object–Relational Mapping). Przeprowadzono badania w oparciu o 8 różnych kryteriów, w celu wyłonienia najbardziej efektywnej metody i narzędzia do pracy z relacyjnymi bazami danych, podczas tworzenia aplikacji w języku Java. Poszczególnym kryteriom przyznano wagi, które zostały określone poprzez ankietę przeprowadzoną wśród programistów języka Java i studentów informatyki. Najlepszym narzędziem bez uwzględnienia pozyskanych wag okazał się Hibernate, a z uwzględnieniem wag narzędzie JDBC.

Słowa kluczowe: połączenia z bazą danych; Java; wydajność; szkielety ORM

*Corresponding author

Email address: mateusz.zuchnik@pollub.edu.pl (M. Żuchnik)

1. Wstęp

We współczesnym świecie z roku na rok liczba osób korzystających z Internetu stale się powiększa. Jak podaje Główny Urząd Statystyczny w Polsce w roku 2020 dostęp do Internetu posiadało 90.4% gospodarstw domowych. W porównaniu z poprzednim rokiem odsetek ten był wyższy o 3,7 punktów procentowych [1]. Wzrost użytkowników korzystających z Internetu wpływa na objętość generowanych w nim danych. Przewiduje się, że w 2025 roku globalna ilość generowanych danych dziennie będzie wynosiła 463 eksabajtów [2]. Z uwagi na to, niezbędne jest posiadanie odpowiedniego systemu do gromadzenia danych i do szybkiego zarządzania nimi.

Jednym z rozwiązań, które pozwala gromadzić i zarządzać dużą ilością danych są bazy danych. Same bazy danych, a co za tym idzie systemy do zarządzania bazami danych znane jako DBMS (ang. DataBase Managment System) stanowią niemal nieodłączny element każdej aplikacji internetowej, desktopowej czy też mobilnej. Można wyróżnić takie DBMS jak: MySql, PostgreSQL, czy H2. Sam dostęp aplikacji do danych odbywa się często poprzez interfejs aplikacyjny ODBC (ang. Open Database Connectivity) [3], który pozwala na komunikację programu z bazą danych w celu przeprowadzenia operacji na tych danych. Język Java posiada swój odpowiednik interfejsu aplikacyjnego API (ang. Application Programming Interface) do łączenia się z DBMS. Nazwa odpowiednika to JDBC (ang. Java DataBase Connectivity) [4].

Język Java od momentu swojego pierwszego wydania 23 stycznia 1996 r. doczekał się bardzo dużej liczby bibliotek i narzędzi wykorzystującej interfejs JDBC. Wymienić tutaj można m.in. szkielety programistyczne takie jak: jOOQ Object Oriented Querying, MyBatis oraz Hibernate. Programiści często potrzebują wybrać konkretne rozwiązanie najbardziej pasujące do zdefiniowanych wymagań. Dotyczy to także połączenia z bazą danych. Z tej też potrzeby przeprowadzono badania mające na celu porównanie wymienionych narzędzi do komunikacji z bazami danych. Wyniki badań pomogą określić efektywność każdego z podejść i mogą ułatwić odpowiedni wybór.

Autorzy zapoznali się z podobnymi przeprowadzonymi badaniami. Wskazać należy artykuł pt.: „Analiza porównawcza technologii odwzorowania obiektowo-relacyjnego dla aplikacji Java”, autorstwa Piotra Błocha i Marka Wojciechowskiego. Celem autorów było zba-

danie możliwości i wydajności narzędzi do mapowania obiektowo-relacyjnego. Autorzy w swoich badaniach skupili się na takich narzędziach jak Hibernate, Oracle Toplink oraz JDO (ang. Java Data Objects). W przeprowadzonych badaniach udało się wykazać, że narzędzia Hibernate oraz Toplink są lepszym rozwiązaniem niż JDO [5].

Kolejnym przeanalizowanym artykułem była praca pt.: „Hybrydowe metody pracy z bazami danych w aplikacjach JEE", autorstwa Katarzyny Jóźwickiej i Mariusza Mitrusa. Celem artykułu była analiza narzędzi w kontekście operacji typu CRUD. W skład badanych narzędzi weszły: JDBC, Hibernate oraz szkielet aplikacyjny Spring. Przeprowadzone badania wykazały, iż najlepszym rozwiązaniem okazała się aplikacja hybrydowa łącząca JDBC i Hibernate. Takie połączenie narzędzi pozwoliło zmniejszyć czas na wykonanie zapytań SQL oraz zoptymalizować wykorzystanie pamięci RAM [6].

Artykuł naukowy pt.: „JEE database applications performance" autorstwa Magdaleny Grzesińskiej, Magdaleny Waszczyńskiej oraz Beaty Pańczyk, prezentuje porównanie wydajności bazodanowych aplikacji JEE z wykorzystaniem różnych interfejsów programistycznych
(JDBC, Hibernate, JOOQ). Wnioskami autorek z przeprowadzonych badań było to, że najlepszym rozwiązaniem dla operacji bazodanowych będzie podejście hybrydowe. W prostych operacjach typu CRUD sugerują, aby wykorzystać narzędzie Hibernate, a dla bardziej złożonych zapytań korzystać z rozwiązań jakie dostarcza jOOQ [7].

2. Cel i problem badawczy

Celem opisywanych badań była analiza porównawcza wydajności połączeń z bazami danych poprzez interfejs JDBC i szkielety programistyczne ORM. Do badań porównawczych wybrano następujące narzędzia: JDBC, jOOQ, MyBatis oraz Hibernate. Wpływ na wybór powyższych narzędzi miała ich rosnąca liczba wyszukiwań w serwisie Google [8] oraz gotowa konfiguracja dostarczana wraz z projektem Spring Data [9]. Rysunek 3 przedstawia charakterystykę popularności każdego z badanych narzędzi.

Problemem badawczym było wyznaczenie, które z wymienionych narzędzi jest najbardziej wydajne pod kątem określonych kryteriów. Uwzględniono kryteria, które w sposób bezpośredni lub pośredni wpływają na pracę programisty z danym narzędziem. Badania zostały przeprowadzone z wykorzystaniem relacyjnego systemu bazodanowego MySQL w wersji 8.0 [10].

W skład kryteriów wchodziły: czas wykonywania zapytania SQL (ang. Structured Query Language), w przypadku operacji typu CRUD, średnie zużycie pamięci RAM, średnie procentowe wykorzystanie procesora, popularność danego narzędzia, potrzeba znajomości SQL, niepodatność na ataki typu „SQL Injection", wsparcie dla dialektów języka SQL oraz sprawdzanie składni SQL. Wagi dla poszczególnych kryteriów zostały określone na podstawie badań ankietowych przeprowadzonych wśród programistów języka Java. Wartości uzyskanych wag zostały przedstawione w Tabeli 3.

3. Wybrane narzędzia

W ramach przygotowań do badań został przeprowadzony przegląd literatury dla badanych narzędzi do komunikacji z bazami danych. Podstawowe cechy każdego z nich przedstawiono poniżej.

3.1. JDBC

JDBC jest interfejsem programowania stworzonym przez ówczesnego właściciela języka Java, czyli firmę Sun Microsystems [11]. JDBC jest odpowiednikiem standardu ODBC i jego zasada działania jest bardzo zbliżona. Rozwiązanie to umożliwia na bezpośrednie połączenie z serwerem baz danych przy wykorzystaniu sterownika. JDBC umożliwia wysyłanie gotowych zapytań SQL, a w wyniku ich przetworzenia otrzymujemy zbiór rezultatów w formie tablicy danych.

3.2. jOOQ

Narzędzie jOOQ posiada mechanizm generowania kodu, który pozwala na budowanie bezpiecznych składniowo zapytań SQL i uzyskanie pełnej kontroli nad generowanym SQL poprzez czyste i wydajne API. Rozwiązanie to daje programiście możliwość uzyskania kontroli nad wygenerowanym kodem [12]. Narzędzie jOOQ w porównaniu do JDBC nie wymaga obsługiwania wszystkich wyjątków (ang. exceptions) w sposób jawny związanych z komunikacją z bazą danych.

3.3. MyBatis

MyBatis jest szkieletem programistycznym z otwartym kodem źródłowym (ang. Open Source). Zapewnia on wsparcie dla operacji typu CRUD (ang. Create Read Update Delete), gdzie logika samego mapowania SQL na obiekty jest oddzielona od kodu odpowiedzialnego za logikę biznesową. Dodatkowo zapewnia dwa sposoby konfiguracji: poprzez pliki konfiguracyjne XML, jak i przy użyciu adnotacji języka Java [13].

3.4. Hibernate

Ostatnim z badanych podejść do komunikacji z relacyjnymi bazami danych jest szkielet Hibernate. Hibernate jest przykładem technologii typu ORM. Dodatkowo szkielet ten w pełni zapewnia implementację dla specyfikacji JPA (ang. Java Persistence API), która została opracowana przez firmę Sun Microsystems [14]. Narzędzie to jest najbardziej rozbudowane i szeroko stosowane przez programistów Java [15].

4. Środowisko i metoda badawcza

Ze względu na charakter prowadzonych badań, niezbędnym było zdefiniowanie środowiska badawczego, na którym zostały przeprowadzone badania. Zdecydowana większość aspektów wykorzystywanych przy badaniu musiała zostać dobrana w taki sposób, aby była powtarzalna. Pozyskane wyniki dla każdej innej konfiguracji środowiska mogłyby znacząco odbiegać od

uzyskanych w niniejszej pracy. Biorąc pod uwagę fakt, iż głównym narzędziem potrzebnym do przeprowadzania badań był komputer, to samo środowisko badawcze zastosowane do wszystkich badań, zostało zdefiniowane poprzez sekcje sprzętową oraz programową.

4.1. Środowisko badawcze

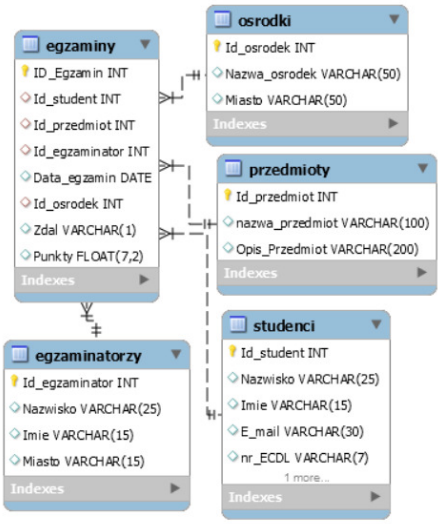
Specyfikacja sprzętowa została przedstawiona w tabeli 1, a specyfikacja oprogramowania została przedstawiona w tabeli 2. Baza danych składała się z 5 tabel: egzaminy, egzaminatorzy, studenci, ośrodki i przedmioty. Schemat struktury bazy danych przedstawiono na rysunku 1. Aplikacja testowa była uruchamiana z wykorzystaniem maszyny wirtualnej HotSpot JVM (ang. Java Virtual Machine) dostarczaną wraz z darmowym odpowiednikiem JDK 11 (ang. Java Development Kit), czyli OpenJDK w wersji 11.0.12+7. Wirtualna maszyna Javy została poddana dodatkowej konfiguracji. Początkowy rozmiar sterty ustawiono na 512MB, czego odpowiednikiem jest flaga (-Xms512m) oraz całościowy rozmiar sterty ustawiono na 2048MB (-Xmx2048m) dla JVM. Dla aplikacji testowej został wykorzystany domyślny GarbageCollector (GC) tzn. „G1 GC'. Dla potwierdzenia wykorzystania tego właśnie GarbageCollector'a, wykorzystano flagę „-XX:+UseG1GC''. Wersje narzędzi JDBC, jOOQ, MyBatis i Hibernate, były dostarczane wraz z wersją Spring Boota 2.4.5. Wykorzystany sterownik JDBC do połączeń z bazą danych MySQL, był również dostarczany w ramach projektu Spring Boota.

Tabela 1: Wykaz specyfikacji sprzętowej, na której zostały przeprowadzone badania

Producent	Lenovo
Model	ideapad 700-15isk
Procesor	Intel Core i5-6300HQ CPU 2,30Ghz
Pamięć RAM	8GB (SO-DIMM DDR4, 2133MHz)
Karta graficzna	NVIDIA GeForce GTX950M Intel HD Graphics 530
Dysk	Samsung SSD 970 EVO PLUS 1TB

Tabela 2: Wykaz oprogramowania. na którym zostały przeprowadzone badania

System operacyjny	Windows 10 Education
Java SDK	OpenJdk 11.0.12+7
Środowisko programistyczne	Intellij IDEA 2021.1
Baza danych	MySQL 8.0
Silnik bazodanowy	InnoDB
Spring Boot	2.4.5



Rysunek 1: Schemat bazy danych wykorzystany podczas przeprowadzania badań.

Każde z narzędzi dostępu do baz danych badano pod kątem 8 różnych kryteriów. Z uwagi na fakt, iż kryteria nie muszą być rozpatrywane na tym samym poziomie ważności tzn. jedno kryterium może być ważniejsze od drugiego, koniecznym było wyznaczenie ich wag. W związku z tym zdecydowano o przeprowadzeniu ankiety wśród programistów Java i studentów informatyki ostatniego roku studiów magisterskich. Do samej ankiety przystąpiło 26 osób. Ankieta składała się z 12 pytań, z czego dwa pierwsze dotyczyły doświadczenia programowania w języku Java. Pozostałe pytania dotyczyły bezpośrednio badanych kryteriów. Celem osoby wypełniającej ankiety było zaznaczenie jednej odpowiedzi w skali od 1 do 5, gdzie 1 oznaczało kryterium mało istotne, a 5 oznaczało kryterium bardzo ważne. Ankieta zawierała jedno pytanie kontrolne, w celu wyeliminowania nierzetelnych odpowiedzi. Samo pytanie kontrolne wymagało od użytkownika zaznaczenia konkretnej odpowiedzi, zdefiniowanej w pytaniu np.: w skali odpowiedzi od 1 do 5, użytkownik powinien zaznaczyć opcję 4. Wszyscy ankietowani odpowiedzieli na pytanie kontrolne poprawnie, zatem uwzględniono wszystkie wyniki zebrane z ankiety. Każde z kryteriów podlegało ocenie. Sposobem obliczania wagi danego kryterium była suma wszystkich odpowiedzi (uzyskanych punktów dla danego kryterium), dzielona przez liczbę ankietowanych. W taki sposób wagę stanowiła średnia arytmetyczna. Wyniki ankiety zostały przedstawione w sekcji „Wyniki".

4.2. Sposób punktowania kryteriów

Sposobem na wyznaczenie, które z badanych narzędzi jest najbardziej efektywne była ich weryfikacja pod kątem opracowanych kryteriów porównawczych wraz z uwzględnieniem wyznaczonych ankietowo wag. Analiza została podzielona na dwie części określone za pomocą aspektów ilościowych i jakościowych. Kryteria ilościowe to takie, których wynikiem była konkretna wartość np. czas wykonania zapytania wyrażony w milisekundach [ms] lub średnie wykorzystanie pa-

mięci RAM wyrażone w megabajtach [MB]. Kryteria jakościowe to takie, których nie da się zmierzyć, np. wsparcie dla dialektów. Uzyskanie wyników względem kryteriów jakościowych było możliwe dzięki analizie dokumentacji technicznej. Sam sposób punktowania był zależny od tego, czy dane kryterium było ilościowe lub jakościowe. Dla kryteriów ilościowych można na podstawie otrzymanych wyników każdemu narzędziu przyporządkować „uzyskane miejsce''. Jeśli dana metoda wypadła najlepiej w kontekście danego kryterium to otrzymywała 3 punkty, drugie najlepsze podejście 2 punkty itp. Najgorzej wypadające rozwiązanie otrzymywało 0 punktów. W przypadku kryterium jakościowego narzędzia mogły go spełniać lub nie, dlatego wynik był jedną z dwóch możliwych wartości. W zależności od tego czy narzędzie spełniało dane kryterium otrzymywało 2 punkty bądź 0 punktów. W końcowym procesie oceniania uzyskane wagi stanowiły mnożnik do uzyskanych punktów dla danego kryterium.

4.3. Kryteria ilościowe

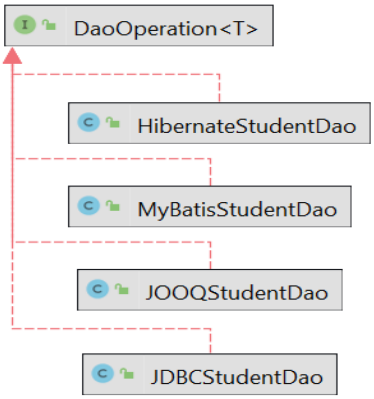
Sposobem pozyskiwania wyników dla poszczególnych kryteriów ilościowych było przeprowadzenie eksperymentu mającego na celu zagregowanie wyników dla poszczególnych przypadków testowych. W skład tych kryteriów wchodziły następująco: czas wykonania zapytania SQL (operacje CRUD), średnie zużycie pamięci RAM, średnie procentowe wykorzystanie procesora oraz popularność danego podejścia mierzona poprzez liczbę zapytań w wyszukiwarce Google. Dla każdego kandydata z grupy przygotowano zapytania SQL typu CRUD i przeprowadzono eksperyment dla 5 zestawów danych. Były to odpowiednio: 1 rekord, 10 rekordów, 100 rekordów, 1000 rekordów i 10 000 rekordów. W zależności od typu operacji dane te były odczytywane, dodawane, aktualizowane lub usuwane z bazy danych. Wszystkie zapytania podlegały mechanizmowi „grupowania" (ang. batch), w celu zwiększenia wydajności dla dużych porcji danych.

4.4. Kryteria jakościowe

W skład kryteriów jakościowych wchodziły następująco: wymaganie znajomości przez programistę języka SQL, zabezpieczenie przed atakami typu „SQL Injection" [16], wsparcie dla dialektów SQL oraz wspomaganie programisty poprzez sprawdzanie składni SQL. Kryteria tego typu najczęściej wynikały bezpośrednio z dokumentacji technicznej dostępnej na stronach autorów.

5. Aplikacja testowa

W celu zbadania kryteriów ilościowych, konieczne było napisanie aplikacji do komunikacji z bazą danych. Zadaniem aplikacji było realizowanie przypadków testowych, mierzenie czasu wykonania danej operacji SQL, mierzenie wykorzystania pamięci RAM oraz obserwacja procentowego wykorzystania procesora przy komunikacji z bazą danych. W pierwszym kroku stworzono interfejs dla języka Java, który definiował, jakie metody mają zostać zaimplementowane przez badane narzędzia do komunikacji z bazą danych. Metody z interfejsu Java reprezentowały każdą z operacji typu CRUD. Struktura implementacji dla stworzonego interfejsu przedstawia Rysunek 2.



Rysunek 2: Diagram przedstawiający klasy implementujące interfejs DaoOperation.

Cała aplikacja została napisana w języku Java z wykorzystaniem szkieletu programistycznego Spring Boot w wersji 2.4.5 [17]. Dane testowe były generowane w aplikacji z wykorzystaniem biblioteki JavaFaker w wersji 1.0.2. Kod aplikacji testowej mierzący czas wykonania operacji nie powinien wpływać na wynik samego testu, więc wykorzystano podejście programowania aspektowego dzięki projektowi Spring AOP (ang. Aspect-Oriented Programming). Aby całość działała poprawnie, stworzono dodatkową adnotację języka Java (measureExecutionTime), która była wykorzystywana do oznaczania metod, podlegających pomiarowi. Zadaniem stworzonej adnotacji był pomiar czasu wykonania danej metody. Każda z metod dla danego badanego narzędzia odpowiedzialnego za komunikację została oznaczona taką adnotacją na poziomie dostępu do danych (ang. Data Access Object - DAO). Zadaniem omawianej adnotacji było pobranie czasu systemowego przed wykonaniem operacji na bazie danych i bezpośrednio po wykonaniu. Obliczając różnice zmierzonych czasów otrzymano wynik w postaci czasu wykonania samej operacji na bazie danych. Dla zmierzenia średniego procentowego wykorzystania procesora oraz średniego wykorzystania pamięci RAM, wykorzystano oprogramowanie JProfiler [18]. Oprogramowanie to umożliwia śledzenie aktywności GC, dzięki czemu można wykluczyć te pomiary podczas których GC rozpoczął swoje działanie. W badaniu uwzględniono tylko te pomiary dla których GC rozpoczął i zakończył swoje działanie bezpośrednio przed pomiarem. Dodatkowo GC nie uruchamiał się, aż do zakończenia samego pojedynczego pomiaru. Wszystkie dane pozyskane podczas badań były zapisywane do pliku, który później posłużył do obliczenia wartości średnich arytmetycznych. Dzięki zewnętrznemu oprogramowaniu możliwe było poznanie charakterystyki wykorzystania procesora w przypadku większych zapytań SQL.

6. Wyniki

W pierwszej kolejności zostały wyliczone wagi dla każdego z kryterium na podstawie danych pozyskanych poprzez przeprowadzenie ankiety. Wagi zostały przedstawione w Tabeli 3. Tabela 4 przedstawia wyniki 3 pomiarów dla operacji odczytywania z bazy danych. Tabela 5 przedstawia te same pomiary tym razem dla operacji dodawania rekordów do bazy danych. Tabela 6 przedstawia pomiary dla aktualizowania rekordów, a Tabela 7 przedstawia pomiary dla usuwania rekordów z bazy danych.

Tabela 3: Uzyskane wagi dla poszczególnych kryteriów

Nr	Kryterium	Waga
1	Szybkość wykonywania zapytań SQL	4,65
2	Stopień wykorzystania procesora	3,69
3	Stopień wykorzystania pamięci RAM	3,46
4	Popularność technologii	3,08
5	Potrzeba znajomości SQL	2,77
6	Niepodatność na ataki typu „SQL Injection”	4,19
7	Wykorzystanie dialektów	3,12
8	Sprawdzanie składni SQL	2,77

w procesie ankietowania

Tabela 4: Pomiary dla operacji odczytu z bazy danych (liczba powtórzeń 50)

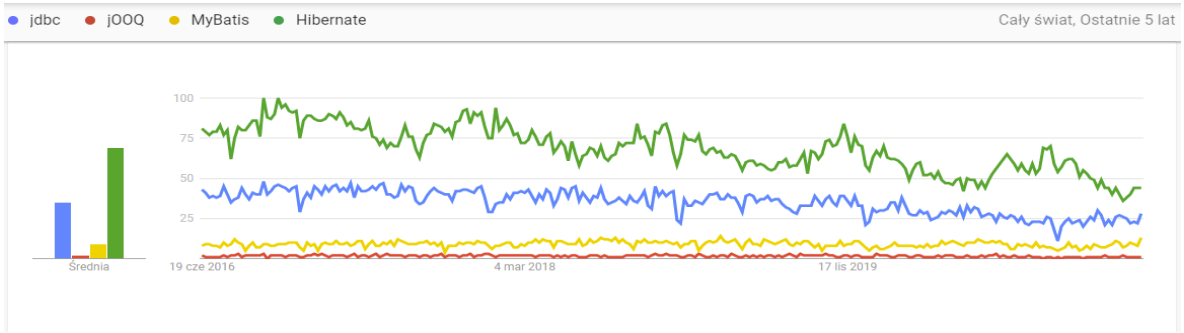
Zapytanie SQL	SELECT * FROM Studenci			
Mierzone parametry	Średni czas wykonania operacji [ms] Średnie zużycie pamięci RAM [MB] Średnie wykorzystanie procesora [%]			
Liczba rekordów	JDBC	jOOQ	MyBatis	Hibernate
1	1,4191 29,25 0,80	2,8537 30,55 2,32	2,0039 29,32 2,32	4,4142 30,52 3,21
10	1,8799 29,26 0,77	3,0520 30,82 1,94	2,3297 29,81 3,43	5,0645 30,56 3,90
100	2,7873 29,80 1,17	4,9258 30,89 2,35	3,9625 29,82 4,52	7,3024 30,77 4,27
1 000	7,1629 30,10 1,17	10,4747 31,57 2,75	11,2639 30,93 5,85	14,4853 32,10 4,29
10 000	28,0151 38,38 2,34	43,8685 46,28 5,07	41,3272 43,34 7,02	54,0078 49,02 7,04

Tabela 5: Pomiary dla operacji dodawania rekordów do bazy danych (liczba powtórzeń 20)

Zapytanie SQL	INSERT INTO Studenci VALUES (…)			
Mierzone parametry	Średni czas wykonania operacji [ms] Średnie zużycie pamięci RAM [MB] Średnie wykorzystanie procesora [%]			
Liczba rekordów	JDBC	jOOQ	MyBatis	Hibernate
1	2,3616 30,27 0,37	2,8759 30,66 1,17	3,009 30,32 0,37	4,9857 30,48 0,37
10	8,7373 30,28 0,40	15,332 31,18 1,94	9,3764 30,38 1,17	23,3646 31,02 0,49
100	34,2451 30,38 0,49	52,0805 32,76 1,94	35,1963 32,42 1,94	132,9326 33,11 5,85
1 000	202,2956 33,69 1,59	247,1499 50,99 7,42	222,75 47,58 5,1	580,8474 56,97 12,5
10 000	1793,546 79,85 10,10	2058,063 237,4 16,80	1893,3111 204,2 11,23	4573,8852 290,40 17,57

Tabela 6: Pomiary dla operacji aktualizowania rekordów w bazie danych (liczba powtórzeń 20)

Zapytanie SQL	UPDATE Studenci Studenci SET imie = -[…] WHERE id = […]			
Mierzone parametry	Średni czas wykonania operacji [ms] Średnie zużycie pamięci RAM [MB] Średnie wykorzystanie procesora [%]			
Liczba rekordów	JDBC	jOOQ	MyBatis	Hibernate
1	3,6718 30,25 0,37	4,9383 30,42 0,37	4,6767 30,30 0,37	4,3478 30,22 1,17
10	10,3425 30,26 0,40	12,0667 30,66 0,77	12,7007 30,32 0,77	13,6078 31,25 2,73
100	41,4706 30,85 1,17	39,4382 32,25 1,17	45,0012 30,87 0,79	123,5245 34,60 3,54
1 000	217,625 34,20 3,90	214,275 47,96 9,77	222,6127 35,79 4,30	621,0228 67,74 17,30
10 000	1667,94 78,66 8,90	1812,87 196,40 17,57	1781,577 91,05 14,62	4690,511 323,50 25,34



Rysunek 3: Wykres przedstawiający popularność wyszukiwania danego narzędzia w przeglądarce Google.

Tabela 7: Pomiary dla operacji usuwania rekordów z bazy danych (liczba powtórzeń 20)

Zapytanie SQL	DELETE FROM Studenci WHERE id = […]			
Mierzone parametry	Średni czas wykonania operacji [ms] Średnie zużycie pamięci RAM [MB] Średnie wykorzystanie procesora [%]			
Liczba rekordów	JDBC	jOOQ	MyBatis	Hibernate
1	5,1196 29,25 0,39	6,6558 30,15 0,40	6,6676 30,31 0,40	10,3639 30,48 0,34
10	10,1224 29,25 0,77	13,1132 30,67 0,40	12,9217 30,33 1,17	36,5221 30,50 0,36
100	34,4795 30,31 0,79	43,0083 30,72 0,77	36,1948 30,38 2,72	126,9967 30,57 0,39
1 000	226,9031 33,69 1,94	259,0934 33,58 4,29	245,3846 34,77 5,06	567,6597 36,07 5,74
10 000	2010,299 62,36 11,50	2208,017 133,90 14,30	2049,5607 80,29 11,87	4247,6016 285,8 17,80

Tabele od 8 do 11 przedstawiają wyniki badań pod względem kryteriów jakościowych. Tabela 8 przedstawia czy dane narzędzie wymaga znajomości SQL od programisty. Tabela 9 przedstawia czy dane rozwiązanie zabezpiecza przed atakami typu „SQL Injection”.

Tabela 8: Potrzeba znajomości SQL

Potrzeba znajomości SQL			
JDBC	jOOQ	MyBatis	Hibernate
TAK	NIE	TAK	NIE

Tabela 9: Zapewnienie mechanizmu ochrony przed atakami typu „SQL Injection"

Ochrona przed atakami typu „SQL Injection’’			
JDBC	jOOQ	MyBatis	Hibernate
NIE	TAK	TAK	TAK

Tabela 10: Wsparcie dla dialektów

Wsparcie dla dialektów			
JDBC	jOOQ	MyBatis	Hibernate
NIE	TAK	NIE	TAK

Tabela 11: Sprawdzanie poprawności składni SQL

Sprawdzanie składni SQL			
JDBC	jOOQ	MyBatis	Hibernate
NIE	TAK	NIE	TAK

Tabela 10 przedstawia czy dane narzędzie wprowadza wsparcie dla dialektów. Tabela 11 przedstawia, które z rozwiązań sprawdza poprawność składni SQL.

6.1. Punktacja

Tabela 12 prezentuje punktację, którą uzyskały badane narzędzia w kontekście poszczególnych, określonych wcześniej kryteriów. Do ostatecznej oceny na poziomie kryterium zostały uwzględnione dodatkowo wagi. Po zsumowaniu wszystkich punktów w ramach danego rozwiązania otrzymano finalną punktację, na podstawie której wskazano najbardziej efektywne podejście do pracy z bazami danych w języku Java.

Tabela 12: Punktacja technologii bez uwzględnienia wag

Nr kryterium	JDBC	jOOQ	MyBatis	Hibernate
1	3	1	2	0
2	3	0	2	1
3	3	1	2	0
4	2	0	1	3
5	0	2	0	2
6	0	2	2	2
7	0	2	0	2
8	0	2	0	2
Suma	11	10	9	12

Tabela 13: Finalna suma punktów z uwzględnieniem wag

Tech.	JDBC	jOOQ	MyBatis	Hibernate
Suma	41,56	33,81	35,06	38,63

7. Wnioski

Celem badań było wykazanie, które narzędzie do komunikacji z relacyjną bazą danych jest najefektywniejsze. Rezultat ten udało się osiągnąć przeprowadzając niniejsze badania. Należy zauważyć, że wprowadzenie wag uzyskanych w procesie ankietowania, znacząco zmienia finalną punktację i końcową ocenę badanych technologii. Jeżeli kryteria zostałyby potraktowane jako równoważne, najefektywniejszym narzędziem do pracy z bazami danych zostałby Hibernate, następne w kolejności byłby JDBC, potem jOOQ i na końcu MyBatis. Przy wprowadzeniu systemu wag, wyniki narzędzi zostały rozpatrzone pod kątem najważniejszych kryteriów i to one w głównej mierze wpływały na końcową ocenę. Przy uwzględnieniu wag zgodnie z tabelą 13 najefektywniejszym rozwiązaniem zostaje JDBC, następnie Hibernate, potem MyBatis i jOOQ. Dla ankietowanych najważniejszymi kryteriami były: szybkość wykonywania zapytań SQL oraz niepodatność na ataki typu „SQL Injection". Za to najmniej ważnymi kryteriami zostały: potrzeba znajomości SQL oraz sprawdzanie składni SQL.

Dodatkowo analizując wyniki stwierdzono, że operacją potrzebującą najwięcej zasobów komputera, w którego skład wchodzą: wykorzystanie procesora oraz pamięci RAM jest operacja dodawania rekordów do bazy danych. Za to operacją potrzebującą najmniej zasobów jest operacja pobierania rekordów z bazy danych.

Literatura

[1] Główny urząd statystyczny, społeczeństwo informacyjne w Polsce w 2020 roku, https://stat.gov.pl/obszary-tematyczne/nauka-i-technika-spoleczenstwo-informacyjne/spoleczenstwo-informacyjne/spoleczenstwo-informacyjne-w-polsce-w-2020-roku,1,14.html , [17.09.2021].

[2] J. Desjardins, How much data is generated each day?, World Economic Forum 2019, https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/ , [17.09.2021].

[3] Dokumentacja programistyczna ODBC, https://docs.microsoft.com/en-us/sql/odbc/reference/odbc-programmer-s-reference?view=sql-server-ver15 , [19.09.2021].

[4] Dokumentacja programistyczna JDBC, https://docs.oracle.com/cd/E11882_01/java.112/e16548/toc.htm , [19.09.2021].

[5] P. Błoch, M. Wojciechowski, Analiza porównawcza technologii odwzorowania obiektowo-relacyjnego dla aplikacji Java. XIII Konferencja PLOUG: Systemy informatyczne. Projektowanie, implementowanie, eksploatowanie, Zakopane (2007).

[6] K. Jóźwicka, M. Mitrus, Hybrydowe metody pracy z bazami danych w aplikacjach JEE. Journal of Computer Sciences Institute, (2019) 12.

[7] M. Grzesińska, M. Waszczyńska, B. Pańczyk, JEE DATABASE APPLICATIONS PERFORMANCE. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 6(4) (2016) 73-76.

[8] Liczba wyszukiwań badanych narzędzi w serwisie Google, https://trends.google.pl/trends/explore?q=jdbc,jooq,mybatis,hibernate , [20.09.2021].

[9] Dokumentacja techniczka Spring Data, https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference ,[20.09.2021].

[10] Dokumentacja techniczna systemu bazodanowego MySQL, https://dev.mysql.com/doc/ , [17.09.2021].

[11] G. Reese, Database Programming with JDBC and JAVA, O'Reilly Media Inc, 2000.

[12] K. Siva Prasad Reddy, Working with JOOQ. In: Beginning Spring Boot 2, Apress, Berkeley CA (2017) 71-82.

[13] K. Siva Prasad Reddy, Java Persistence with MyBatis3, Packt Publishing Ltd, 2013.

[14] Dokumentacja Java Persistence API, https://javadoc.io/doc/javax.persistence/javax.persistence-api/latest/index.html , [20.09.2021].

[15] P. T. Fisher, B. D. Murphy, Spring persistence with Hibernate, Apress (2010).

[16] J. Clarke-Salt, SQL injection attacks and defense, Elsevier, 2009.

[17] C. Walls, Spring Boot in action, Manning Publications, 2016.

[18] N. Kumari, R. Kumar, Profiling JVM for AI Applications Using Deep Learning Libraries, In Machine Learning for Predictive Analysis, Springer, Singapore (2021) 395-404.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 316–323
Received: 09 September 2021
Accepted: 20 September 2021

Examination of text's lexis using a Polish dictionary

Badanie leksyki tekstu na podstawie słownika języka polskiego

Roman Voitovych*, Edyta Łukasik

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

This paper presents an approach to compare and classify books written in the Polish language by comparing their lexis fields. Books can be classified by their features, such as literature type, literary genre, style, author, etc. Using a preassembled dictionary and Jaccard index, the authors managed to prove a lexical likeness for books. Further analysis with the PAM clustering algorithm presented a lexical connection between books of the same type or author. Analysis of values of similarities of any particular field on one side and some anomalous tendencies in other cases suggest that recognition of other features is possible. The method presented in this article allows to draw conclusions about the connection between any arbitrary books based solely on their vocabulary.

Keywords: natural language processing; lexis analysis; Jaccard similarity coefficient; Partitioning Around Medoids;

Streszczenie

Artykuł prezentuje metodę porównania i klasyfikacji książek napisanych w języku polskim na podstawie ich leksyki. Książki można dzielić, korzystając z ich cech, np. rodzaju literatury, gatunku literackiego, stylu, autora itp. Korzystając ze skompilowanego słownika i indeksu Jaccarda, udowodniona została hipoteza dotycząca podobieństwa książek rozpatrywanego pod kątem ich leksyki. Kolejna analiza za pomocą algorytmu klastrowego PAM wskazuje na związek leksykalny pomiędzy książkami jednego rodzaju literatury lub autora. Analiza wartości współczynników poszczególnych obszarów z jednej strony i anomalia w zachowaniu w niektórych przypadkach sugeruje, że wyodrębnienie kolejnych cech jest możliwe. Metoda przedstawiona w tym artykule pozwala wyciągać wnioski o relacjach między książkami, korzystając wyłącznie z ich słownictwa.

Słowa kluczowe: przetwarzanie języka naturalnego; analiza leksyczna; indeks Jaccarda; Partitioning Around Medoids;

*Corresponding author

Email address: voytroman@protonmail.com (R. Voitovych)

1. Introduction

The purpose of this article is to investigate the possibility to automatically compare different books with each other by their lexis fields and draw conclusions regarding their features, such as literature type, literary genre, scientific field, author, etc. The hypothesis concerning connection between lexis likeness and common characteristics is studied. The target language of this study is Polish, which has an enormous amount of morphology-driven declensions which causes much greater difficulties for any natural language processing tool. For example, English and other, more inflectional, Germanic and Romance languages are much more simpler than Polish from this point of view.

This idea implies the usage of a thorough dictionary including every possible declension, that allows to lemmatise texts and analyse them as a set of lemmas. It also requires a storage solution, both for the dictionary and analysis results, a comparison algorithm such as a similarity or difference coefficient of measures and finally a clustering algorithm to ease the analysis of big comparison scenarios.

Former works [1-5], some of which examine similar literature classification problem [6-7], present a lot of possible solutions for text differentiation yet part of them concerns not only lexical similarity but also a semantic one which is not in the scope of this work. The semantic comparison allows to get a match between two texts by their meanings without relying on the words used. Yet long texts like novels, textbooks or legal codes have a tendency to contain an enormous amount of ideas, events and plots and that may render semantic analysis's results questionable. Other algorithms group neighbouring words into structures like vectors but this research treats every word independently. The primary idea behind this research is that the choice of words in any book is defined by their features.

The theme of this thesis falls into a domain of general natural language processing [8], which in turn is a comprehension approach of computational linguistics [9]. Some elements of language morphology studies concerning lemmas and declensions from [10] are used. Also, the Jaccard similarity coefficient [11] and Partitioning Around Medoids implementation of the k-medoids algorithm [12] are used as research tools.

Three primary research questions of this article are:

- Q1. Does the comparison between lexis fields allow to state the overall difference in terms of any grouping features?
- Q2. Is the lexical similarity of books augmented by the mutual author?
- Q3. Is there a significant difference between different types of non-fiction and fiction literature?

2. Materials and methods

2.1. Dictionary

The SJP (Słownik Języka Polskiego) is a free internet database devoted to tracking every possible Polish word and its morphology [13]. It provides a quite reliable index of Polish words including their declensions. This dictionary appears to have an extremely high word coverage ranging from 97% to 99% for average fiction book's vocabulary but its form is not fully compatible with chosen database schema. Firstly, every word in the database is unique in the scope of one table, thus meaning that the number of declensions will be reduced to only one instance. This rule does not exclude any possible repetition of lemmas as declensions of other lemmas. In such cases, all their declension forms will be discarded with only the lemma form left. Without this procedure, there will be exactly 2 388 or 1.13% of all lemmas stored in the database as declensions.

Many of the words stored in this database are not common names but proper names, abbreviations, initialisms, phrases, etc. Importing function separates proper names (recognised by initial capital letters) into table "proper" and initialisms with abbreviations (recognised by non-initial capital letters, dots and hyphens) into table "abbreviation" discarding all words containing spaces and other non-letter symbols. That process allows to get a refined collection of common words stored in the table "common", yet it does not exclude words derived from proper nouns, foreign words and or similar odd cases.

The newly imported SQLite lexis database with all aforementioned corrections applied has 172 705 lemmas from which 165 737 are mentioned as lemmas in declension entries. The absolute number of all common word entries is 3 980 250. It also includes 303 286 proper word inflexions and 11 241 abbreviation and initialism forms. Its fragment is shown in Table 1.

Table 1: A sample slice of common word forms from the lexis database.

1532119	odchody	odchód	noun
1532120	odchodzie	odchód	noun
1532121	odchów	NULL	noun
1532122	odchowach	odchów	noun
1532123	odchowami	odchów	noun
1532124	odchowem	odchów	noun
1532125	odchowie	odchów	noun
1532126	odchowom	odchów	noun
1532127	odchowowi	odchów	noun
1532128	odchowów	odchów	noun
1532129	odchowu	odchów	noun
1532130	odchowy	odchów	noun
1532131	odchrzaniać	NULL	verb
1532132	nieodchrzaniająca	odchrzaniać	verb
1532133	nieodchrzaniającą	odchrzaniać	verb

2.2. Parser

Research purposes require a robust parser that can read any text written in the Polish language and map every recognised word to an individually created database file based on the lexis database schema. It has additional columns named "count" and "probability", which allow to track the number of parsed words. Such an approach allows to keep track of every word's properties without the need to store the entire dictionary.

The first step is to read text files containing target text. Popular text file formats can store all Polish letters in several encodings like Windows-1250 or Unicode, but the availability of many foreign words from French, Spanish, German and other languages which use Latin diacritics forces the use of variable-width encoding. The chosen code page is the widespread UTF-8.

All of the tools in this article are written in Windows 10 environment. The chosen implementation language is C due to its robust speed and efficiency. More specifically it uses the MinGW-w64 compiler. This combination already causes problems concerning UTF-8 parsing because of default wchar.h library appears to have conflict with a console window that misreads Unicode input. Secondly, all of the conversion and reading functions do not work with symbols that have multiple-byte width. The first problem is omitted by using the Windows-1250 code table at input while all needed read and conversion functions are manually restored.

```
        – Stasiu, powiedz sam, czy jest na świecie człowiek, k
tóremu mógłbym oddać ten mój skarb i to moje kochanie z większ
ą ufnością?
        Młodzi państwo Tarkowscy pozostali aż do śmierci pana
Rawlisona w Anglii, a w rok później wyruszyli w długą podróż.
Ponieważ przyrzekli sobie odwiedzić te miejsca, w których spęd
zili najmłodsze lata, a potem błąkali się niegdyś jako dzieci,
 podążyli więc przede wszystkim do Egiptu. Państwo Mahdiego i
Abdullahiego dawno już runęło, a po jego upadku „nastąpiła", j
ak mówił kapitan Glen, Anglia. Z Kairu zbudowano do Chartumu k
olej. Oczyszczono sudy, czyli rozlewiska nilowe, tak że młoda
para mogła dotrzeć wygodnym parowcem nie tylko do Faszody, ale
 aż do wielkiego jeziora Wiktoria-Nianza. Z miasta Florence, l
eżącego nad brzegiem tegoż jeziora, udali się koleją do Mombas
sa. Kapitan Glen i doktor Clary przenieśli się już byli do Nat
alu, ale żył w Mombassa pod troskliwą opieką miejscowych władz
 angielskich King. Olbrzym poznał natychmiast dawnych swych pa
ństwa i szczególniej Nel witał tak radosnym trąbieniem, że aż
pobliskie drzewa mangrowiowe trzęsły się jak od wiatru. Poznał
 również starego Sabę, który przeżył niemal dwukrotnie zwykłe
psie lata i choć trochę już niewidomy, towarzyszył Stasiowi i
Nel wszędzie.
        Staś dowiedział się na miejscu, że Kali cieszy się dob
rym zdrowiem, że włada, pod protektoratem angielskim, całą kra
iną na południe od Jeziora Rudolfa i że sprowadził misjonarzy,
 którzy szerzą wśród dzikich miejscowych szczepów chrześcijańs
two.



        Po tej ostatniej podróży młodzi państwo Tarkowscy powr
ócili do Europy i osiedli wraz z sędziwym ojcem Stasia na stał
e w Polsce.



Parsed 99996 words, recognized 98211 (98%) words or 98% of sym
bols.
>>_
```

Figure 1: Output produced by parser reading "W pustyni i w puszczy".

While the parser reads UTF-8 text file byte by byte it arranges it in words, counts them and tries to find a match in the lexis database. To find the match it rotates the letter case of initial and other letters and breaks

down words with hyphens or dots into smaller parts which, for example, allows to analyse hyphenated compound adjectives as separate words. When a match is found its corresponding entry is added to the text's database file. When the parser recognises a repeated use of a previously added word it simply amplifies the "count" value by one.

After reading the whole file program writes the total number of recognised and unrecognised words and symbols into a separate "stats" table. It allows to normalise results by dividing count number by total number and to store this value in the "probability" field.

During this process, the console window presents the progress in the form of a plain currently parsed coloured text as it is presented in Figure 1, where green colour indicates recognised symbols, red – unrecognised and black – ignored ones.

2.3. Analyser

The current state of science presents many different methods to compare two sets of labelled data [3,5] part of which are purely lexical and others employ a semantical analysis, yet only lexical ones are used here. To show off the basic idea of text's attributes strictly defined by its vocabulary the most basic and widely known Jaccard similarity coefficient [5,11] is used:

$$\mathrm{J}(\mathrm{A},\mathrm{B}) = \frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A}| + |\mathrm{B}| - |\mathrm{A} \cap \mathrm{B}|} \tag{1}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are finite sets. The definition implies that the Jaccard index receives values between and including 0 and 1, where 0 means no similarity between sets and 1 shows that $\boldsymbol{A}$ and $\boldsymbol{B}$ are identical.

Listing 1: Jaccard index implementation.

```
    sqlite3_exec(_compare_db, SQL_JACCARD_INNER,
                 jaccard_index_inner_callback, NULL, NULL);
    sqlite3_exec(_compare_db, SQL_JACCARD_LEFT,
                 jaccard_index_left_callback, NULL, NULL);
    sqlite3_exec(_compare_db, SQL_JACCARD_RIGHT,
                 jaccard_index_right_callback, NULL, NULL);

    result = _jaccard_ab/(_jaccard_a+_jaccard_b-_jaccard_ab);
    return result;
}

int jaccard_index_inner_callback(void *null_data, int argc,
                                 char **argv, char **argcol){
    double a, b;
    a = atof(argv[0]);
    b = atof(argv[1]);
    if(a>b){
        _jaccard_ab+=b;
        _jaccard_a+=a-b;
    }
    else{
        _jaccard_ab+=a;
        _jaccard_b+=b-a;
    }
    return 0;
}
```

The implementation of this formula (Listing 1) needs to take into consideration the schema of statistical data. Every text database contains a list of words and the probability of their occurrence. Before any comparison can be made analyser module converts declensions into lemmas summarising their probability measures and writes to a comparison database in two separate tables from two texts' databases correspondingly. $\mathbf{A} \cap \mathbf{B}$ is defined as a lesser of two probabilities for every match between lemmas. The remainder of those operations is added to the rest of unmatched lemmas either in set $\boldsymbol{A}$ or $\boldsymbol{B}$.

2.4. Partitioning Around Medoids algorithm

Using the aforementioned implementation of the Jaccard index program compares n books between each other and builds $n \times n$ symmetrical matrix from which every pair value (a, b) hold a coefficient between books a and b. Yet it is still hard to draw conclusions concerning classification without the appropriate clustering algorithm.

The metric in which those similarity indices are measured does not use the standard Euclidean geometry which is required for a k-means clustering algorithm to compute centroids. The only measures available are similarities which do not allow to derive any dependent value without the metric. But similar k-medoids algorithm solves this problem by using only existing points as centroids and requires the set of distances or in this case inverted similarities to run properly.

In this paper a Partitioning Around Medoids (PAM) algorithm [12] which uses a greedy search to speed up the process was implemented. It aims to find clusters with high degree of similarity while the similarity between clusters stays low. There is no strict rule for selecting the number of clusters k so it must be manually determined for every scenario.

PAM has two distinct phases – BUILD and SWAP. During the BUILD phase algorithm selects initial predefined number of centroids. Then in the SWAP phase, it considers pairs of objects with an object selected as centroids and an unselected object to swap them if it decreases the total dissimilarities inside clusters. Every time the algorithm changes an array of selected objects it recalculates two values for every object p - D_p, the dissimilarity between p and the closest selected object and E_p, the dissimilarity to the second closest selected object. $d(i, j)$ is a dissimilarity between objects i and j.

The BUILD phase is initialised by adding a first object for which the sum of distances to other objects is minimal. Then it considers unselected objects to select by computing and choosing the best total gain for every pair of unselected objects i and j:

$$g_i = \sum C_{ji} \tag{2}$$

where

$$C_{ji} = \max\{D_j - d(j, i), 0\} \tag{3}$$

It repeats this calculation $k - 1$ times resulting in a set of k centroids.

Then during the SWAP phase, it considers every pair of (i, h) where i is a selected object and h is an unselected object and calculates the contribution:

$$T_{ih} = \sum K_{jih} \tag{4}$$

where K_{jih} is the contribution of swapping object i and h concerning other unselected object j. If $d(j,i) > D_j$, then $K_{jih} = \min\{d(j,h) - D_j, 0\}$, but if $d(j,i) = D_j$, then $K_{jih} = \min\{d(j,h), E_j\} - D_j$. After selecting the minimal contribution $T_{ih} < 0$ algorithm swapes the corresponding pair (i,h), recalculates values D_p and E_p and tries again.

If total contribution T_{ih} equals zero or more, then the best combination is already found and PAM is stopped. The values of D_p determine to which cluster belongs object p.

Unfortunately due to the unknown metric, there is no way to visualise results from a plain list of clusters and their objects.

3. Research and selected results

Any if not all features, can be purely a subject of semantics and not lexis. Those two concepts are quite related, but lexical units can have an enormous number of semantic definitions and a lot of them are incredibly flexible. One could argue that semantic units can use only a limited number of lexical units unless the author is willing to communicate using words, which are detached from their original meanings. Even if it happens, it usually distorts the denotation of complete phases and still relies on dictionary definitions of separate words, thus making the word's connotations heavily dependent on its original meaning.

Considering that analysis used in this article is purely lexical and does not use semantics or relations between parsed words, such as vectors, its proper function is fully reliant on the connection between the text's dictionary and its properties.

Before concluding any further research the proof using the comparison between loosely related and absolutely similar pairs of books in terms of features is needed. If this relation and comparison by Jaccard coefficient are correlating, then a further k-medoids clustering allows us to group books into clusters with some common features.

All of the scenarios mentioned below use Polish fiction and non-fiction medium-sized books as the test subjects. Fiction books are sourced entirely from the public domain and non-fiction books are taken either from Lublin University of Technology Publishing House or Journal of Laws of the Republic of Poland.

3.1. Theoretical constraints and implications

Research results require a definition of how lexis field similarity is represented by the Jaccard similarity coefficient works. Absolute results i.e. zero and one correspond to absent and total similarity accordingly, yet this is irrelevant in practical comparison. Those two values show one major metric difference between the two ideal states. In a case when a pair with one definite book on the left and one arbitrary book on the right it gets a coefficient value equal to one only when both lexis fields are identical. But to get zero it can use an indefinite number of different lexis fields for the second book all of which will have an unconstrained range of similarity between themselves.

Thus if $s(a,b) = 1$ and $s(a,c) = 1$ (where s is a function of similarity coefficient and sets a,b,c are dictionaries with elements which represent words) are true then the statement $s(b,c) = 1$ is also true, which also means that sets a,b,c are identical. But if $s(a,b) = 0$ and $s(a,c) = 0$ are true then the value of $s(b,c)$ stays unknown and can have any value between 0 and 1.

Also, those two states are probably unachievable in any real scenario. It is highly unlikely for long texts such as books to have an identical set of words with alike probabilities of their individual use, even if the contents of those books are very similar. The abundance of so-called function words such as conjunctions, prepositions, pronouns, etc. and the overall tendency to use high-frequency words makes zero and neighbouring values of coefficient very unlikely. Those two constraints mean that practical absolute values are distanced from their theoretical counterparts.

During the examination of one or more types of features it has to be assured that inside every chosen group of books there is a unique feature-based connection that is not present neither between books from different groups nor inside any other group. Secondly, there is a problem with feature selection. No characteristic is unique only to one element of one feature type. It is safe to assume that most of the literary divisions are intertwined with one another. Authorship, time of writing, publishing house, type of literature, literary style, genre and an unknown number of other divisions are probably connected with different but partially similar lexes. It means that the part of the resulting book's lexis is not concatenated from feature lexes but rather superimposed by their repeating parts, thus augmenting their frequency.

If the clustering algorithm successfully recreates initial grouping then there is surely a correlation between chosen feature likeness and the lexis likeness. However, it is essential to analyse how coefficients behave inside and outside the aforementioned groups to prove causation.

Consequently, two basic terms concerning mean similarities inside and outside these groups are introduced to draw conclusions regarding levels of similarity in those scenarios. *Background noise similarity* is a mean of all relations for every book pair outside the groups. In a properly build scenario it will correspond to a practical minimum for a chosen background feature field. Similarly, *common inner similarity* or separate *inner similarities* for every group show how much more similar those books are to the books of respective groups. The *foreground similarity* is the difference between *common inner similarity* and *background noise*. Its resulting value ignores the "noise" of unomittable basic vocabulary (function words and most frequent words) and common lexis of chosen background features. Plain terms *background* and *foreground* in this

article represent sets of similarity indices outside the groups and inside the groups correspondingly.

Foreground similarity can be also used as a determiner for the most optimal number of clusters for a chosen scenario. Most clustering algorithms do not have any effective rules regarding the number of clusters and it is advised not to impose them but rather select the number of clusters manually. If the distinction between background noise and inner similarity is strict enough and their individual pair values do not deviate from means too much, then the PAM structuring by distances will work most efficiently with the right number of classes. Thus it may allow to automatically select the best number of clusters by choosing the division with the biggest resulting foreground similarity. If the supposed division number is not the optimal one, then there is an issue either with the feature's elements division or there is some other unknown stronger feature in play.

3.2. Scenario 1 – Absolute similarities

The first research question of this article asks whether the comparison between lexis fields allows spotting differences in their features at all. It is tested by comparing the similarity between mostly unrelated books with the similarity between exceptionally similar books. To do this the program chooses the best possible conditions to reach both practical absolute coefficient values.

To reach the practical maximum, books which have identical features have to be selected. While it seems to be impossible, there is a way to achieve this by breaking up one book into two or more parts. These parts will have the same characteristics as the whole book.

To reach the practical minimum value, a set of books with a minimal number of common features has to be selected. However, there is at least one common feature in this set of books – a mutual type of literature. Secondly, this feature may be much stronger than any other feature and thus it has to be examined separately. Due to an unknown number of probable divisions, it is very hard to assure that there are no other common characteristics between chosen books.

But even with the common type of literature and some other unknown features this scenario still has a small background feature similarity compared to the highest achievable foreground feature similarity. If the degree of lexical differences is greater or roughly equal between two parts of one book than to a completely unrelated book, then this method does not work, because it is probably the biggest possible feature distance between background and foreground.

Considering how publishers and authors break up their works into volumes and parts this scenario can use these two division methods and analyse every volume as a separate text. Books chosen are the two volumes of B. Prus's "Lalka" (99% and 98% coverage), four volumes of Reymont's "Chłopi" and all three entries in H. Sienkiewicz's Trilogy – "Ogniem_i_mieczem" (2 vol.), "Pan Wołodyjowski" (1 vol.) and "Potop" (3 vol.). All of these books are fiction books and have different styles, authors and etc.

```
Group 1 (r = 0.505; inner similarity - 0.501):
        books_'Chlopi'_vol._1_-_Reymont (50%)
        books_'Chlopi'_vol._2_-_Reymont (100%)
        books_'Chlopi'_vol._3_-_Reymont (52%)
        books_'Chlopi'_vol._4_-_Reymont (49%)
Group 2 (r = 0.476; inner similarity - 0.542):
        books_'Ogniem_i_mieczem'_vol._1_-_Sienkiewicz (52%)
        books_'Ogniem_i_mieczem'_vol._2_-_Sienkiewicz (54%)
        books_'Pan_Wolodyjowski'_-_Sienkiewicz (54%)
        books_'Potop'_vol._1_-_Sienkiewicz (58%)
        books_'Potop'_vol._2_-_Sienkiewicz (100%)
        books_'Potop'_vol._3_-_Sienkiewicz (57%)
Group 3 (r = 0.425; inner similarity - 0.575):
        books_'Lalka'_vol._1_-_Prus (100%)
        books_'Lalka'_vol._2_-_Prus (58%)

Background noise similarity - 0.331
Common inner similarity - 0.532
Foreground similarity - 0.202

Group's similarity matrix:
1.000 0.323 0.290
0.323 1.000 0.372
0.290 0.372 1.000

Group's distance matrix:
0.000 0.677 0.710
0.677 0.000 0.628
0.710 0.628 0.000
```

Figure 2: The result of clustering 11 volumes taken from 2 books and one trilogy into 3 groups.

As shown in Figure 3, PAM structuring by Jaccard similarities acquired by dictionary comparison allows to successfully reassemble volumes into books and book series. Each cluster has a width between 0.42 and 0.51, while distances between those groups range between 0.63 and 0.71. Common inner similarity equals 0.53, while the background noise is 0.33, resulting in a 20% index difference between background and foreground.

```
Similarity matrix:
1.00 0.50 0.48 0.47 | 0.29 0.29 | 0.33 0.33 0.32 0.32 0.31 0.31
0.50 1.00 0.52 0.49 | 0.30 0.30 | 0.34 0.35 0.34 0.34 0.33 0.33
0.48 0.52 1.00 0.55 | 0.29 0.28 | 0.33 0.33 0.32 0.31 0.31 0.31
0.47 0.49 0.55 1.00 | 0.29 0.29 | 0.33 0.33 0.32 0.32 0.31 0.31
0.29 0.30 0.29 0.29 | 1.00 0.58 | 0.37 0.38 0.39 0.38 0.38 0.36
0.29 0.30 0.28 0.29 | 0.58 1.00 | 0.36 0.37 0.38 0.38 0.37 0.35
0.33 0.34 0.33 0.33 | 0.37 0.36 | 1.00 0.58 0.53 0.53 0.52 0.52
0.33 0.35 0.33 0.33 | 0.38 0.37 | 0.58 1.00 0.54 0.54 0.54 0.53
0.32 0.34 0.32 0.32 | 0.39 0.38 | 0.53 0.54 1.00 0.53 0.54 0.54
0.32 0.34 0.31 0.32 | 0.38 0.38 | 0.53 0.54 0.53 1.00 0.58 0.53
0.31 0.33 0.31 0.31 | 0.38 0.37 | 0.52 0.54 0.54 0.58 1.00 0.57
0.31 0.33 0.31 0.31 | 0.36 0.35 | 0.52 0.53 0.54 0.53 0.57 1.00
```

Figure 3: Similarity matrix and division of three groups. The yellow part shows "Chlopi" volumes, green - "Lalka" volumes and blue - Sienkiewicz's Trilogy.

The symmetric similarity matrix (see Figure 4) shows individual indices for every pair of books in this scenario. Background sectors are uncoloured and individual foreground groups are uniquely coloured. Heatmap provides a visualisation of Values (see Figure 5) in both background and foreground are stable and their mean representations show the accurate picture. Heatmap also shows different static background intensities for every pair of classes.

Given the results, it is certain that there is a connection between features and lexis, represented by a distinct 20% difference between minimally and maximally feature-related books. Static background noise for mainly

unrelated fiction books ranges from 0.28 to 0.39 with mean of 0.33. Foreground values are also static and surprisingly do not range too much in different groups – only by 8%. That implies that differences between book volumes and entries of book series have the same degree.

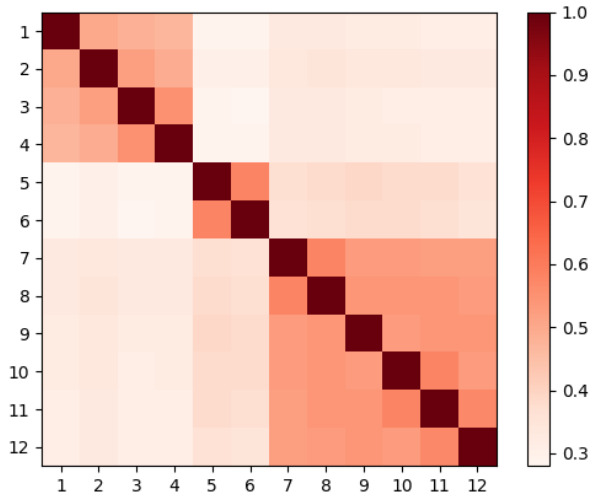


Figure 4: Similarity matrix heatmap.

3.3. Scenario 2 – Author differentiation

The second question is whether this tendency of starker lexical likeness remains between the works of one author compared to the works written by other authors. This scenario targets supposedly the heaviest factor after literature type which influences the book's lexis. To examine this factor three classic Polish authors were selected – Bolesław Prus (7 books), Władysław Reymont (9 books) and Henryk Sienkiewicz (10 books). H. Sienkiewicz's books include all three books of his Trilogy as a reference to absolute levels of similarity. W. Reymont's books also include the "Rok 1794" trilogy, which probably exhibits similar behaviour.

Clustering algorithms mainly recreated supposed division (see Figure 6) but misplaced three of Reymont's books and one Sienkiewicz's book into Prus's cluster. While background noise remains exactly the same (33%) the foreground equals only 40% thus 13% shorter than the practical maximum acquired in the first scenario (53%). It is obvious that the anomaly occurred inside Reymont's cluster, which has only 36,5% compared to 40-42% of two other clusters.

Differences shown in the heatmap (see Figure 6) are not as distinct as in scenario 1 though the background has similar values. An excerpt of high values inside Sienkiewicz's Trilogy (positions 19-21) contrasts with his other books (17-26), which in turn is bordered by smaller background values. Higher values also separate Prus's books (1-7) from the background, but it seems that the three next books (8-10) which belong to Reymont are also part of this cluster. Recreated Prus's cluster holds all these books and also one book written by Reymont (17). All other Reymont's books have an anomaly causing much lower background (less than 30%) and even foreground values around them. The only exceptions are between books from the "Rok 1794" trilogy. Background values between Prus's and Sienkiewicz's clusters are higher (36%) and are harder to separate from their inner similarities (40-42%).

```
Group 1 (r = 0.613; inner similarity - 0.402):
        books_'Dusze_w_niewoli'_-_Prus (43%)
        books_'Anielka'_-_Prus (100%)
        books_'Placowka'_-_Prus (42%)
        books_'Lalka'_-_Prus (46%)
        books_'Emancypantki'_-_Prus (46%)
        books_'Faraon'_-_Prus (40%)
        books_'Dzieci'_-_Prus (42%)
        books_'Komediantka'_-_Reymont (39%)
        books_'Fermenty'_-_Reymont (40%)
        books_'Ziemia_obiecana'_-_Reymont (40%)
        books_'Na_marne'_-_Sienkiewicz (40%)
Group 2 (r = 0.629; inner similarity - 0.421):
        books_'Za_chlebem'_-_Sienkiewicz (37%)
        books_'Ogniem_i_mieczem'_-_Sienkiewicz (56%)
        books_'Pan_Wolodyjowski'_-_Sienkiewicz (100%)
        books_'Potop'_-_Sienkiewicz (58%)
        books_'Bez_dogmatu'_-_Sienkiewicz (37%)
        books_'Rodzina_Polanieckich'_-_Sienkiewicz (42%)
        books_'Quo_vadis'_-_Sienkiewicz (45%)
        books_'Krzyzacy'_-_Sienkiewicz (52%)
        books_'W_pustyni_i_w_puszczy'_-_Sienkiewicz (43%)
Group 3 (r = 0.672; inner similarity - 0.365):
        books_'Chlopi'_-_Reymont (34%)
        books_'Wampir'_-_Reymont (35%)
        books_'Ostatni_sejm_Rzeczypospolitej'_-_Reymont (53%)
        books_'Nil_desperandum'_-_Reymont (100%)
        books_'Insurekcya'_-_Reymont (47%)
        books_'Bunt'_-_Reymont (33%)

Background noise similarity - 0.332
Common inner similarity - 0.403
Foreground similarity - 0.071

Group's similarity matrix:
1.000 0.364 0.303
0.364 1.000 0.309
0.303 0.309 1.000

Group's distance matrix:
0.000 0.636 0.697
0.636 0.000 0.691
0.697 0.691 0.000
```

Figure 5: The result of clustering 26 books into 3 classes.

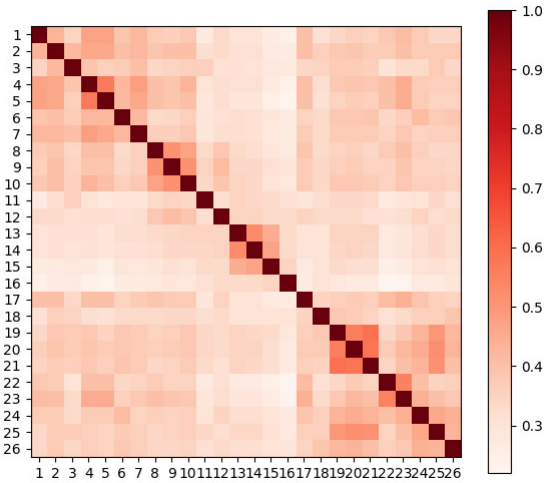


Figure 6: Similarity matrix heatmap. 1 to 7 are Prus's books, 8 to 16 – Reymont's books and 17 to 26 are Sienkiewicz's books.

Overall while Prus's and Reymont's clusters have predictable values, which are at least 4% higher inside them, Reymont's works display a much stronger distance between themselves and to every other book, excluding his early works (8: "Komediantka", 9: "Fermenty", 10: "Ziemia obiecana") which still obey aforementioned tendencies. It may be caused by Reymont's ability to change his style and lexis selection according

to his needs resulting in nearly background-level likeness between his books or just by the use of enormous vocabulary replacing a lot of more commonplace synonyms which causes anomalously low background levels. Anyway W. Reymont is known for his thematically, literary and valuably diverse works.

3.4. Scenario 3 – Types of literature differentiation

The third research question reviews one of the most popular beliefs about language reading and comprehension difficulty. Non-fiction literature, especially academic one is believed to be much more demanding to its reader due to a great amount of subject-specific terminology and more sophisticated academic or professional writing.

```
Group 1 (r = 0.790; inner similarity - 0.223):
        books_law_'Kodeks_wyborczy' (22%)
        books_law_'Kodeks_morski' (100%)
        books_law_'Kodeks_pracy' (27%)
        books_law_'Kodeks_wykroczen' (21%)
        books_law_'Kodeks_karny' (21%)
        books_law_'Kodeks_cywilny' (31%)
Group 2 (r = 0.693; inner similarity - 0.327):
        books_'Ziemia_obiecana'_-_Reymont (100%)
        books_'Emancypantki'_-_Prus (40%)
        books_'Quo_vadis'_-_Sienkiewicz (36%)
        books_'Popioly'_-_Zeromski (34%)
        books_'Komornicy'_-_Orkan (31%)
        books_'Prochno'_-_Berent (36%)
Group 3 (r = 0.812; inner similarity - 0.184):
        books_textbooks_'Metody_numeryczne_w_przykladach' (100%)
        books_textbooks_'Fizyka' (19%)
        books_textbooks_'Podstawy_programowania' (19%)
        books_textbooks_'Zastosowanie_programu_Mathcad' (21%)

Background noise similarity - 0.109
Common inner similarity - 0.260
Foreground similarity - 0.151

Group's similarity matrix:
1.000 0.110 0.110
0.110 1.000 0.105
0.110 0.105 1.000

Group's distance matrix:
0.000 0.890 0.890
0.890 0.000 0.895
0.890 0.895 0.000
```

Figure 7: The result of clustering fiction and two types of non-fiction literature into three separate classes.

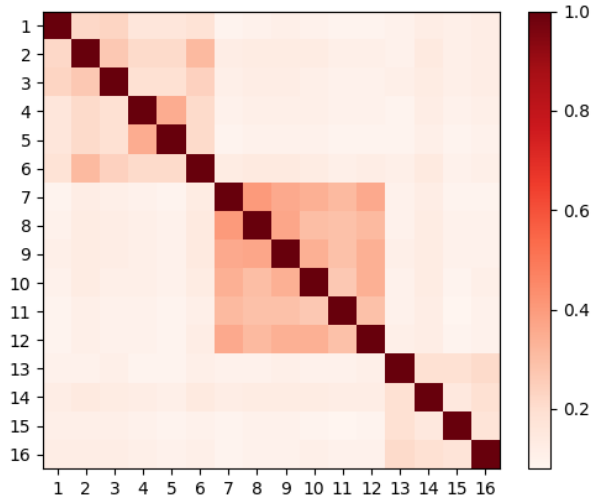


Figure 8: Similarity matrix heatmap. 1 to 6 are codes of law, 7 to 12 – fiction books and 13 to 16 are textbooks.

This scenario compares three sets of books – namely six works of legal literature, six fiction books and four textbooks. All of these books have diversified features, such as different authors or different fields for textbooks.

As shown in Figure 7 clustering had successfully recreated the supposed three groups with 15% of foreground similarity. Inner similarities are falling from 30% inside fiction books through 22% between legal documents down to 18% for textbooks but stay significantly higher than static 11% background noise. These conclusions can be drawn also from heatmap visualisation in Figure 8.

4. Conclusions

The research concluded in this work indicates that lexical similarity can be used to state "feature" distance, starting from the type of literature, author and ending with genres and styles. Thus the compact hypothesis for books' classes and features is proved.

The first scenario analyses how the greatest and the smallest achievable values for the similarity between two arbitrary fiction books correlate with nonexistent and absolute feature similarity. Background for nonexistent similarity is 33% when inner similarity goes above 50%.

The second scenario that evaluates books written by different authors shows that in most cases their works can be recognised by more than 5% of foreground similarity. This relation works unless these authors specifically use much more distinct lexis for every other work as W. Reymont did.

The third scenario shows a successful attempt to separate fiction books, textbooks and legal literature apart. The practically featureless background shows an astonishing 11% similarity which breaks the practical minimum achieved in the first scenario by 22%. It also displays significant differences in similarity coefficients inside these three classes.

Fiction books appear to have a much bigger tendency to share common lexis. Also, it is possible to differentiate types of further divisions inside this group.

Practical absolute similarity values found in this research are 11% for absent feature similarity (including the type of literature) and 53% for an identical set of features. Thus there is a 42% of similarity coefficient range available for research.

The solution presented in this article allows to do basic analysis concerning most outstanding features, but many steps taken in this article can be refined for better results including a more precise dictionary with a better lemma selection procedure and more appropriate similarity coefficient. This is not limited to the Polish language and requires only a change of dictionary complete with declensions and respective lemmas to analyse texts in other languages. Lexis-feature relation appears to be a promising field for further thorough research.

References

[1] R. Singh, S. Singh, Text Similarity Measures in News Articles by Vector Space Model Using NLP, Journal of The Institution of Engineers (India): Series B 102 (2021) 329–338.

[2] A. Huang, Similarity Measures for Text Document Clustering, Proceedings of the Sixth New Zealand Computer Science Research Student Conference 4 (2008) 49–56.

[3] M. B. Magara, S. O. Ojo, T. Zuva, A Comparative Analysis of Text Similarity Measures and Algorithms in Research Paper Recommender Systems, 2018 Conference on Information Communications Technology and Society (2018) 1–5.

[4] A. W. Qurashi, V. Holmes, A. P. Johnson, Document Processing: Methods for Semantic Text Similarity Analysis, In 2020 International Conference on INnovations in Intelligent SysTems and Applications (2020) 1–6.

[5] W. H. Gomaa, A. A. Fahmy, A Survey of Text Similarity Approaches, International Journal of Computer Applications 68 (2013) 13–18.

[6] S. Bekmirzaev, T. H. Kim, B. C. Lee, Pairwise Similarity Analysis and Quality Estimation on Classical Chinese Poetry of Ancient Korea in 15th Century, International Journal of Applied Engineering Research 12 (2017) 13884–13890.

[7] D. M. Kaplan, D. M. Blei, A Computational Approach to Style in American Poetry, In Seventh IEEE International Conference on Data Mining (2007) 553–558.

[8] C. D. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, MIT press, 1999.

[9] R. Grishman, Computational Linguistics: An Introduction, Cambridge University Press, 1986.

[10] R. Grzegorczykowa, R. Laskowski, H. Wróbel, Gramatyka współczesnego języka polskiego. Morfologia, Wydawnictwo Naukowe PWN, 1999.

[11] S. Niwattanakul, J. Singthongchai, E. Naenudorn, S. Wanapu, Using of Jaccard Coefficient for Keywords Similarity, In Proceedings of the International Multiconference of Engineers and Computer Scientists 1 (2013) 380–384.

[12] L. Kaufman, P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 2009.

[13] Słownik języka polskiego, https://sjp.pl, [18.09.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 324–329
Received: 10 September 2021
Accepted: 10 October 2021

Comparison of capabilities of the Unity environment and LibGDX in terms of computer game development

Porównanie możliwości środowiska Unity oraz LibGDX w kontekście tworzenia gier komputerowych

Piotr Kosidło*, Karol Kowalczyk*, Marcin Badurowicz

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

As part of the work on the article, two 2D games were created – one based on the Unity environment and the other based on LibGDX. Main focus in the work was to compare the performance of both games. For this purpose, research was carried out to determine which game has a better impact on the usage of CPU and RAM resources. Attention was also paid to community support for both tools and the programmer's comfort during the work in both of these tools. The results of the performance studies suggest that LibGDX may be a better choice for creating small projects where performance is a priority. However, the support of the community and the comfort of working with the environment and the lack of need to use external programs speak in favor of Unity.

Keywords: Unity; LibGDX; computer games

Streszczenie

W ramach pracy nad artykułem stworzone zostały dwie gry 2D - jedna przy użyciu środowiska Unity oraz druga przy użyciu LibGDX. Szczególną uwagę w pracy poświęcono porównaniu wydajności obu gier. W tym celu przeprowadzono badania, które miały na celu określenie, która z gier ma lepszy wpływ na zużycie zasobów procesora oraz pamięci RAM. Poświęcono również uwagę wsparciu społeczności dla obu narzędzi oraz komfortowi programisty podczas pracy w obu wspomnianych narzędziach. Wyniki badań wydajności sugerują, że LibGDX może być lepszym wyborem do tworzenia niewielkich projektów, których priorytetem jest wydajność. Na korzyść Unity przemawia jednak wsparcie społeczności oraz komfort korzystania z tego środowiska i brak konieczności korzystania z programów zewnętrznych.

Słowa kluczowe: Unity; LibGDX; gry komputerowe

*Corresponding author

Email address: piotr.kosidlo@pollub.edu.pl (P. Kosidło), karolkow1995@gmail.com (K. Kowalczyk)

1. Wstęp

Rynek gier komputerowych w dzisiejszych czasach osiąga ogromne rozmiary. Obecnie najpopularniejsze oraz najbardziej rozbudowane produkcje tworzone są przez studia zatrudniające setki pracowników, jednak do stworzenia interesującej produkcji nie jest wymagany duży zespół programistów. Rozwijający się w ostatnich latach segment "Indie" pozwolił małym studiom, a także pojedynczym deweloperom na znalezienie własnej niszy, w której mogą zyskiwać popularność.

Obecnie na rynku można znaleźć wiele narzędzi ułatwiających projektowanie oraz tworzenie gier komputerowych. Wybór jednego z nich może być uwarunkowany różnymi czynnikami. Jednym z takich czynników z pewnością jest znajomość języków programowania. Większość środowisk służących do tworzenia gier komputerowych jest zorientowana na jeden konkretny język programowania. Java oraz C# są jednymi z najbardziej popularnych języków [1], szczególnie w kontekście programowania gier komputerowych. Dlatego też artykuł skupia się na narzędziach, które można zastosować mając doświadczenie w pracy ze wspomnianymi językami programowania.

W pracy nad niniejszym artykułem wykorzystano dwa narzędzia – Unity oraz LibGDX. Korzystając z nich utworzono dwie gry komputerowe – jedna napisana została w środowisku Unity, a druga w LibGDX. Na ich podstawie przeanalizowany został wpływ zastosowanego narzędzia na wydajność gry oraz zużycie zasobów sprzętowych. W pracy zwrócono również uwagę na funkcjonalności oferowane przez oba narzędzia. Niektórzy programiści przy wyborze technologii mogą kierować się tym czy dane środowisko zapewnia wszystkie funkcje potrzebne do zaprogramowania gry, czy jednak konieczne w tym celu jest wykorzystanie programów zewnętrznych. Kwestia ta wpływa na komfort programisty w procesie tworzenia gry, co może skutkować gorszą jakością końcowego projektu.

Wydajność tworzonych gier komputerowych jest szczególnie ważna ze względu na mnogość konfiguracji sprzętowych dostępnych na rynku. Istotną kwestią jest to, aby gry były możliwe do uruchomienia na jak największej liczbie konfiguracji sprzętowych.

2. Technologie

2.1. LibGDX

LibGDX jest frameworkiem pozwalającym na tworzenie gier komputerowych oraz wizualizacji w języku Java. Jest to narzędzie wieloplatformowe – umożliwia ono skompilowanie jednokrotnie napisanego kodu na

kilka różnych platform. Pozwala on na tworzenie aplikacji, które można uruchomić na systemach: Windows, Linux, macOS, Android, iOS oraz HTML5 [2] przy pomocy tego samego kodu napisanego w języku Java. LibGDX wspiera najnowsze wersje OpenGL. Framework ten pozwala na tworzenie gier zarówno 2D jak i 3D.

LibGDX jest dosyć kompaktowym frameworkiem i nie oferuje wbudowanych funkcji i narzędzi, które pozwalają na np. tworzenie map. W LibGDX konieczne jest użycie zewnętrznego programu w celu stworzenia mapy, która będzie renderowana w grze. Framework ten oferuje generator, który pozwala na wygenerowanie projektu zawierającego odpowiednie biblioteki. W generatorze można również wybrać platformy na jakie zostanie skompilowany kod oraz dodatkowe rozszerzenia.

2.2. Unity

Unity jest środowiskiem pozwalającym na tworzenie wieloplatformowych gier komputerowych, wizualizacji oraz animacji. Gry tworzone przy użyciu Unity mogą być uruchamiane na wielu różnych systemach m.in. Windows, Linux, macOS, iOS oraz Android, a także na konsolach siódmej i ósmej generacji. Silnik ten pozwala na tworzenie gier 2D oraz 3D. Językiem stosowanym do programowania gier komputerowych w Unity jest C#.

Środowisko Unity oferuje wiele narzędzi takich jak Unity Editor. Jest w nim dostępnych wiele przydatnych wbudowanych funkcji takich jak Scene View, Tile Palette czy Asset Store [3]. Scene View pozwala na komfortowe tworzenie gry – możliwe jest dodawanie i pozycjonowanie obiektów takich jak obiekt gracza, obiekty przeciwników, elementów otoczenia czy podłoża, po którym można się poruszać w grze. Tile Palette pozwala na wybór poszczególnych tekstur spośród plików projektu w celu użycia ich w grze. Dzięki temu narzędziu możliwe jest renderowanie tekstur i przypisywanie ich do poszczególnych obiektów. Sposób jego użycia jest komfortowy i przypomina korzystanie z programu służącego do modyfikacji plików graficznych. Asset Store natomiast pozwala na wybieranie komponentów takich jak tekstury mapy oraz pobieranie ich i dołączanie do plików projektu w celu użycia ich w trakcie tworzenia gry [4].

3. Implementacja

W ramach pracy nad artykułem napisano dwie gry – jedna napisana została przy pomocy LibGDX, a druga w środowisku Unity. Zdecydowano się na stworzenie możliwie jak najbardziej podobnych do siebie gier pod względem graficznym jak i funkcjonalnym. W tym celu w obu grach wykorzystano te same tekstury gracza, przeciwników, obiektów oraz otoczenia. Dzięki takiemu podejściu porównanie wydajności gier mogło być bardziej miarodajne niż w przypadku, gdy gry znacznie różniłyby się od siebie.

Obie gry są grami platformowymi. Gracz ma za zadanie poruszać się po ziemi i platformach w celu dostania się na koniec mapy, co oznacza wygraną. Gracz po drodze zbiera wiśnie – zebranie jednej z nich powoduje dodanie punktu na konto gracza. Przeszkodą dla gracza są cztery postaci wrogów – kolizja z nimi oznacza przegraną. W międzyczasie zliczany jest czas trwania jednego podejścia.

3.1. Gra napisana w LibGDX

Do zaimplementowania gry opartej na LibGDX użyto środowiska IntelliJ IDEA w wersji Community Edition. Gra została napisana w języku Java oraz skompilowana do wykonywalnego pliku JAR przy pomocy Java Development Kit w wersji 8. Do zainicjalizowania projektu użyto generatora oferowanego przez LibGDX, który pozwala na dołączenie potrzebnych plików i bibliotek, tak aby od razu po zaimportowaniu projektu w wybranym środowisku programistycznym można było zacząć pracę nad grą.

Jako że LibGDX jest jedynie kompaktowym frameworkiem, podczas pracy nad projektem istnieje potrzeba korzystania z programów zewnętrznych takich jak Tiled. Program ten pozwala na tworzenie map, które później można renderować w grze. Pozwala on na tworzenie tzw. Tilesetów bazując na plikach graficznych, które można zaimportować. Dzięki wspomnianym Tilesetom możliwe jest wybieranie poszczególnych części zaimportowanego pliku graficznego i umieszczanie ich w odpowiednich miejscach w celu stworzenia mapy. Cały proces przypomina korzystanie z narzędzi do rysowania dostępnych w większości programów służących do modyfikacji plików graficznych. Gotowy plik o rozszerzeniu .tmx można później wykorzystać w projekcie gry. Wszystkie klasy powiązane z mapami można znaleźć w pakiecie com.badlogic.gdx.maps [5]. W pracy do załadowania pliku .tmx oraz renderowania mapy użyto trzy obiekty klas TmxMapLoader, TiledMap i OrthogonalTiledMapRenderer. Obiekt klasy TmxMapLoader pozwala na załadowanie mapy z pliku tmx dzięki metodzie load() oraz na zainicjalizowanie obiektu TiledMap. Obiekt TiledMap w następnej kolejności zostaje użyty do zainicjalizowania obiektu OrthogonalTiledMapRenderer, który później można użyć do renderowania mapy. Jest to możliwe poprzez wywołanie metody render() należącej do wspomnianej klasy.

Innym programem zewnętrznym, którego użyto podczas pracy nad grą był Texture Packer [6]. Posłużył on do złączenia tekstur wszystkich postaci w jeden plik .png, oraz wygenerowania tekstowego pliku z rozszerzeniem .pack. Plik ten posłużył później do renderowania tekstur postaci, a także wyświetlania animacji poruszania się dla postaci gracza.

Istotnym interfejsem dostępnym w LibGDX jest interfejs Screen. Zawiera on metody konieczne do poprawnego działania gry. Klasa, która implementuje ten interfejs służy do określenia m.in. tego co ma być renderowane na ekranie gry. Prawdopodobnie najważniejszą metodą należącą do tego interfejsu jest metoda render(). Jest ona wywoływana w trakcie wyświetlania każdej klatki w grze. To dzięki tej metodzie możliwe było zaprogramowanie takich rzeczy jak renderowanie mapy czy poruszanie się graczem.

Jedną z najważniejszych bibliotek LibGDX użytych do napisania gry była biblioteka Box2D [7]. Posłużyła ona do zaimplementowania fizyki w grze. Dzięki niej możliwe było nadanie obiektom z gry takich właściwości jak pozycja, masa czy prędkość. Umożliwiła ona detekcję kolizji między obiektami w grze. Dzięki temu można wykrywać kolizję między np. postacią gracza, a przeciwnikiem i podjąć odpowiednie operacje po takiej kolizji – w tym przypadku zrestartować grę.

Rysunek 1: Zrzut ekranu z gry napisanej w LibGDX.

3.2. Gra napisana w Unity

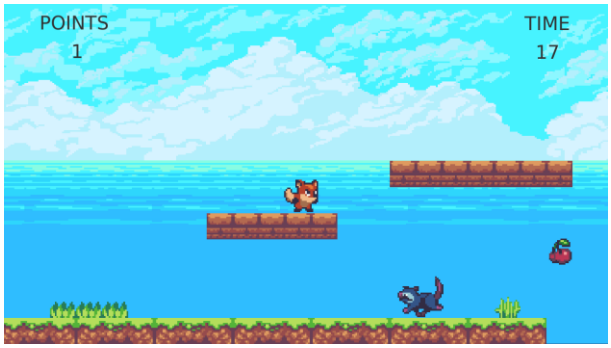
W celu napisania gry użyto środowiska Unity. Przy użyciu Unity Hub stworzono nowy projekt gry 2D. Całość prac wymagała użycia Unity Editora oraz Visual Studio Code bez konieczności korzystania z innych zewnętrznych programów. Gra została napisana w języku C#.

Stworzenie mapy było możliwe dzięki narzędziom udostępnianym przez Unity Editor. Domyślnie przy tworzeniu projektu otwarty jest Scene View oraz utworzona zostaje scena. Gra może składać się z kilku scen [8], jednak w grze stworzonej na potrzeby niniejszego artykułu zastosowano pojedynczą scenę. W tym przypadku wspomniana scena zawiera wszystkie obiekty, które znajdują się w grze – obiekt postaci gracza, przeciwników, kamery czy elementów otoczenia. Jednym z najważniejszych obiektów w kontekście tworzenia mapy był Grid. Dodany został do niego komponent Grid [9], który pozwala na podział sceny na siatkę kwadratów, dzięki czemu tworzenie mapy jest dużo bardziej przejrzyste i komfortowe. Wewnątrz obiektu Grid zawarte są warstwy – Tilemapy. Każda taka warstwa jest obiektem, który może mieć nadaną konkretną wartość Order in Layer w komponencie Tilemap Renderer, dzięki czemu można określić priorytet danej warstwy podczas procesu renderowania sceny. Wszystkie obiekty Tilemap zastosowane w grze mają kształt prostokątów. W grze stworzono oddzielną warstwę zawierającą tekstury ziemi, oddzielną warstwę zawierającą tekstury cegieł, oddzielną warstwę dla tła oraz obiektów takich jak wiśnie. Ma to później zastosowanie w wykrywaniu kolizji oraz jest istotne pod względem wspomnianych wcześniej priorytetów warstw.

W Unity część gry można wykonać bez pisania kodu, jednak do zaprogramowania takich rzeczy jak poruszanie się, czy detekcja kolizji, napisanie skryptów było niezbędne. W pracy konieczne było napisanie ośmiu skryptów dotyczących różnych aspektów gry takich jak zliczanie czasu i punktów, poruszanie się przeciwników i wykrywanie kolizji z nimi, czy zaprogramowanie sposobu poruszania się gracza. Każdy skrypt wygenerowany w Unity Editorze domyślnie posiada dwie metody dziedziczone z klasy MonoBehaviour – Start() oraz Update(). Metoda Start() jest wywoływana jednokrotnie podczas działania gry – jeszcze przed wyświetleniem pierwszej klatki [10], natomiast metoda Update() wywoływana jest podczas renderowania każdej kolejnej klatki. W związku z tym metoda Start() najczęściej służy inicjalizacji obiektów używanych w skrypcie, a metoda Update() np. do sprawdzania na bieżąco czy w grze zaszło konkretne zdarzenie.

Zaimplementowanie detekcji kolizji było możliwe dzięki komponentom Tilemap Collider 2D, Box Collider 2D oraz Capsule Collider 2D. Komponent Tilemap Collider 2D został dodany do obiektów reprezentujących poszczególne warstwy mapy takich jak podłoże czy cegły. Box Collider 2D został dodany do obiektów przeciwników, a Capsule Collider 2D został dodany do obiektu gracza. Do zaimplementowania detekcji kolizji kluczowe były metody: OnTriggerEnter2D() oraz OnTriggerExit2D(). Pierwsza metoda wywoływana jest, gdy jeden obiekt wchodzi w kontakt z drugim obiektem [11]. Oba obiekty muszą mieć dołączony komponent z grupy Collider 2D, np. Capsule Collider 2D. Metoda OnTriggerExit2D() jest natomiast wywoływana wtedy, gdy obydwa obiekty tracą ze sobą kontakt. Metoda OnTriggerExit2D() została wykorzystana m.in. do obsługi kolizji pomiędzy graczem, a punktem końcowym mapy. Zostało to zaimplementowane w taki sposób, że jeśli gracz wchodzi w kontakt z dowolnym obiektem zawierającym komponent Collider 2D, to sprawdzane jest czy obiekt ten zawiera odpowiednio ustawiony tag – musi on mieć treść „House". Jeśli obiekt, z którym gracz wszedł w kontakt nie zawiera tegoż tagu, to nic się nie dzieje. Jeśli jednak dany obiekt zawiera wspomniany tag, to następuje koniec gry. Wyłącznie obiekt reprezentujący punkt końcowy mapy zawiera tag „House", więc koniec gry następuje tylko wtedy, gdy gracz wchodzi w kontakt z tym obiektem.

Do wykonania animacji posłużono się narzędziem Animation dostępnym w edytorze Unity. Z plików projektu wybrane zostały odpowiednie tekstury gracza i utworzono z nich dwa pliki animacji o rozszerzeniu „.anim". Animacje działają podczas poruszania się gracza oraz w momencie, gdy stoi on w jednym miejscu.



Rysunek 2: Zrzut ekranu z gry napisanej w Unity.

4. Metoda badawcza

Obie gry opisane w punkcie 3 zostały przebadane pod kątem wydajności na maszynie wirtualnej VMware Workstation. Badanym aspektem był wpływ gry na zużycie zasobów procesora oraz pamięci RAM. Badania zostały przeprowadzone na dwóch systemach operacyjnych zainstalowanych na maszynie wirtualnej – Windows 10 oraz Ubuntu 20.04.3, w celu określenia różnic w wydajności obu gier w zależności od systemu operacyjnego. Dla obu ze wspomnianych systemów wybrano te same cztery kombinacje przydzielonych zasobów sprzętowych, w celu przetestowania gier nie tylko na różnych systemach, ale również na różnych ustawieniach danej maszyny wirtualnej. Dokładne wartości przydzielonych zasobów opisane są w punkcie 4.1.

Badania wykonywane były najpierw na systemie Ubuntu, a w dalszej kolejności na Windows 10. Oba systemy były uruchamiane na każdej z czterech kombinacji zasobów, a następnie włączane były gry. W pierwszej kolejności badana była wydajność gry napisanej w LibGDX, a następnie gry napisanej w Unity. Badania zostały przeprowadzone z użyciem domyślnego monitora zasobów w obu systemach. Po uruchomieniu gier zapisane zostały wartości zużycia zasobów procesora oraz pamięci RAM. Spisanie tychże wartości z tylko jednego momentu mogłoby być jednak niemiarodajne ze względu na wahania wartości zużycia zasobów w czasie. Z tego względu zdecydowano się na zapisanie dziesięciu kolejnych wartości po każdej ich aktualizacji w monitorze zasobów. Następnie wyliczona została średnia arytmetyczna dla wszystkich takich dziesięciu pomiarów zarówno w przypadku zasobów procesora, jak i pamięci RAM. Istotnym założeniem badań była minimalizacja wpływu postronnych aplikacji na zasoby maszyny. W tym celu podczas przeprowadzania badań zamknięte zostały wszystkie aplikacje działające w tle na maszynie wirtualnej, jak i macierzystej.

4.1. Środowisko testowe

Maszyna wirtualna VMware Workstation została uruchomiona na komputerze o następujących parametrach:

- System operacyjny Elementary OS 5.1.7
- Procesor Intel Core i7-6500U 2.5 GHz
- Pamięć operacyjna DDR3-1600 16 GB
- Karta graficzna AMD Radeon R5 M330
- Dysk twardy SATA III 1 TB

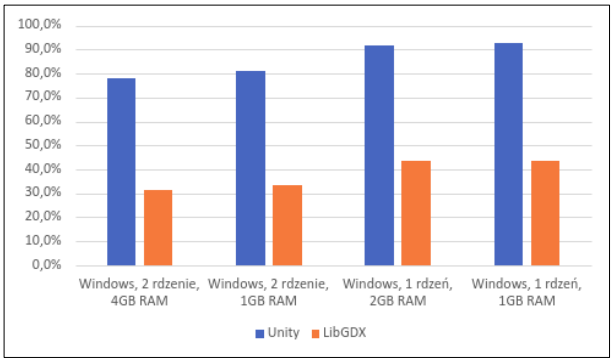
Badania przeprowadzone zostały na systemie Windows 10 oraz Ubuntu 20.04.3. Oba systemy zostały uruchomione na czterech kombinacjach przydzielonych zasobów sprzętowych – liczby rdzeni procesora oraz pamięci RAM. Kombinacje te przedstawione są w poniższej tabeli.

Tabela 1: Wartości przydzielonych zasobów maszyny wirtualnej

Liczba rdzeni CPU	Pamięć RAM (MB)
2	4096
2	1024
1	2048
1	1024

4.2. Wyniki badań

Łącznie przeprowadzonych zostało 16 testów. Obie gry zostały uruchomione na czterech kombinacjach zasobów maszyny wirtualnej na obu systemach operacyjnych – Windows 10 oraz Ubuntu.

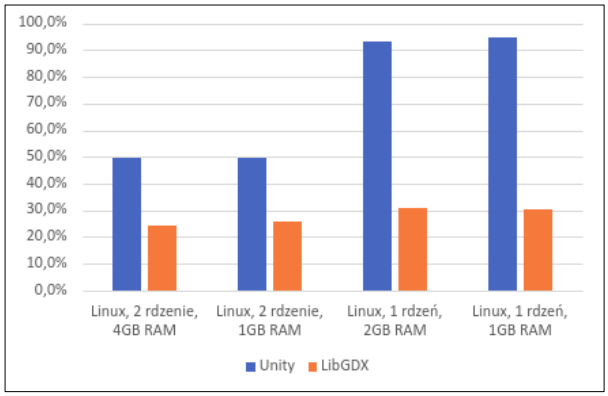


Rysunek 3: Zużycie zasobów procesora podczas badań przeprowadzonych dla obu gier na systemie Windows 10.

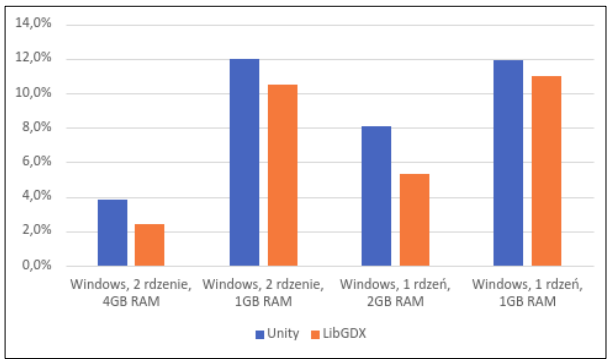
Pierwszą badaną kwestią było zużycie zasobów procesora przez obie gry uruchomione na systemie Windows 10 na czterech różnych konfiguracjach sprzętowych. Jak widać na Rysunku 3 zużycie zasobów procesora było istotnie wyższe w przypadku gry napisanej w Unity niż gry napisanej w LibGDX. Co więcej – jest to widoczne w wynikach testów na każdej z czterech konfiguracji sprzętowych. W każdym z czterech testów gra napisana w LibGDX wypadła lepiej i zużywała ponad dwukrotnie mniej zasobów procesora. Możemy wywnioskować również fakt, że zmiana wartości pamięci RAM przydzielonej dla maszyny wirtualnej nie miała istotnego znaczenia w kontekście zużycia zasobów procesora, natomiast zmniejszenie liczby rdzeni przydzielonych maszynie wirtualnej spowodowało spadek wydajności.

Kolejną badaną kwestią było zużycie zasobów procesora przez obie gry uruchomione na systemie Ubuntu 20.04.3 na czterech różnych konfiguracjach sprzętowych. Jak widać na Rysunku 4 zużycie zasobów procesora było ok. dwukrotnie wyższe w przypadku gry napisanej w Unity, kiedy maszyna wirtualna miała przydzielone dwa rdzenie procesora. W przypadku, gdy maszyna miała przydzielony jeden rdzeń różnica jeszcze bardziej zwiększyła się na niekorzyść gry napisanej w Unity. Oba testy na maszynie z przydzielonym jednym rdzeniem procesora wykazały, że gra napisana w Unity powoduje ok. trzykrotnie większe zużycie zasobów

procesora niż gra napisana w LibGDX. Zmiana wartości pamięci RAM nieznacznie wpłynęła na wyniki badania.



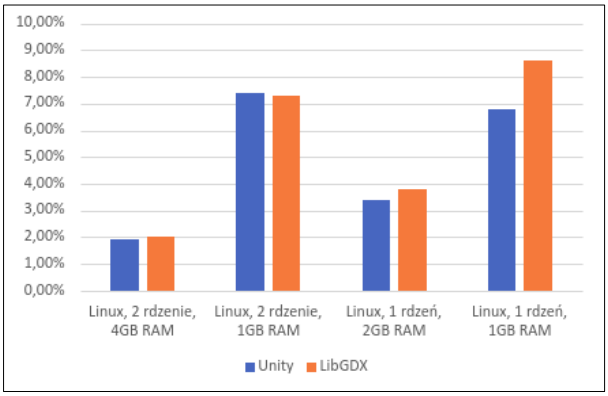
Rysunek 4: Zużycie zasobów procesora podczas badań przeprowadzonych dla obu gier na systemie Ubuntu 20.04.3.



Rysunek 5: Zużycie zasobów pamięci RAM podczas badań przeprowadzonych dla obu gier na systemie Windows 10.

Kolejnym badanym zagadnieniem było zużycie pamięci RAM przez obie gry uruchomione na systemie Windows 10 na czterech różnych konfiguracjach sprzętowych. W przypadku tego badania różnice pomiędzy wydajnością obu gier nie były aż tak duże jak podczas badania zużycia zasobów procesora. Po raz kolejny gra napisana w LibGDX wykazała się lepszą wydajnością, jednak tym razem różnice w zużyciu zasobów pamięci RAM wahały się między jednym, a trzema procentami. Z przeprowadzonych testów można wywnioskować również, że liczba rdzeni przydzielonych maszynie wirtualnej nie miała istotnego wpływu na zużycie zasobów pamięci RAM. Wartość przydzielonej pamięci RAM miała natomiast zdecydowany wpływ na wyniki testów. Zmniejszenie przydzielonych zasobów pamięci spowodowało duży wzrost zużycia zasobów.

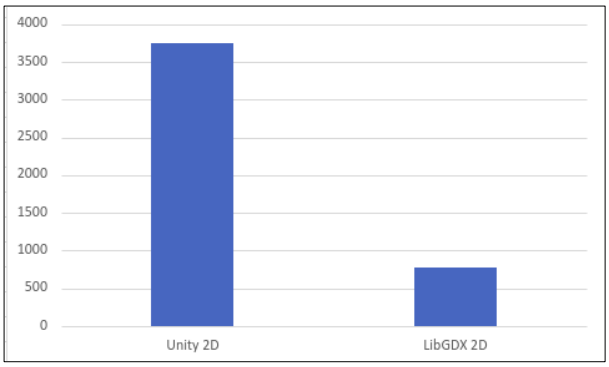
Ostatnie wykonane badanie miało na celu przeanalizowanie zużycia pamięci RAM przez obie gry uruchomione na systemie Ubuntu 20.04.3 na czterech różnych konfiguracjach sprzętowych. Przyniosło ono odmienne rezultaty niż poprzednie badania. Tym razem różnice w wydajności gier były marginalne, a w przypadku testu na maszynie z przydzielonym jednym rdzeniem procesora oraz 1 GB pamięci RAM, to gra napisana w Unity okazała się być nieznacznie lepsza – zużyła o 1.5 punktu procentowego mniej pamięci RAM niż gra napisana w LibGDX. Na podstawie wyników przeprowadzonych badań można było dojść do wniosku, że im mniej przydzielonej pamięci RAM dla maszyny wirtualnej, tym gry zużywały więcej jej zasobów.



Rysunek 6: Zużycie zasobów pamięci RAM podczas badań przeprowadzonych dla obu gier na systemie Ubuntu 20.04.3.

4.3. Wsparcie społeczności

W trakcie pracy nad grami istotną kwestią jest wsparcie społeczności. Jest ono szczególnie ważne dla początkujących programistów, którzy dopiero uczą się danej technologii i mogą poszukiwać rozwiązań swoich problemów w wątkach na tematycznych forach dyskusyjnych. Jednym z najbardziej popularnych serwisów społecznościowych dostępnych dla programistów jest Stack Overflow [12].



Rysunek 7: Porównanie liczby wyników dla Unity 2D i LibGDX na Stack Overflow.

Jak widać na Rysunku 7 liczba rezultatów dla “LibGDX 2D” oraz “Unity 2D” znacząco się różni. Wsparcie społeczności jest większe w przypadku Unity - wynosi 3755 wyników, a w przypadku LibGDX - 770. Wyników dla Unity jest o ok. 388% więcej niż dla LibGDX.

5. Wnioski

Głównym celem prac było zbadanie wpływu gier napisanych w Unity oraz LibGDX na zużycie zasobów komputera. Na podstawie przeprowadzonych badań można dojść do wniosku, że gra napisana w LibGDX spowodowała znacznie niższe zużycie zasobów procesora od gry napisanej w Unity. Było ono ok. dwukrotnie, a nawet trzykrotnie niższe w zależności od danej konfiguracji zasobów maszyny wirtualnej. Zużycie zasobów procesora było niższe w przypadku gry napi-

sanej w LibGDX w każdym teście, na obu systemach oraz na wszystkich konfiguracjach sprzętowych maszyny wirtualnej. W przypadku zużycia pamięci RAM różnica pomiędzy grami nie była aż tak wyraźna. W badaniach przeprowadzonych na systemie Windows 10 gra napisana w LibGDX miała jedynie nieznacznie mniejszy wpływ na zużycie pamięci RAM niż gra napisana w Unity. W wyniku badań przeprowadzonych na systemie Ubuntu 20.04.3 różnica pomiędzy grami została zatarta – obie gry miały bardzo zbliżone zużycie pamięci RAM. W przypadku testu na maszynie z przydzielonym jednym rdzeniem procesora 1 GB pamięci RAM, to gra napisana w Unity okazała się być nieznacznie wydajniejsza w kontekście zużycia pamięci. Obie gry zostały napisane tak, aby były możliwie jak najbardziej zbliżone do siebie pod względem graficznym i funkcjonalnym, a także użyto w nich takich samych plików graficznych i wykonano identyczne animacje. Można więc stwierdzić, że LibGDX może być lepszym wyborem do tworzenia niezbyt skomplikowanych gier platformowych 2D niż Unity, jeśli priorytetem jest wydajność gry. Przeprowadzone badania nie mogą jednak dać jednoznacznej odpowiedzi na to, które ze wspomnianych narzędzi mogłoby mieć przewagę w przypadku tworzenia bardziej skomplikowanych i zaawansowanych gier. Badania przeprowadzone na grach np. 3D napisanych przy pomocy obydwu narzędzi opisywanych w artykule mogłyby zasugerować inne wnioski ze względu na odmienne wyniki zużycia zasobów sprzętowych w środowisku testowym.

Kolejnym badanym aspektem było wsparcie społeczności dla Unity oraz LibGDX. Liczba rezultatów po wyszukaniu obu haseł na serwisie Stack Overflow była znacznie większa w przypadku Unity niż LibGDX. Oznacza to, że programiści korzystający z Unity mają dostępnych dużo więcej potencjalnie pomocnych materiałów w internecie. Jest to szczególnie ważne dla początkujących osób.

Unity okazało się być narzędziem bardziej przyjaznym programiście. W przeciwieństwie do kompaktowego frameworka jakim jest LibGDX, oferuje ono całą gamę przydatnych narzędzi ułatwiających pracę nad grą. W przypadku LibGDX konieczne jest wykorzystanie co najmniej kilku programów zewnętrznych do takich czynności jak tworzenie mapy czy animacji. W przypadku Unity jedynym czego potrzebuje programista jest środowisko oraz dowolny edytor do modyfikowania skryptów. W Unity więcej czynności można wykonać przy użyciu interfejsu graficznego, a w LibGDX praca skupia się głównie na pisaniu kodu. To również może być istotne szczególnie dla początkujących programistów.

Literatura

[1] Statista, https://www.statista.com/chart/21017/most-popular-programming-languages, [03.03.2020].

[2] J. Cook, LibGDX Game Development By Example, Packt Publishing, 2015.

[3] J. Halpern, Developing 2D games with Unity: independent game programming with C#, Apress, 2019.

[4] J. Hocking, Unity In Action: Multiplatform Game Development In C# With Unity 5, Manning Publications 2015.

[5] LibGDX maps documentation, https://github.com/libgdx/libgdx/wiki/Tile-maps, [03.05.2021].

[6] Github - Texture Packer, https://github.com/libgdx/libgdx/wiki/Texture-packer, [10.07.2021].

[7] LibGDX - Box2D documentation, https://github.com/libgdx/libgdx/wiki/Box2d, [15.05.2021].

[8] Unity Scenes - dokumentacja, https://docs.unity3d.com/Manual/CreatingScenes.html, [30.08.2021].

[9] Unity Grid - dokumentacja, https://docs.unity3d.com/Manual/class-Grid.html, [30.08.2021].

[10] Unity - metoda Start(), https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html, [30.08.2021].

[11] Unity - metoda OnTriggerEnter2D(), https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnTriggerEnter2D.html, [30.08.2021].

[12] Stack Overflow, https://stackoverflow.com, [07.09.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 330–335
Received: 12 September 2021
Accepted: 10 October 2021

Performance analysis of the TensorFlow library with different optimisation algorithms

Analiza wydajności biblioteki TensorFlow z wykorzystaniem różnych algorytmów optymalizacji

Maciej Wadas*, Jakub Smołka

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

This paper presents the results of performance analysis of the Tensorflow library used in machine learning and deep neural networks. The analysis focuses on comparing the parameters obtained when training the neural network model for optimization algorithms: Adam, Nadam, AdaMax, AdaDelta, AdaGrad. Special attention has been paid to the differences between the training efficiency on tasks using microprocessor and graphics card. For the study, neural network models were created in order to recognise Polish handwritten characters. The results obtained showed that the most efficient algorithm is AdaMax, while the computer component used during the research only affects the training time of the neural network model used.

Keywords: machine learning; neural networks

Streszczenie

W artykule zaprezentowano wyniki analizy wydajności biblioteki TensorFlow wykorzystywanej w uczeniu maszynowym i głębokich sieciach neuronowych. Analiza skupia się na porównaniu parametrów otrzymanych podczas treningu modelu sieci neuronowej dla algorytmów optymalizacji: Adam, Nadam, AdaMax, AdaDelta, AdaGrad. Zwrócono szczególną uwagę na różnice pomiędzy efektywnością treningu na zadaniach wykorzystujących mikroprocesor i kartę graficzną. Do przeprowadzenia badań utworzono modele sieci neuronowej, której zadaniem było rozpoznawanie znaków języka polskiego pisanych odręcznie. Otrzymane wyniki wykazały, że najwydajniejszym algorytmem jest AdaMax, zaś podzespół komputera wykorzystywany podczas badań wpływa jedynie na czas treningu wykorzystanego modelu sieci neuronowej.

Słowa kluczowe: uczenie maszynowe; sieci neuronowe

*Corresponding author

Email address: maciej.wadas@pollub.edu.pl (M. Wadas)

1. Wstęp

Rozwój sztucznej inteligencji na przestrzeni XXI wieku dowiódł, że istnieje wiele zadań, których wykonanie jest niemożliwe bez ingerencji człowieka. Ludzkość traktuje sztuczną inteligencję jako narzędzie wspomagające ich życie [1]. Obecnie systemy oparte na uczeniu maszynowym otwierają przed ludźmi wiele nowych możliwości, a co za tym idzie wiele nowych problemów i rozwiązań. Na przestrzeni ostatnich lat znaleziono wiele zastosowań sztucznej inteligencji takich jak chatboty, których zadaniem jest replikacja zachowań ludzkich poprzez automatyzację odpowiedzi na często pojawiające się pytania [2]. Implementacja sztucznej inteligencji została zastosowana również w tłumaczeniu języka naturalnego np. w aplikacji Tłumacz Google [3]. Uczenie maszynowe znajduje również zastosowanie w rozpoznawaniu głosu (mowy ludzkiej) czy obrazów [4, 5].

Ze względu na rosnącą liczbę systemów opartych o sztuczną inteligencję na przestrzeni ostatnich lat powstało wiele bibliotek wykorzystywanych w uczeniu maszynowym i głębokich sieciach neuronowych, takich jak SciKit-Learn, Keras, Theano czy TensorFlow [6-9]. Ze względu na rosnącą popularność uczenia maszynowego wielu naukowców z całego świata rozpoczęło badania dotyczące wydajności tego typu technologii. Z czasem uzyskane wyniki rozpoczęły wpływać na rynek informatyczny. Czego skutkiem był wzrost lub spadek popularności wielu narzędzi. Niniejszy artykuł poświęcony jest bibliotece TensorFlow, która jest jedną z najpopularniejszych bibliotek do uczenia maszynowego na rynku [10].

2. Cel badań

Celem badania jest analiza wydajności biblioteki TensorFlow na podstawie parametrów otrzymanych podczas treningu modelu sieci neuronowej. Trening odbędzie się przy pomocy pięciu algorytmów optymalizacji AdaGrad, AdaDelta, Adam, AdaMax, Nadam, dla pojedynczego mikroprocesora (CPU) i karty graficznej (GPU).

Przeprowadzone badania posłużą do zweryfikowania następujących hipotez badawczych:

H1: Najwydajniejszym algorytmem optymalizacji, wśród wybranych, jest Adam.

H2: Urządzenie peryferyjne nie ma wpływu na parametry otrzymane przy treningu modelu.

3. Kategorie uczenia maszynowego

Istnieje wiele różnic między bibliotekami do uczenia maszynowego, jednak w działaniu każdej z nich można wyróżnić powtarzające się schematy postępowania. Widoczne jest to w procesie analizy danych, mającym na celu nauczenie maszyny rozpoznawać pewien wzór występujący w wybranym zestawie danych. Ze względu na to można wyróżnić trzy rodzaje uczenia maszynowego: uczenie nadzorowane, uczenie nienadzorowane, uczenie przez wzmacnianie [11].

W procesie uczenia nadzorowanego (ang. *supervised learning*) uczenie modelu przebiega za pomocą oznakowanych danych uczących (ang. *training data*). Tak wytrenowany model pozwala przewidzieć niewidoczne lub wygenerowane w przyszłości informacje. Człon nadzorowanie odnosi się do zestawu danych wykorzystywanych w treningu, gdzie każdy badany element ma przypisaną etykietę. Przykładowo chcąc wytrenować model danych tak, aby rozpoznawał czy na danym obrazie jest kot czy pies, należy w zestawie oznaczyć, który obraz przedstawia jakie zwierze.

W przypadku pracy z uczeniem przez wzmacnianie (ang. *reinforcement learning*) nie jest wymagane przygotowanie zestawu danych uczących. Celem jest utworzenie systemu (agenta), który poprawia własną skuteczność na podstawie danych otrzymanych w wyniku interakcji ze środowiskiem. Przykładem może być silnik aplikacji gry szachowej. Agent wybiera kolejne ruchy figur na podstawie aktualnego stanu szachownicy (środowiska).

Ostatnim rodzajem uczenia maszynowego jest uczenie nienadzorowane (ang. *unsupervised learning*), gdzie dane treningowe są nieoznakowane lub posiadają nieznaną strukturę. Modele utworzone za pomocą tej techniki uczenia maszynowego pozwalają na poznanie struktury przetwarzanych danych oraz uzyskanie użytecznych informacji bez stosowania znanej zmiennej wyjściowej [11].

4. Architektura biblioteki TensorFlow

TensorFlow jest biblioteką typu open source opracowaną przez firmę Google, która w ciągu ostatnich kilku lat zyskała na popularności. Jest wykorzystywana przez znane firmy takie jak Lenovo, PayPal, inSpace, Intel, Spotify i wiele innych [12]. Platforma TensorFlow cechuje się wieloplatformową strukturą, udostępnia interfejsy API wysokiego poziomu: Keras i Estymator do tworzenia modeli uczenia głębokiego. Na najniższym poziomie operacje TensorFlow są implementowane przy pomocy języka C++ [13].

5. Algorytmy optymalizacji

Biblioteka uczenia maszynowego TensorFlow udostępnia szereg algorytmów optymalizacji. Wybór odpowiedniego ma istotny wpływ na proces uczenia oraz ostateczny uzyskany wynik detekcji na zbiorze walidacyjnym. W niniejszej pracy zostanie porównane pięć algorytmów optymalizacji: Adam, Nadam, AdaMax, AdaDelta, AdaGrad.

Optymalizator Adam jest stochastyczną metodą zejścia gradientowego, która opiera się na adaptacyjnej estymacji momentów pierwszego i drugiego rzędu. Charakteryzuje się tym, że nie przeprowadza optymalizacji funkcji dla wszystkich danych treningowych tylko dla kolejnych partii (ang. *batch*) danych. Ma małe wymagania pamięciowe i jest wydajna obliczeniowo dla problemów dużych pod względem danych [14].

AdaMax jest odmianą algorytmu Adam opartą na normie nieskończoności. Posiada możliwość dostosowania prędkości uczenia do charakterystyki danych, dlatego też często wykorzystywany jest podczas uczenia, gdzie proces jest zmienny w czasie [14].

Algorytm Nadam działa bardzo podobnie do algorytmu Adam. Różnica polega na tym, że Adam postrzegany jest jako połączenie RMSProb i pędu, zaś Nadam łączy algorytm Adam z przyspieszonym gradientem Nesterowa [15].

AdaGrad jest pierwszym algorytmem z rodziny metod gradientowych, dynamicznie wykorzystującym informacje o geometrii danych zaobserwowanych we wcześniejszych iteracjach. Szybkość uczenia algorytmu AdaGrad jest zmienna. AdaGrad utrzymuje wysoki współczynnik uczenia dla rzadko występujących cech, zaś niski w przypadku cech pojawiających się często w zestawie danych [16].

Optymalizator AdaDelta powstał w celu poprawy dwóch głównych wad algorytmu AdaGrad: ciągłego spadku szybkości uczenia się w trakcie treningu oraz konieczności ręcznego wyboru globalnego współczynnika uczenia (ang. *learning rate*) [17].

6. Realizacja badań

Do realizacji badań wymagane jest przygotowanie następujących elementów: środowisko badawcze, zbiór danych treningowych, sieć neuronową, na której podstawie będzie trenowany model. W celu odpowiedzi na hipotezy badawcze zawarte w rozdziale 2 wymagane również jest określenie metody badawczej.

6.1. Środowisko badawcze

W badaniach wykorzystano laptop z 15 calowym wyświetlaczem LCD o rozdzielczości 1920x1080. Standardową myszką i wbudowaną klawiaturą. Procesorem firmy Intel Core i5 oraz kartą graficzną firmy Nvidia. Szczegółową specyfikację urządzenia przedstawiono w Tabeli 1.

Tabela 1: Parametry systemu użytego w badaniach

Pamięć RAM	8 GiB
Procesor	Intel Core i5-6300HQ @ 2,3GHz x 4
Grafika	Nvidia GeForce GTX 960M
Pojemność dysku	1,0 TB
System operacyjny	Ubuntu 20.04.2 LTS

Do pracy z biblioteką zostanie wykorzystane środowisko Docker. Rozwiązanie to zapewnia łatwe urucho-

mienie TensorFlow dla CPU i GPU. W badaniach zostaną wykorzystane obrazy biblioteki Tensorflow w wersji 2.5.0.

6.2. Zbiór danych treningowych

W eksperymencie wykorzystano zestaw danych, w którym zebrano dane o odręcznie pisanych znakach języka polskiego. Dane zawierają:

- cyfry: 0-9,
- małe i wielkie litery alfabetu łacińskiego: A-z
- małe litery alfabetu polskiego: ą, ć, ę, ł, ń, ó, ś, ź, ż,
- wielkie litery alfabetu polskiego: Ą, Ć, Ę, Ł, Ń, Ó, Ź, Ż,
- znaki specjalne: +, -, :, ;, $, !, @.

Zgodnie z dokumentacją dla każdego znaku zebrano co najmniej 6000 obrazów o wielkości 32px (szerokość) na 32px (wysokość) [18]. W pracy wykorzystano małe litery alfabetu polskiego i łacińskiego. Zebrane dane podzielono na zestaw uczący i testowy, gdzie zestaw testowy stanowi 10% całości.

6.3. Model sieci neuronowej

W badaniach wykorzystano model sieci neuronowej, który oparty został o architekturę sieci CNN (ang. *convolutional neural network*) wykorzystanej w artykule [19]. Danymi wejściowymi jest macierz binarna o wymiarach 32x32. Dane wejściowe są propagowane przez dwie warstwy konwolucyjne, które mają kolejno 32 i 64 filtry o rozmiarze 3x3 i rozstępie (ang. *stride*) 1. Otrzymane dane zostają poddane spłaszczeniu i są przepuszczane przez w pełni połączoną warstwę sieci o 256 neuronach. Warstwa wyjściowa jest warstwą sieci w pełni połączonej z funkcją aktywacji Softmax. Dane otrzymane na wyjściu pozwolą na rozpoznawanie znaków ze zbioru danych treningowych. W sieci użyto funkcji straty *categorical cross-entropy*. Schemat powstałej sieci przedstawiono na Rysunku 1.

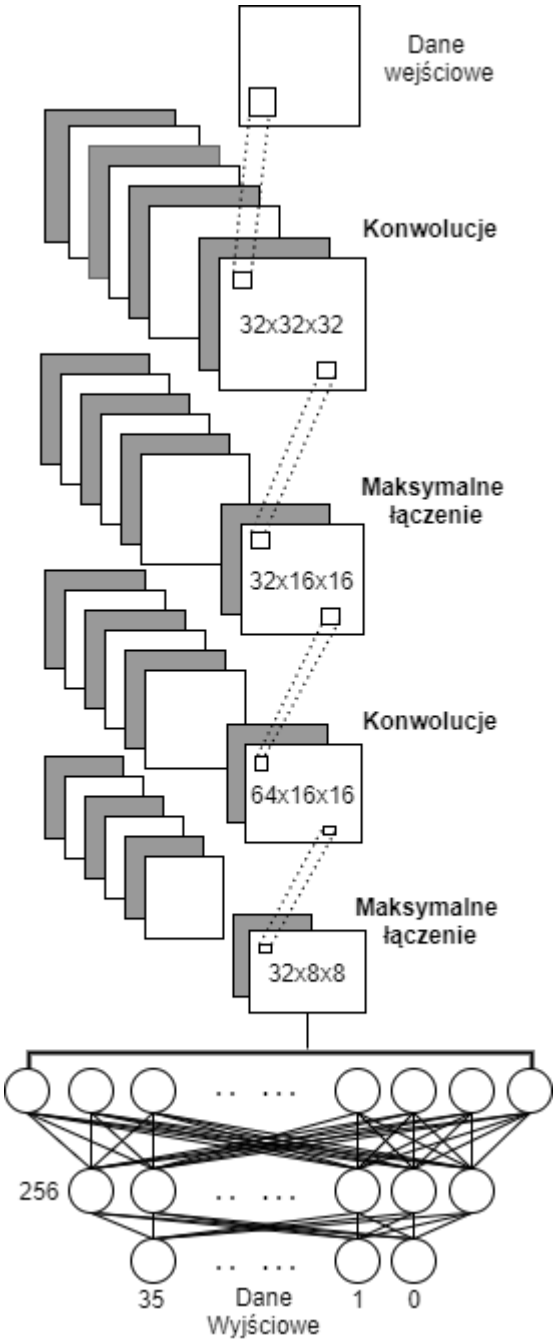
6.4. Metoda badawcza

Dla każdego algorytmu optymalizacji przeprowadzono badanie na CPU i GPU. W celu określenia, która konfiguracja najefektywniej wykonuje zadanie, przeprowadzono analizę wielokryterialną metodą sumy ważonej [20]. Polega ona na łączeniu kolejnych kryteriów w funkcję celu przy pomocą wzoru:

$$F(x) = \sum_{i=1}^{m} w_i f_i(x) \qquad (1)$$

gdzie $w_i \in [0, 1]$ jest wagą kryterium, x wektorem rozwiązań. Metoda ta wymaga wcześniejszej normalizacji danych. Normalizacja została przeprowadzona za pomocą metody unitaryzacji zerowej [21]. Aby zastosować tę metodę należy wyznaczyć kryteria oceny oraz określić, które z nich wpływają pozytywnie na ocenę końcową, a które negatywnie. Wyznaczono następujące kryteria oceny:

1) precyzja (ang. *accuracy*) - oblicza jak często predykcje pasują do etykiet danych szkoleniowych,
2) strata (ang. *loss*) - określa jak bardzo przewidywane wartości odbiegają od rzeczywistych wartości w danych szkoleniowych,
3) precyzja walidacji - precyzja dla danych testowych,
4) strata walidacji - strata dla danych testowych,
5) czas - łączny czas uczenia.



Rysunek 1: Schemat sieci neuronowej na podstawie [19].

Wśród kryteriów oceny należało wskazać stymulatory i destymulatory. Stymulatory wraz ze wzrostem wartości wpływają pozytywnie na ocenę końcową. Wśród badanych kryteriów te cechy posiada: precyzja i precyzja walidacji. Stymulatory wyznaczano za pomocą wzoru:

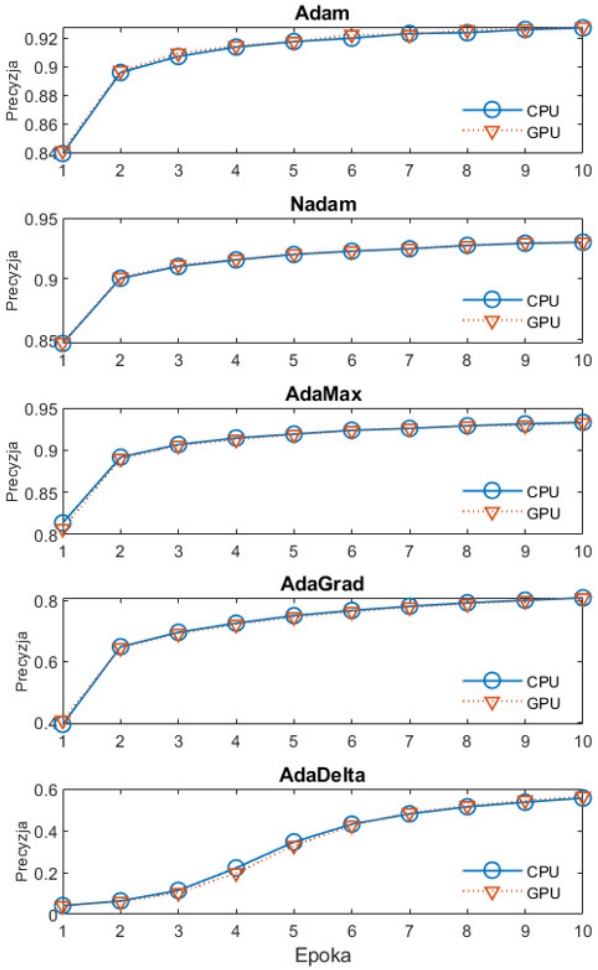
$$Z_{ij} = \frac{X_{ij} - min(X_{ij})}{max(X_{ij}) - min(X_{ij})} \quad (2)$$

gdzie i, j ∈ N, X_{ij} jest wartością badanego kryterium, zaś
$max(X_{ij})$ oraz $min(X_{ij})$ to kolejno wartości maksymalne i minimalne ze zbioru badanego kryterium oceny.

Destymulatory wraz ze wzrostem wartości mają negatywny wpływ na ocenę końcową. Wśród badanych kryteriów te cechy posiada: strata, strata walidacji, czas. Destymulatory wyznaczano za pomocą wzoru:

$$Z_{ij} = \frac{max(X_{ij}) - X_{ij}}{max(X_{ij}) - min(X_{ij})} \quad (3)$$

gdzie *i, j*, X_{ij}, $max(X_{ij})$, $min(X_{ij})$ odpowiadają zmiennym ze wzoru 2. W ostatnim kroku wyznaczono wagi dla poszczególnych kryteriów oceny. Najważniejszymi kryteriami oceny algorytmu optymalizacji jest precyzja oraz strata, które kolejno mają wagi: 0,4; 0,3. Kryteriom precyzja oraz strata walidacji nadano wagę 0,2, gdyż dane dla tych parametrów dobierane są losowo. Na ostatnie kryterium badawcze wpływa w dużym stopniu środowisko badawcze, dlatego też czas ma wagę 0,3.



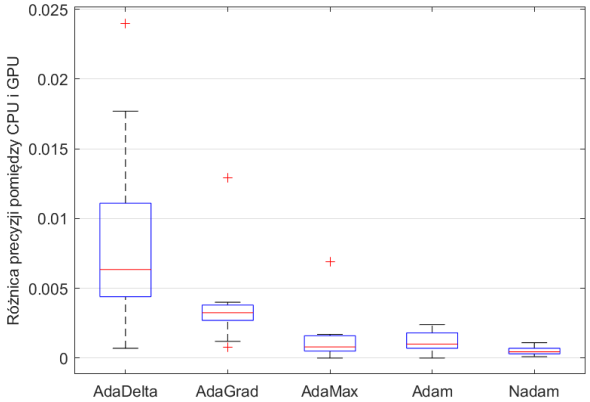
Rysunek 2: Porównanie precyzji treningu algorytmów dla badanych podzespołów.

7. Analiza rezultatów

Badania przeprowadzono według scenariusza badawczego, który składał się z:

- załadowania danych do programu,
- określenia algorytmu optymalizacji,
- przeprowadzenia szkolenia,
- zapisu otrzymanych rezultatów.

Na Rysunku 2 przedstawiono wykres obrazujący precyzję badanych algorytmów dla kolejnych epok (ang. *epochs*) programu. Dane przedstawione na Rysunku 2 pokazują, że dla każdego badanego algorytmu wartość precyzji na kolejnych epokach jest bardzo podobna dla mikroprocesora i karty graficznej. Różnice jest trudno dostrzec ludzkim okiem, dlatego też utworzono wykres pudełkowy pokazujący, który z analizowanych algorytmów optymalizacji wykazuje największe różnice w wartościach precyzji dla CPU i GPU (Rysunek 3).



Rysunek 3: Różnice precyzji algorytmów dla CPU i GPU.

Z Rysunku 3 można odczytać, że największe różnice wykazał algorytm AdaDelta, różnice otrzymane dla tego algorytmu wahały się pomiędzy 0,0007 oraz 0,0240. W porównaniu do algorytmów Adam, Nadam oraz AdaMax różnica ta jest stosunkowo duża. Wśród badanych optymalizatorów najmniejsze różnice wykazał algorytm Nadam.

Tabela 1: Rezultaty eksperymentu dla GPU

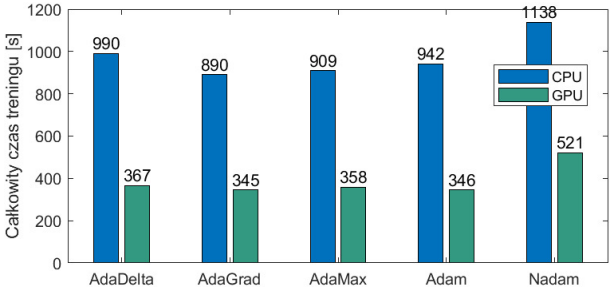
	precyzja	strata	precyzja walidacji	strata walidacji
AdaDelta	0,5635	1,4788	0,6547	1,2370
AdaGrad	0,8089	0,6353	0,8556	0,4923
Adam	0,9278	0,2285	0,9299	0,2308
Nadam	0,9302	0,2180	0,9331	0,2210
AdaMax	0,9324	0,2187	0,9374	0,2055

Tabela 2: Rezultaty eksperymentu dla CPU

	precyzja	strata	precyzja walidacji	strata walidacji
AdaDelta	0,5561	1,4929	0,6477	1,2632
AdaGrad	0,8097	0,6336	0,8542	0,4902
Adam	0,9271	0,2310	0,9322	0,2216
Nadam	0,9301	0,2213	0,9347	0,2204
AdaMax	0,9329	0,2166	0,9364	0,2059

W Tabelach 1 i 2 przedstawiono rezultaty eksperymentu dla poszczególnych podzespołów komputera w ostatniej iteracji programu. Z danych zamieszczonych w tabelach 1 i 2 wynika, że algorytm AdaMax osiągnął najlepsze rezultaty pod względem każdego z badanych kryteriów. Precyzja tego algorytmu osiągnęła wartość 0,9329 dla GPU, przy stosunkowo niskich stratach 0,2166. Najgorsze wyniki uzyskał algorytm AdaDelta. Precyzja osiągnęła wartość 0,5635 przy czym straty algorytmu osiągnęły wysokie wartości w porównaniu z pozostałymi algorytmami. Dla każdego algorytmu wartości precyzji walidacji pokrywają się z wartościami precyzji szkolenia.

W celu porównania czasów treningu poszczególnych algorytmów sporządzono wykres zależności algorytmu od jego całkowitego czasu treningu (Rysunek 4).



Rysunek 4: Czas treningu algorytmów optymalizacji.

Z Rysunku 4 wynika, że trening przeprowadzony za pomocą GPU jest blisko 3 razy szybszy dla każdego z algorytmów. Obliczenia wykazały ponad 60% spadek czasu treningu dla algorytmów AdaDelta, AdaGrad, Adam i AdaMax. Algorytm Nadam osiągnął 54% spadek czasu treningu, przy czym okazał się najwolniejszym ze wszystkich algorytmów.

W Tabeli 3 i 4 przedstawiono dane po przeprowadzeniu normalizacji danych przedstawionych w Tabelach 1 i 2.

Tabela 3: Znormalizowane rezultaty badań dla CPU

Algorytm	precyzja	strata	precyzja walidacji	strata walidacji	czas
AdaDelta	0,0000	0,0000	0,0000	0,0000	0,5968
AdaGrad	0,6730	0,6733	0,7153	0,7311	1,0000
Adam	0,9846	0,9887	0,9855	0,9852	0,7903
Nadam	0,9926	0,9963	0,9941	0,9863	0,0000
AdaMax	1,0000	1,0000	1,0000	1,0000	0,9234

Tabela 4: Znormalizowane rezultaty badań dla GPU

Algorytm	precyzja	strata	precyzja walidacji	strata walidacji	czas
AdaDelta	0,0000	0,0000	0,0000	0,0000	0,8750
AdaGrad	0,6652	0,6690	0,7106	0,7220	1,0000
Adam	0,9875	0,9917	0,9735	0,9755	0,9943
Nadam	0,9940	1,0000	0,9848	0,9850	0,0000
AdaMax	1,0000	0,9994	1,0000	1,0000	0,9261

Wyniki otrzymane w Tabelach 3 i 4 zostały wykorzystane do przeprowadzenia analizy wielokryterialnej metodą sumy ważonej. Wyniki przeprowadzonych obliczeń przedstawiono w Tabeli 5.

Tabela 5: Suma ważona badanych algorytmów optymalizacji

	CPU	GPU	
Algorytm	Suma ważona		Średnia
AdaDelta	0,1279	0,1875	0,1577
AdaGrad	0,7575	0,7524	0,7550
Adam	0,9441	0,9861	0,9651
Nadam	0,7800	0,7797	0,7799
AdaMax	0,9836	0,9841	0,9839

Według danych przedstawionych w Tabeli 5 najwydajniejszym algorytmem optymalizacji dla GPU jest algorytm Adam, zaś dla CPU AdaMax, jednak różnice pomiędzy tymi wartościami są bardzo małe. Wyliczone średnie sum ważonych, badanych podzespołów komputera, wskazują na to, że w przypadku korzystania z obu podzespołów najlepiej sprawdzi się algorytm AdaMax.

8. Podsumowanie i wnioski

W artykule przeprowadzono badania wydajności biblioteki Tensorflow przy pomocy metody sumy ważonej, która pozwoliła na wybór najlepszego algorytmu optymalizacji dla mikroprocesora i karty graficznej. Po obliczeniu średniej z sumy ważonej dla poszczególnych podzespołów (Tabela 5) można uznać hipotezę H1 za nieprawdziwą. Najbardziej wydajnym algorytmem optymalizacji jest AdaMax.

Na podstawie analizy danych przedstawionych na Rysunku 2 potwierdzono prawdziwość hipotezy H2. Czas treningu na GPU dla większości algorytmów był o blisko 60% krótszy niż w przypadku CPU. Wykorzystywany podzespół komputera nie ma dużego wpływu na precyzję w procesie uczenia. Różnice w wartościach parametrów otrzymanych podczas treningu były bardzo małe.

Wnioski wynikające z przeprowadzonego badania mogą być wykorzystane w procesie wyboru algorytmu optymalizacji w procesach tworzenia systemów uczenia maszynowego do rozpoznawania znaków pisma odręcznego.

Literatura

[1] J. McCarthy, From here to human-level AI, Artificial Intelligence 171 (2007) 1174–1182, https://doi.org/10.1016/j.artint.2007.10.009.

[2] T. Okuda, S. Shoda, AI-based chatbot service for financial industry, Fujitsu Scientific and Technical Journal 54 (2018) 4–8.

[3] S. Green, J. Heer, C. D. Manning, Natural language translation at the intersection of AI and HCI, Communications of the ACM 58 (2015) 46–53, https://doi.org/10.1145/2767151.

[4] K. Chakraborty, A. Talele, S. Upadhya, Voice recognition using MFCC algorithm, International Journal

of Innovative Research in Advanced Engineering (IJIRAE) 1 (2014) 158–161.

[5] H. Fujiyoshi, T. Hirakawa, T. Yamashita, Deep learning-based image recognition for autonomous driving, IATSS research 43 (2019) 244–252, https://doi.org/10.1016/j.iatssr.2019.11.008.

[6] A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais, A. Mueller, J. Kossaifi, A. Gramfort, B. Thirion, G. Varoquaux, Machine learning for neuroimaging with scikit-learn, Frontiers in neuroinformatics 8 (2014) 1–14, https://doi.org/10.3389/fninf.2014.00014.

[7] J. Moolayil, J. Moolayil, S. John, Learn Keras for Deep Neural Networks, Birmingham: Apress, 2019.

[8] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer et al., Theano: A Python framework for fast computation of mathematical expressions, Computing Research Repository (2016) 1–19.

[9] G. Zaccone, M. R. Karim, Deep learning with TensorFlow: Explore neural networks and build intelligent systems with python, Packt Publishing Ltd, 2018.

[10] S. Bahrampour, N. Ramakrishnan, L. Schott, M. Shah, Comparative study of deep learning software frameworks, Computing Research Repository (2015).

[11] S. Raschka, Python. Uczenie maszynowe, Packt Publishing Ltd, 2017.

[12] Firmy wykorzystujące bibliotekę Tensorflow, https://www.tensorflow.org/about/case-studies, [26.06.2021].

[13] Opis architektury biblioteki Tensorflow, https://developers.googleblog.com/2017/09/introducing-tensorflow-datasets.html, [26.06.2021].

[14] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, International Conference on Learning Representations (2015) 1–15.

[15] T. Dozat, Incorporating Nesterov Momentum into Adam, International Conference on Learning Representations (2016) 1–4.

[16] A. Lydia, S. Francis, Adagrad–an optimizer for stochastic gradient descent, International Journal of Information and Computing Science 6 (2019) 566–568.

[17] M. D. Zeiler, Adadelta: an adaptive learning rate method, Computing Research Repository (2012).

[18] M. Tokovarov, M. Kaczorowska, M. Miłosz, Development of Extensive Polish Handwritten Characters Database for Text Recognition Research, Advances in Science and Technology Research Journal 14 (2020) 30–38, https://doi.org/10.12913/22998624/122567.

[19] E. Łukasik, M. Charytanowicz, M. Miłosz, M. Tokovarov, M. Kaczorowska, D. Czerwinski, T. Zientarski, Recognition of handwritten Latin characters with diacritics using CNN, Bulletin of the Polish Academy of Sciences: Technical Sciences 69 (2021) 1–12, http://dx.doi.org/10.24425/bpasts.2020.136210.

[20] E. Zitzler, Evolutionary algorithms for multiobjective optimization: Methods and applications, Ithaca: Shaker. 1999.

[21] K. Kukuła, Metoda unitaryzacji zerowanej na tle wybranych metod normowania cech diagnostycznych, Acta Scientifica Academiae Ostroviensis 4 (1999) 5–31.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 336–343
Received: 12 September 2021
Accepted: 23 September 2021

Analysis of user experience during interaction with selected CMS platforms

Analiza doświadczenia użytkownika podczas interakcji z wybranymi platformami CMS

Michał Miszczak*, Mariusz Dzieńkowski

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The purpose of this study was assessing user experience while working with two popular CMS systems: WordPress and PrestaShop. The evaluation was done using a questionnaire and an eye tracking technique. Average task completion time, the number of fixations, the percentage of correctly completed tasks and the SUS index were used for comparisons. On the basis of the obtained results which, were collected during and after the users' interaction with a given system, it is difficult to clearly state which CMS proved to be better.

Keywords: user experience; Content Management System; CMS; e-commerce; eye tracking

Streszczenie

Celem pracy była ocena doświadczenia użytkownika podczas pracy z dwoma popularnymi systemami CMS: WordPress i PrestaShop. Oceny dokonano za pomocą ankiety oraz z użyciem techniki eyetrackingowej. Do porównań wykorzystano średni czas realizacji zadań, liczbę fiksacji, odsetek poprawnie zrealizowanych zadań oraz wskaźnik SUS. Na podstawie otrzymanych wyników, zebranych podczas i po interakcji użytkowników z danym systemem trudno jednoznacznie stwierdzić, który CMS okazał się lepszy.

Słowa kluczowe: doświadczenie użytkownika; System Zarządzania Treścią; CMS; handel elektroniczny; eyetracking

* Corresponding author

Email address: michal.miszczak@pollub.edu.pl (M. Miszczak)

1. Wstęp

Bardzo ważnym aspektem w projektowaniu interfejsów dla serwisów internetowych czy też różnego typu systemów informatycznych jest doświadczenie użytkownika (ang. User Experience - UX). Użytkownik jest najważniejszym elementem systemu i należy dbać o to, aby dostarczyć mu produkt w postaci aplikacji WWW, serwisu internetowego czy rzeczywistego interfejsu, którym będzie się intuicyjnie posługiwał oraz będzie zadowolony z interakcji. Odpowiednio zaprojektowany i zoptymalizowany interfejs aplikacji internetowej bądź systemu pozwala użytkownikowi wykonywać pracę w szybki i prosty sposób.

System CMS (ang. Content Management System) to pakiet oprogramowania, który zapewnia określony poziom automatyzacji zadań do skutecznego zarządzania kontentem. Jest to oprogramowanie dla wielu użytkowników działające na serwerze, które współdziała z treścią przechowywaną w repozytorium znajdującą się na tym samym serwerze. CMS pozwala użytkownikom tworzyć, edytować oraz redagować treści i udostępniać je osobom trzecim [1]. Systemy CMS znalazły również zastosowanie w e-commerce do budowania i obsługi sklepów internetowych. Możemy wyróżnić dwa rodzaje takich systemów. Pierwszy z nich to autorskie systemy zbudowane jako niepowtarzalne produkty. Drugi rodzaj to systemy typu Open Source czyli o otwartym kodzie, który można dostosowywać do indywidualnych potrzeb. Przykładem drugiego typu systemów mogą być PrestaShop i WordPress. W systemach CMS panele administracyjne powinny być tak zaprojektowane, aby każdy użytkownik, tj. osoba zajmująca się obsługą serwisu, pracownik, czy też właściciel firmy był w stanie korzystać z tego systemu bez posiadania specjalistycznej wiedzy i umiejętności informatycznych.

User Experience jest to całość wrażeń, doświadczeń, jakie odczuwa użytkownik podczas korzystania z produktu. Testowanie UX, czyli badanie użyteczności serwisów jest kluczowym elementem w zrozumieniu końcowego użytkownika oraz oszacowaniu wartości biznesowej projektu (serwisu/aplikacji). Badanie takie pomaga odpowiedzieć na pytanie, w jaki sposób można usprawnić funkcjonalność produktu informatycznego, sprawdzić, czy np. witryna działa prawidłowo oraz jak potencjalni użytkownicy radzą sobie podczas poruszania się po interfejsie serwisu [2]. Istnieje wiele metod badania UX, do których można zaliczyć: wywiad indywidualny, badanie kwestionariuszowe (ankietowe), studium przypadku (ang. case study), badanie eyetrackingowe, czy sondę kulturową.

Jedną z coraz częściej stosowanych metod testowania doświadczenia użytkownika jest technika eyetrackigowa. Metoda ta jest bardzo interesującym testem UX, ponieważ dzięki niej możemy dowiedzieć się, w jaki sposób użytkownicy oglądają witrynę/aplikację, na jakie elementy zwracają uwagę, a jakie elementy są pomijane. Wyróżnia się kilka technik, które mogą posłużyć do zbierania odpowiedniej ilości danych, niezbędnych do

realizacji analiz ilościowych i jakościowych. Należą do nich patrzenie swobodne oraz zadania, których cel warunkuje rodzaj ruchów i stan oczu. Podczas badania eyetrackingowego ruchy te są śledzone, rejestrowane i ostatecznie podlegają analizie [3]. Analizie poddaję się głównie dwa rodzaje ruchów oczu fiksacje oraz sakady. Fiksacje to momenty, w których człowiek zatrzymuję swój wzrok w danym punkcie, dzięki czemu możemy wyodrębnić tak zwane punkty zainteresowań. Sakady są to szybkie ruchy przemieszczania wzroku pomiędzy fiksacjami. Dzięki fikascjom możemy określić ścieżki skanowania wzroku. Ważnym elementem analizowania badań okulograficznych są mapy cieplne, które pokazują, na jakich elementach interfejsu użytkownik skupił wzrok najbardziej, a jakie elementy pominął. Dzięki takiej wiedzy projektując dany interfejs, wiadomo, jakie elementy wymagają jeszcze modyfikacji i poprawy. Do niedawna badania eyetrackigowe były dość rzadko stosowane w procesie badań UX ze względu na wysokie koszty sprzętu, z jakimi się one wiązały. Obecnie, gdy urządzenia do badań okulograficznych stały się tańsze, a przez to jednocześnie bardziej dostępne, coraz częściej wykorzystuję się tę technikę w pracach badawczych.

Podczas testowania UX wykorzystuje się więcej niż jedną metodę badawczą w celu uwiarygodnienia badań, poszerzenia obszaru badanego, czy pozyskania nowych danych. W ramach tej pracy zdecydowano się na połączenie kwestionariuszowego sondażu diagnostycznego z badaniami eyetrackingowymi.

2. Przegląd literatury

Ocena heurystyczna zaliczana jest do metod identyfikowania problemów z użytecznością. Bazuję się w niej na standardowych zasadach wypracowanych przez specjalistów. W pracy [4] dokonano oceny heurystycznej trzech otwartych systemów zarządzania treścią: WordPress, Joomla ora Drupal. Autor zaproponował sześć prostych zadań do wykonania przy użyciu wszystkich wymienionych systemów. Następnie zmierzył czas potrzebny na wykonanie konkretnego zadania, dokonał opisu, w jaki sposób polecenie zostało wykonane, podał problemy, jakie napotkał podczas ich wykonania oraz określił ich wagę. Badacz przygotował zadania dotyczące instalacji systemu, instalacji i aktywacji nowego motywu, utworzenia nowej strony i dodania jej do menu, umieszczenia zdjęcia na stronie, stworzenia strony kontaktowej oraz utworzenia nowego użytkownika. Po przeprowadzeniu eksperymentu i zebraniu wyników najpierw została zrealizowana ogólna analiza porównawcza wybranych systemów CMS, a następnie dokonano dogłębnej oceny heurystycznej każdego systemu z osobna. Wyniki badań ukazały główne problemy związane z interfejsem użytkownika oraz wskazały zasady heurystyczne, które zostały naruszone. Na podstawie uzyskanych rezultatów, wśród trzech testowanych systemów, najlepiej pod względem użyteczności został oceniony interfejs WordPressa.

W artykule [5] autorzy stwierdzili, że wykorzystanie najbardziej popularnych heurystyk Nielsena nie jest najbardziej odpowiednią i skuteczną metodą oceny CMS-ów i w związku z tym zdecydowali się opracować tzw. specyficzne heurystyki, które pomagają lepiej ocenić systemy CMS.

Z kolei w pracy [6] autorzy opracowali model oceny struktury nawigacyjnej systemów CMS z perspektywy programisty. Do tego celu wybrali trzy systemy CMS: WordPress, Joomla i Drupal, a następnie przeprowadzili badania ankietowe, w których wzięło udział kilkudziesięciu programistów. Uzyskane wyniki po zrealizowaniu badań porównawczych wybranych CMS-ów, pod kątem wsparcia nawigacyjnego wykazały, że istnieje korelacja między zidentyfikowanymi wskaźnikami oraz, że badane systemy zapewniają pomocne i skuteczne wsparcie nawigacyjne użytkownikom.

Autorka pracy [7] opisuje badania, które polegały na porównaniu istniejącej strony internetowej z prototypem serwisu. W badaniach, w których uczestniczyło kilkanaście osób, została użyta metoda eyetrackingowa. Skoncentrowano się w nich głównie na pomiarach wydajności, biorąc pod uwagę dokładność kliknięć i czas wykonania zadań. Scenariusz badawczy obejmował 25 zadań. Badanie śledzenia wzroku przeprowadzono w celu oceny nowego projektu strony, sprawdzenia, czy ułatwia ona szybkie zlokalizowanie określonych miejsc oraz znalezienia ewentualnych niedoskonałości strony.

W pracy [8] autorzy do oceny użyteczności trzech portali internetowych użyli metody eyetrackingowej oraz metody głośnego myślenia (ang. think aloud). Podczas badania uczestnicy mieli do wykonania dwa zadania na każdym z portali. Analiza otrzymanych danych pozwoliła zidentyfikować typowe wzorce zachowań, które wystąpiły u uczestników badań.

Z kolei w artykule [9] przy pomocy metody eyetrackingowej dokonano analizy zachowań konsumenta podczas wykonywania zakupów w sklepie internetowym. Wykorzystując wyniki w postaci mapy cieplnej, określono co przyciąga uwagę i na czym skupia swój wzrok użytkownik korzystający z serwisu e-commerce.

Podobne badania zrealizowano w pracy [10], w której przy pomocy eyetrackera sprawdzono, jak użytkownicy poruszają się po stronie sklepu internetowego, co przyciąga ich uwagę i gdzie napotykali ewentualne trudności. Analiza wyników z przeprowadzonych badań pozwoliła im odpowiedzieć na pytanie, jakich zmian należy dokonać na stronie, aby zwiększyć jej użyteczność i poprawić komfort podczas jej obsługi.

Eyetracking staje się coraz bardziej popularny w obszarze User Experience, co można zauważyć na podstawie rosnącej liczby publikacji z tej tematyki, a także poszerzającej się oferty firm komercyjnych z tej branży, włączających to narzędzie do swojego instrumentarium. Do rozpowszechnienia eyetrackingu jako techniki badawczej przyczyniło się wytworzenie urządzeń nowej generacji, które umożliwiają śledzenie ruchu gałek ocznych w warunkach naturalnych z dużą precyzją pomiaru.

3. Cel i zakres pracy

Celem pracy jest ocena dwóch popularnych systemów CMS do handlu elektronicznego. Ocena tych aplikacji

została dokonana na podstawie doświadczenia użytkownika podczas interakcji z tymi platformami między innymi za pomocą techniki eyetrackingowej.

Zakres pracy obejmuje: analizę literatury dotyczącej badań UX za pomocą techniki eyetrackingowej, wybór obiektów badań, dobór metod badawczych, opracowanie scenariuszy badawczych – zadań do wykonania przez użytkowników, realizację badań, wstępną analizę danych oraz ostateczną analizę wyników i sformułowanie wniosków.

Przed rozpoczęciem badań autorzy sformułowali następujące pytania badawcze, na które chcieli znaleźć odpowiedź:

1. Który z badanych systemów CMS-ów posiada bardziej przyjazny interfejs administratora?
2. Przy pomocy którego systemu CMS użytkownik wykona swoją pracę szybciej?

4. Metoda badawcza

Do wykonania badania UX zdecydowano się wykorzystać technikę eyetrackingową oraz badanie ankietowe. Szczegółowe etapy badań:

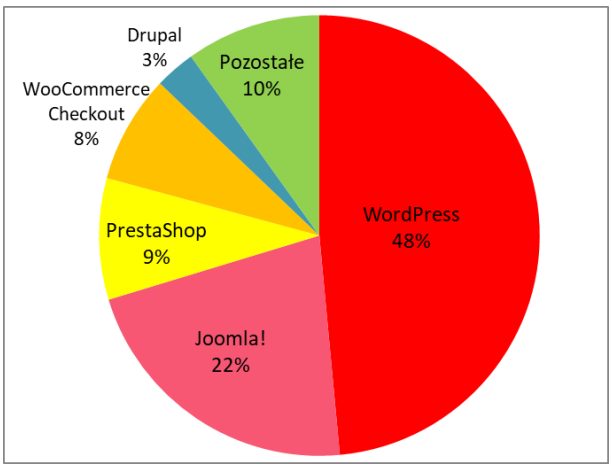
- badanie eyetrackingowe po którym została przeprowadzona analiza ilościowa wykorzystująca miary eyetrackingowe (liczbę fiksacji) oraz analiza jakościowa wykorzystująca ścieżki skanowania i mapy cieplne,
- pomiar czasów realizacji zadań,
- weryfikacja poprawności wykonania zadań oraz zliczenie liczby błędów i ich identyfikacja
- ocena satysfakcji podczas pracy z danym systemem CMS zrealizowana na bazie ankiety i wskaźnika SUS.

4.1. Obiekty badań

Przy wyborze systemów CMS do badań przyjęto kilka kryteriów. Pierwszym z nich była bezkosztowa dostępność, a drugim otwartość (ang. open source), czyli wzięto pod uwagę system, z którego każdy może nieodpłatnie skorzystać, i do którego kod źródłowy jest ogólnodostępny. Kolejne kryterium dotyczyło przeznaczenia systemu, który miał umożliwiać prowadzenie sprzedaży internetowej (e-commerce). Następnym kryterium była popularność wykorzystywania CMS-ów na stronach internetowych w Polsce (Rysunek 1).

Przy wyborze obiektów badań nie bez znaczenia były także doświadczenia zawodowe autorów i znajomość konkretnych systemów CMS. Do badania ostatecznie zdecydowano się wybrać dwa systemy tj. WordPress oraz PrestaShop, które w rankingu wykorzystania technologii Open Source w Polsce znalazły się w ścisłej czołówce (Rysunek 1) [11].

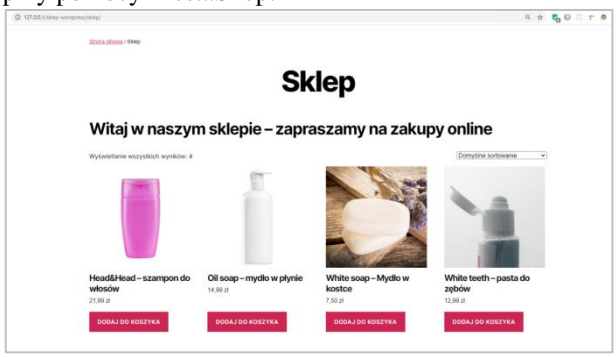
WordPress to najpopularniejszy system CMS, który oferuje wiele możliwości. System ten umożliwia tworzenie prostych stron internetowych, prowadzenie blogów oraz sklepów internetowych. Dzięki rozbudowanej bazie dodatków i możliwości budowania własnych wtyczek pozwala on na prowadzenie dużych portali internetowych, z których mogą korzystać tysiące użytkowników [12]. Wtyczką, która pozwala na wykorzystanie WordPressa w obszarze e-commerce jest WooCommerce i to tej wtyczki użyto, przygotowując system CMS do realizacji badań.



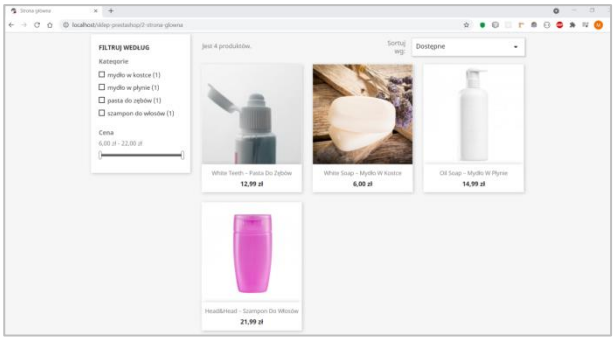
Rysunek 1: Statystyka wykorzystania technologii Open Source w aktywnych witrynach WWW w Polsce [11].

PrestaShop to otwarte oprogramowanie, które umożliwia prowadzenie sklepu internetowego. System ten stworzony jest z myślą o e-commerce. Posiada on również szeroką bazę dodatków - darmowych oraz płatnych, które można zainstalować w sklepie, choć do utworzenia prostego sklepu nie ma potrzeby instalacji dodatkowych wtyczek [13].

W celu przeprowadzenia badania utworzono dwa testowe sklepy internetowe wykonane odpowiednio przy pomocy systemów PrestaShop oraz WordPress z wtyczką WooCommerce. Obie aplikacje pobrano z oficjalnych stron internetowych producentów [12, 13], a następnie je zainstalowano i odpowiednio skonfigurowano. Sklepy działają na serwerze lokalnym Apache. Baza danych została utworzona przy pomocy narzędzia phpMyAdmin, które ułatwia zarządzanie bazą danych MySQL. Do obsługi serwera i bazy danych wykorzystano darmowe, wieloplatformowe oprogramowanie XAMPP. W utworzonych sklepach umieszczono po pięć produktów o identycznych nazwach, opisach, cenach i zdjęciach. Produkty te przypisano do odpowiednich kategorii, które wcześniej zostały utworzone. Rysunek 2 przedstawia wygląd aplikacji testowej widzianej po stronie klienta sklepu wykonanego przy pomocy WordPress, natomiast rysunek 3 sklepu wykonanego przy pomocy PrestaShop.



Rysunek 2: Widok aplikacji testowej - strony klienta, sklepu utworzonego na platformie WordPress.



Rysunek 3: Widok aplikacji testowej - strony klienta, sklepu utworzonego na platformie PrestaShop.

4.2. Grupa badawcza

W badaniu wzięło udział 11 osób (2 kobiety oraz 9 mężczyzn). Wszyscy uczestnicy badania to studenci Wydziału Elektrotechniki i Informatyki Politechniki Lubelskiej. Większość z nich to studenci studiów magisterskich ostatniego roku kierunku Informatyka. Z wywiadu przeprowadzonego przed badaniem ustalono, że większość uczestników przed przystąpieniem do wykonywania eksperymentu miała już styczność z jakimś systemem CMS.

4.3. Stanowisko badawcze

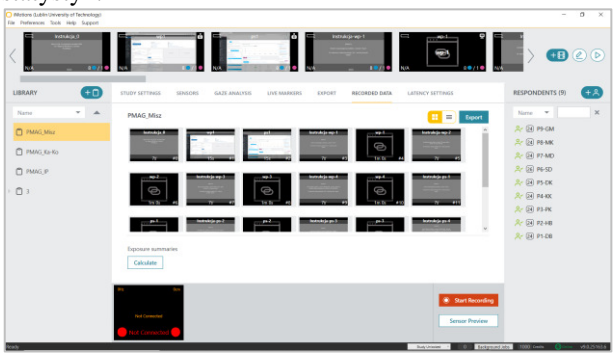
Eksperyment przeprowadzono w laboratorium Katedry Informatyki Politechniki Lubelskiej, w którym zostały zapewnione sztuczne warunki oświetleniowe, przy zasłoniętych oknach. Podczas badań każdy z uczestników siedział na regulowanym fotelu, co zapewniało nieskrępowane ruchy. W sesji badawczej oprócz osoby badanej brał udział moderator.

Rysunek 4: Stanowisko badawcze.

Stanowisko badawcze (Rysunek 4) składało się z eyetrackera Gazepoint GP3 HD podłączonego do laptopa, na którym było zainstalowane oprogramowanie Gazepoint Control oraz iMotions 9.0. Oprogramowanie służyło do zaprojektowania i przeprowadzenia eksperymentu oraz do analizy danych.

Użyty do badań eyetracker Gazepoint GP3 HD to urządzenie małe, lekkie i łatwo przenośne. Może być on postawiony na saniach nad klawiaturą laptopa lub przymocowany na stałe do monitora przy pomocy uchwytu VESA. Eyetracker Gazepoint GP3 HD może pracować z częstotliwością próbkowania 60 lub 150 Hz. Umożliwia on śledzenie obu oczu z dokładnością od 0,5 do 1° (w idealnych warunkach). Do detekcji stanów oka (fiksacji, sakad, zmian średnicy źrenicy, mrugnięć) wykorzystywana jest technika śledzenia tzw. jasnej źrenicy. Ten typ eyetrackera pozwala na swobodne poruszanie głową do 35 cm w poziomie i 22 cm w pionie. Optymalna odległość respondenta od eyetrackera to około 65 cm [14].

Urządzenie do śledzenia ruchów oczu było połączone z komputerem – laptopem ThinkPad T540p wyposażonym w procesor Intel Core i7-4710MQ (2,50GHz), pamięć operacyjną 16GB, kartę graficzną NVIDIA GeForce N14M-GS 730M, dysk 512GB SSD oraz ekran 15.6” z rozdzielczością 1920x1080. Komputer działał pod kontrolą systemu operacyjnego Windows 10, na którym zostało zainstalowane oprogramowanie Gazepoint Control, które jest niezbędne do działania eyetrackera i musi być uruchomione w tle podczas jego pracy. Natomiast do zaprojektowania eksperymentu, przeprowadzenia kalibracji, rejestracji sesji uczestników oraz późniejszego odtwarzania i edycji nagrań wykorzystano zaawansowaną platformę iMotions w wersji 9.0 (Rysunek 5) [15]. Oprogramowanie to pozwala na wizualizację wyników w postaci map cieplnych, map uwagowych, ścieżek fiksacji, czy obszarów zainteresowania. Możliwy jest również eksport danych oraz generowanie statystyk.



Rysunek 5: Okno program iMotions 9.0 przedstawiające otworzony projekt PMAG_Misz przygotowany do oceny dwóch systemów CMS (po lewej stronie lista eksperymentów, po prawej lista przebadanych uczestników, na górze projekt eksperymentu).

4.4. Opis eksperymentu

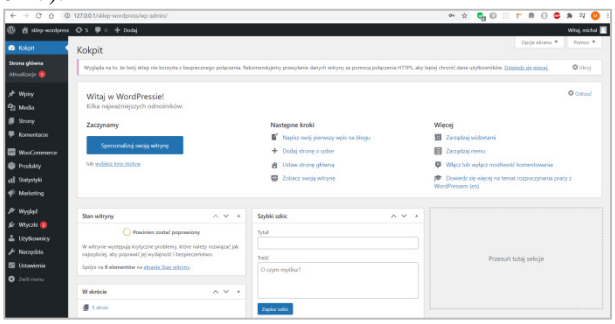
Do realizacji celu badań zaprojektowano eksperyment, który składał się z następujących etapów:

1. Sformułowanie problemu badawczego, celu badań, pytań badawczych
2. Wybór obiektów do badań, opracowanie scenariuszy badawczych (zadań dla użytkowników)
3. Przygotowanie projektu eksperymentu na platformie iMotions
4. Badanie pilotażowe oraz korekta eksperymentu
5. Nagrywanie uczestników badań
6. Weryfikacja uzyskanych wyników i analiza danych
7. Sformułowanie wniosków i ocena badanych CMSów

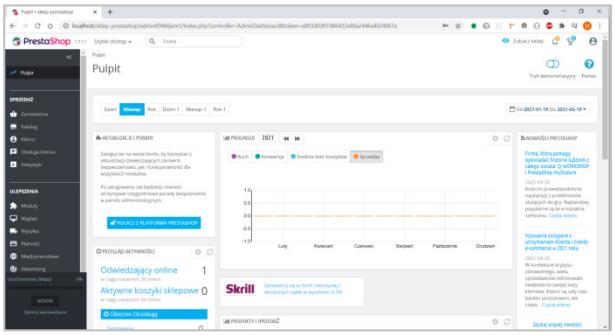
Tabela 1: Zadania do wykonania przez uczestników na dwóch CMS-ach

Lp.	Treść zadania
1	Utwórz nową kategorię produktu o nazwie „nowa”
2	Zmodyfikuj opis produktu o nazwie „Oil soap mydło w płynie” na nowy, dowolny opis.
3	Zmień cenę produktu o nazwie „White soap – mydło w kostce” z 6,00 zł na inną, dowolną.
4	Zmień obrazek produktu o nazwie „White teeth – pasta do zębów” na dowolne zdjęcie znajdujące się w folderze „Pliki” umieszczonym na pulpicie.

Wykorzystując utworzone testowe sklepy, uczestnicy badania mieli do wykonania odpowiednio po cztery zadania (Tabela 1) na każdym z nich. Zadania rozpoczynały się w tym samym miejscu, od widoku głównego panelu administracyjnego danego CMSa (Rysunki 6 i 7).



Rysunek 6: Widok główny panelu administracyjnego WordPress.



Rysunek 7: Widok główny panelu administracyjnego PrestaShop.

Przebieg sesji badawczej:

1. Przedstawienie uczestnikowi informacji na temat eksperymentu, jego celów, zapewnienie o nieinwazyjności badań, poinstruowanie jak ma się zachowywać w czasie badań.
2. Wyrażenie zgody uczestnika na udział w eksperymencie.
3. Kalibracja urządzenia.
4. Nagrywanie uczestników podczas wyświetlania instrukcji i realizacji zadań.
5. Wypełnienie przez respondentów ankiety badającej ich satysfakcję podczas interakcji z jednym i drugim CMS-em.
6. Krótkie poinformowanie użytkowników o wynikach badań – pokazanie przykładowej mapy cieplnej, ścieżki skanowania oraz filmu.

W badaniu ankietowym sprawdzającym satysfakcję użytkownika podczas korzystania z systemów wykorzystano Skalę Użyteczności Systemu (ang. System Usability Scale, SUS). Ankieta SUS składa się z 10 pytań, do których możemy przypisać odpowiednią ocenę z zakresu 0-4 [16]. W celu lepszej oceny badanych systemów CMS nieznacznie zmodyfikowano ankietę SUS, tzn. zmieniono w niej treść pierwszego punktu, który brzmiał: „Będę często korzystał z systemu”, a po zmodyfikowaniu przez autorów otrzymał następującą treść: „Gdybym miał korzystać z systemu CMS do prowadzenia sklepu internetowego, to skorzystałbym z tego systemu”. Narzędzie SUS pozwala określić poziom użyteczności systemu. Po przeprowadzeniu badania i zebraniu wyników oblicza się wskaźnik SUS, który przyjmuje wartość w skali od 0 do 100. Algorytm obliczania SUS jest bardzo prosty i przedstawia się następująco: zsumowanie punktów ze wszystkich punktów, pomnożenie obliczonej sumy przez 2,5, a następnie policzenie średniej. Im wyższy uzyskany wynik, tym lepsza użyteczność. Wartości powyżej 68 są interpretowane jako dobry wynik [17].

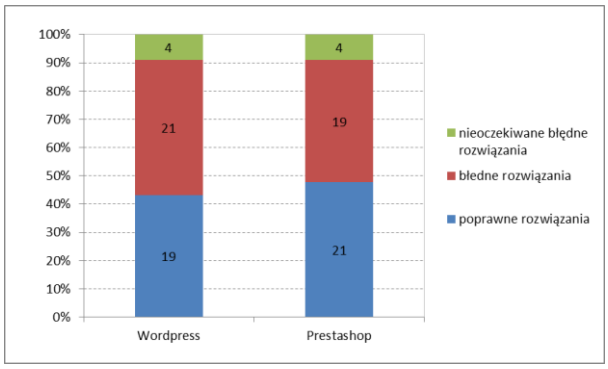
Tabela 2: Treść zmodyfikowanej ankiety SUS – badającej satysfakcję użytkowników [16]

Lp.	Treść pytania
1	Gdybym miał korzystać z systemu CMS do prowadzenia sklepu internetowego, to skorzystałbym z tego systemu.
2	System jest niepotrzebnie skomplikowany.
3	System jest łatwy w użyciu.
4	Będę potrzebował wsparcia technicznego, aby korzystać z systemu.
5	Różne funkcje systemu są łatwo dostępne.
6	W systemie jest zbyt wiele niespójności.
7	Większość osób będzie w stanie opanować system bardzo szybko.
8	System jest kłopotliwy w użyciu.
9	Czuję się bardzo pewnie korzystając z systemu.
10	Musiałem opanować wiele rzeczy przed rozpoczęciem pracy z systemem.

5. Wyniki badań

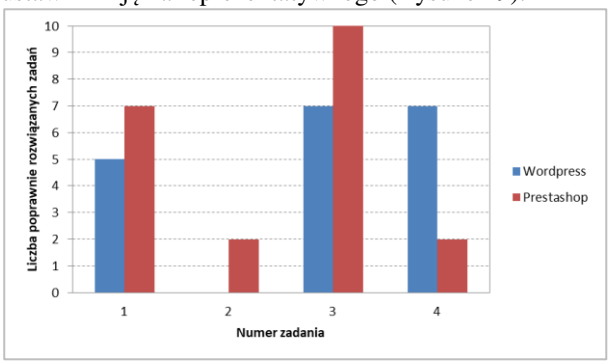
5.1 Identyfikacja poprawnych i błędnych realizacji zadań

Po przeprowadzeniu eksperymentu badawczego, w którym wzięło udział 11 uczestników, dokonano analizy nagrań. W pierwszej kolejności skupiono się na sprawdzeniu poprawności wykonanych zadań. W tym zestawieniu lepiej wypadł CMS PrestaShop, ponieważ łączna liczba poprawnych rozwiązań zadań wykonanych na tym systemie wynosiła 21, a na systemie WordPress 19. Zauważono, że niektóre błędne rozwiązania spowodowane były nieoczekiwanym kliknięciem spacji, które wywoływało przejście do kolejnego zadania. Zanotowano po cztery przypadki takich nieoczekiwanych błędów (Rysunek 8).



Rysunek 8: Liczba poprawnych oraz niepoprawnych rozwiązań.

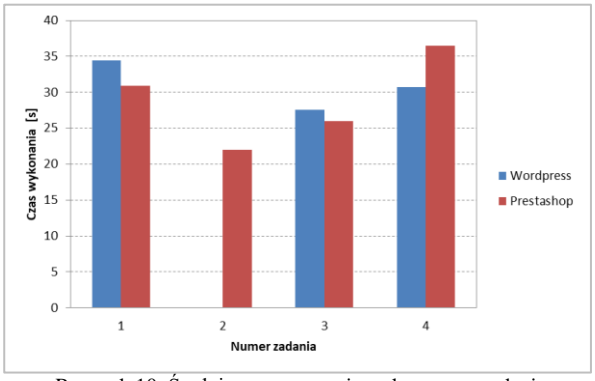
Dodatkowo sprawdzono poprawność wykonania poszczególnych zadań, co wykazało, że uczestnicy biorący udział w badaniach mieli największe problemy z zadaniem numer 2. Zadanie to polegało na zmodyfikowaniu opisu produktu. Błąd, jaki robili respondenci, polegał na tym, że większość z nich dokonywała edycji nazwy produktu, a nie opisu produktu. Niski odsetek poprawnych rozwiązań zanotowano również w zadaniu nr 4, głównie dla systemu PrestaShop. W zadaniu tym należało zastąpić bieżące zdjęcie, nowym zdjęciem, będącym główną grafiką firmującą określony produkt na stronie sklepu. Większość uczestników badania dodawała kolejne zdjęcie do galerii danego produktu, a nie ustawiła zdjęcia reprezentatywnego (Rysunek 9).



Rysunek 9: Liczba poprawnie rozwiązanych zadań.

5.2 Pomiar czasów wykonania zadań

W następnym etapie badań zmierzono czasy wykonania poszczególnych, dobrze wykonanych zadań i obliczono średnie czasy dla każdego zadania (Rysunek 10).

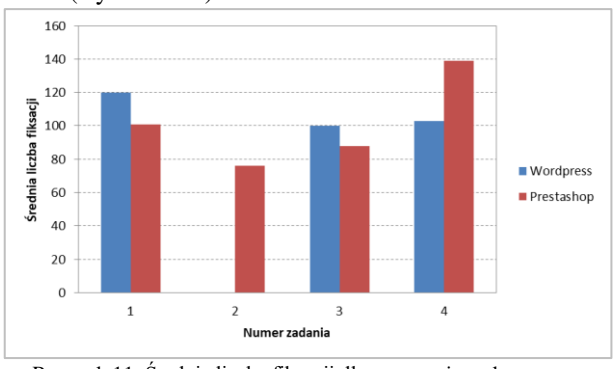


Rysunek 10: Średni czas poprawnie wykonanego zadania.

Okazało się, że średnie czasy realizacji zadań były zbliżone dla obu CMS-ów. W przypadku pierwszego zadania różnica wynosiła 3,5 sekundy, trzeciego 1,6 sekundy, a czwartego 5,8 sekundy.

5.3 Eyetrackingowa analiza ilościowa

Dokonano również analizy ilościowej, bazującej na wynikach z eyetrackera, w której wzięto pod uwagę miarę eyetrackingową – liczbę fiksacji występujących do momentu osiągnięcia celu czyli wykonania określonego zadania. Po zebraniu danych od wszystkich uczestników liczbę fiksacji dla każdego zadania uśredniono (Rysunek 11).



Rysunek 11: Średnia liczba fiksacji dla poprawnie wykonanego zadania.

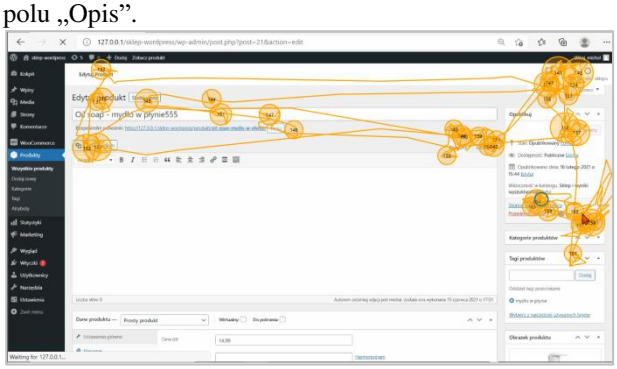
Średnia liczba fiksacji była proporcjonalna do średniego czasu realizacji zadań - im dłuższy był czas, tym więcej wystąpiło fiksacji. Dla pierwszego zadania realizowanego w systemie WordPress miało miejsce średnio 120 fiksacji, natomiast w systemie PrestaShop 101. Także w przypadku zadania numer 3, średnio więcej fiksacji wystąpiło podczas pracy z systemem WordPress niż PrestaShop – było to odpowiednio 100 i 88 fiksacji. Sytuacja inaczej wyglądała podczas wykonywania zadania czwartego. Tutaj więcej fiksacji miało miejsce przy pracy z systemem PrestaShop – 139 niż z systemem WordPress – 103. Duża liczba fiksacji może odzwierciedlać problem ze zrozumieniem danego obszaru interfejsu aplikacji. Może wynikać również z tego, że dany obszar jest bardziej skomplikowany i wymaga od użytkownika bardziej wytężonego myślenia.

5.4 Eyetrackingowa analiza jakościowa

Eyetrackingową analizę jakościową najczęściej przeprowadza się na podstawie ścieżek skanowania wygenerowanych dla wybranych respondentów po wykonaniu poszczególnych zadań. Polega ona na prześledzeniu zachowań wzrokowych uczestników eksperymentu w odpowiedzi na wyświetlony bodziec i zadane polecenie do wykonania. Innym sposobem wizualizacji wyników badań eyetrackingowych są mapy cieplne. Przedstawiają one zagregowane wyniki dla określonego zadania dla całej grupy uczestników. Za pomocą kolorów oznaczone są miejsca, w których respondenci skupiali swoją uwagę. Dzięki odpowiedniej kolorystyce zauważyć można, które elementy prezentowanej sceny przyciągnęły uwagę uczestników, a które zostały przez nich pominięte.

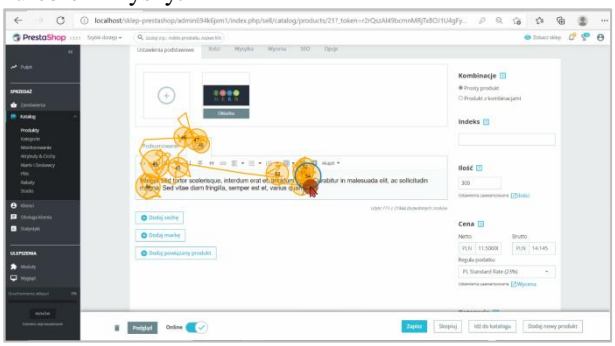
Rysunek 12 przedstawia fragment nagrania prezentujący ścieżkę skanowania dla niepoprawnie wykonane-

go zadania nr 2 na platformie WordPress. Uczestnik GM intensywnie skanuje górną część okna aplikacji oraz panel z prawej strony. Całkowicie pominięty został duży obszar, w którym należało wstawić opis. Widoczny jest brak fiksacji oraz kliknięć kursorem myszy w polu „Opis”.



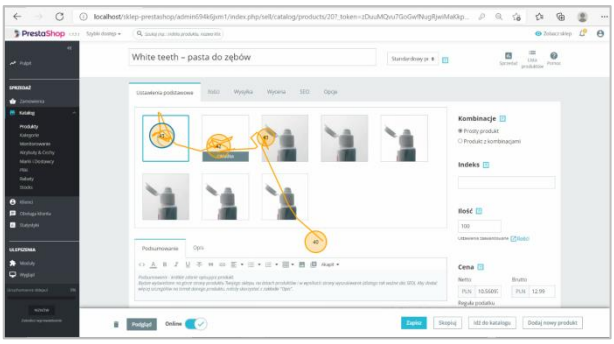
Rysunek 12: Ostatnia faza realizacji zadania nr 2 (interwał 10 sek.) przez uczestnika GM - ścieżka skanowania w oknie panelu administracyjnego WordPress.

Na Rysunku 13 pokazana została ostatnia faza poprawnej realizacji zadania nr 2 dla panelu administracyjnego systemu PrestaShop. Widać tu zarówno fiksacje w obszarze opisu, jak również wodzenie i kliknięcie kursorem myszy.

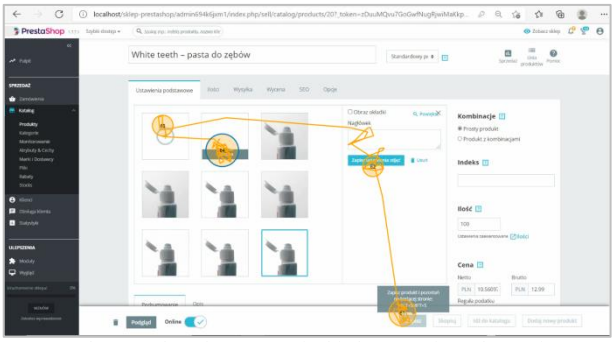


Rysunek 13: Ostatnia faza realizacji zadania nr 2 (interwał 3 sek.) przez uczestnika GM - ścieżka skanowania w oknie panelu administracyjnego Prestashop.

Z kolei na Rysunku 14 zaprezentowano fragment ścieżki skanowania podczas wykonywania zadania nr 4 zakończonego porażką. W tym przypadku użytkownik MK dodał do galerii produktu - kolejne zdjęcie, a nie jego zdjęcie główne, które mógłby ustawić, wybierając komponent o nazwie „Okładka”. Na Rysunku 15 pokazano dalszy fragment tego samego nagrania, w którym widoczna jest duża fiksacja (nr 64) – dłuższe zastanawianie się użytkownika w obszarze właściwego komponentu, na który należało kliknąć. Jednak respondent nie zdecydował się na zastosowanie tego elementu. Takie zachowanie może świadczyć o tym, że użytkownik nie był do końca pewny wcześniej podjętej decyzji.

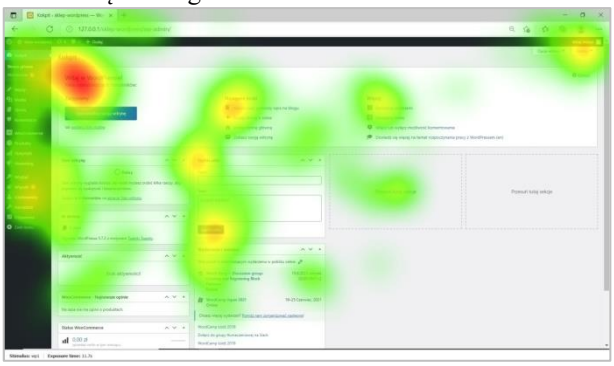


Rysunek 14: Wybrany fragment ścieżki skanowania podczas niepoprawnego wykonywania zadania nr 4 (interwał 1 sek.) przez uczestnika MK - panel administracyjny Prestashop.

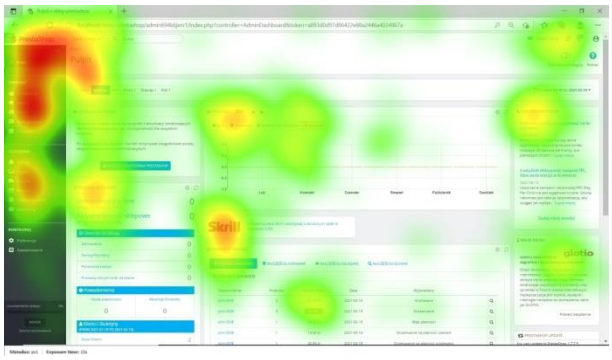


Rysunek 15: Dalszy fragment ścieżki skanowania podczas niepoprawnego wykonywania zadania nr 4 (interwał 2 sek.) przez uczestnika MK - panel administracyjny PrestaShop.

Rysunki 16 i 17 przedstawiają mapy cieplne dla obu systemów CMS podczas oglądania przez użytkowników ich interfejsów - okien głównych, bez wcześniej zadanego konkretnego polecenia do wykonania. Okna te były wyświetlone uczestnikom badań na początku eksperymentu przez 10 sekund. Różnice w rozkładzie obszarów gorących wynikają przede wszystkim z budowy poszczególnych CMS-ów. W przypadku systemu PrestaShop zauważalna jest większa koncentracja uwagi użytkowników na całym lewym panelu, w którym znajdowało się menu główne.



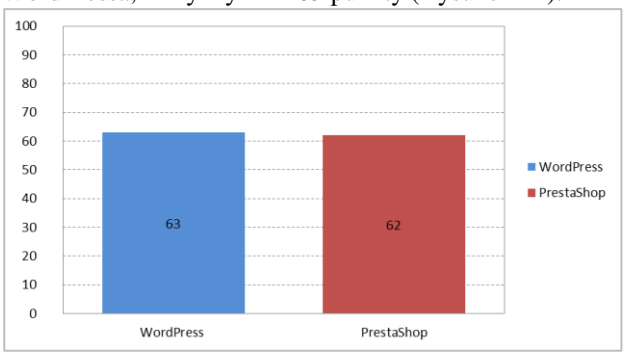
Rysunek 16: Mapa cieplna wygenerowana dla wszystkich uczestników badań dla panelu administracyjnego systemu WordPress.



Rysunek 17: Mapa cieplna wygenerowana dla wszystkich uczestników badań dla panelu administracyjnego systemu PrestaShop.

5.5 Badanie kwestionariuszowe - wskaźnik SUS

Końcowym etapem realizacji badań była analiza badania ankietowego, dzięki której udało się wyliczyć wskaźnik SUS, określający skalę użyteczności systemu. Oba systemy uzyskały wynik poniżej 68 punktów, który określany jest za wynik dobry. Badanym systemom zabrakło niewiele do osiągnięcia tej granicy punktowej. Wskaźnik SUS dla systemu PrestaShop wyniósł 62 punkty i był tylko o jeden punkt niższy od wskaźnika WordPressa, który wyniósł 63 punkty (Rysunek 14).



Rysunek 14: Wskaźnik SUS dla obu systemów.

6. Wnioski

Praca miała na celu sprawdzenie w sposób obiektywny – techniką eyetrackingową oraz subiektywny – metodą ankietową zadowolenia użytkowników po wykonaniu zadań w panelu administratora dwóch popularnych w Polsce systemów CMS dedykowanych do handlu elektronicznego. Przeprowadzone badania i zebrane wyniki nie pozwalają jednoznacznie odpowiedzieć na pierwsze pytanie badawcze: który CMS posiada bardziej przyjazny interfejs panelu administratora? Brak jednoznacznej odpowiedzi wynika z faktu, że oba systemy uzyskały bardzo zbliżone, różniące się jednym punktem rezultaty w skali SUS. Podobna sytuacja wystąpiła z drugim pytaniem badawczym, ponieważ średnie czasy wykonania zadań także były podobne, co uniemożliwiło jednoznaczne wskazanie CMS-a, za pomocą którego użytkownicy szybciej wykonują swoją pracę. Małe różnice pomiędzy testowanymi systemami mogą wynikać z faktu, że badanie uwzględniało tylko panele administracyjne tych CMS-ów, pozwalające na zarządzanie sklepami internetowymi, które mają podobną i dość skomplikowaną strukturę. Poziom wskaźnika SUS, przyjmujący wartość poniżej wyznaczonej doświadczalnie granicy 68 punktów dla obu systemów, jest informacją, wskazującą że interfejsy obu paneli administracyjnych wymagają poprawy.

Literatura

[1] D. Barker, Web Content Management: Systems, Features, and Best Practices, O'Reilly, 2016.

[2] T. Tullis, B. Albert, Measuring the user experience: collecting, analyzing, and presenting usability metrics, Morgan Kaufmann Publishers, Elsevier, 2008.

[3] M. Nilsson, UX method development from Usability testing with Eye tracking for E-commerce, 2018.

[4] M. H. Ibrahim, Usability Comparison Of Open Source Content Management Systems, (2014), https://lutpub.lut.fi/handle/10024/98501, [25.11.2020].

[5] R. Bos, J. Gurp, J. H. Verpoorten, S. Brinkkemper, Heuristic Evaluation Of Content Management Systems: Cms Specific Heuristics, Published in P. Isaias and M.B. Nunes (Eds.), Proceedings of the IADIS International Conference WWW/Internet (2005), 247-254.

[6] S. Gilani, A. Majeed, M. Muzammal, A.U. Rehman, H. Zaheer, Z. Jan, A Navigational Evaluation Model for Content Management Systems, The Nucleus 53(2) (2016) 82-88.

[7] A. Bojko, Using Eye Tracking to Compare Web Page Designs: A Case Study, A Case Study, Journal of Usability Studies 3(1) (2006) 112-120.

[8] P. Weichbroth, K. Redlarski, I. Garnik, Eye-tracking Web Usability Research, Proceedings of the Federated Conference on Computer Scienceand Information Systems (2016) 1681–1684.

[9] W. Wong, M. Bartels, N. Chrobot, Practical Eye Tracking of the Ecommerce Website User Experience, International Conference on Universal Access in Human-Computer Interaction UAHCI 2014 (2014) 109-118, https://doi.org/10.1007/978-3-319-07509-9_11.

[10] Y. Herawati, S. Halim, C. Tesavrita, Evaluasi Website Rakuten Indonesia dengan Eyetracking Usability Testing, Jurnal Rekayasa Sistem Industri, 5(1) (2016) 60-68.

[11] Statystyka wykorzystania systemów typu open source na stronach WWW w Polsce, https://trends.builtwith.com/cms/open-source/country/Poland, [26.08.2021].

[12] Oficjalna strona WordPress Polska, https://pl.wordpress.org/, [04.06.2021].

[13] Oficjalna strona PrestaShop Polska, https://www.prestashop.com/pl, [04.06.2021].

[14] Gazepoint, https://www.gazept.com/product/gp3hd/, [09.09.2021].

[15] iMotions, https://imotions.com/, [09.09.2021].

[16] SurveyLab, Skala Użyteczności Systemu (SUS), https://www.surveylab.com/pl/blog/skala-uzytecznosci-systemu-sus/, [09.09.2021].

[17] J. Saouro, 5 Ways to Interpret a SUS Score, https://measuringu.com/interpret-sus-score/, [23.10.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 344–348
Received: 14 September 2021
Accepted: 14 October 2021

COVID-19 pandemic impact on the Polish community of twitch.tv streaming platform

Wpływ pandemii COVID-19 na polską społeczność platformy streamingowej twitch.tv

Kamil Jeżowski*, Marcin Badurowicz

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The paper describes the process of analyzing the data of the Polish community on the Twitch.tv streaming platform. A description of the platforms and tools used to research the data was presented. The research was conducted on the basis of three issues. Data on the number of live broadcasts and the number of recipients were analyzed. Separate charts have been generated for each of them. The code used to create the charts was written in the R language. Conclusions were drawn on the basis of the generated charts.

Keywords: streaming; twitch.tv; COVID-19; content creator

Streszczenie

W pracy został opisany proces analizowania danych polskiej społeczności na platformie streamingowej Twitch.tv. Przedstawiony został opis platform oraz narzędzi wykorzystanych do badania danych. Badania zostały przeprowadzone w oparciu o trzy zagadnienia. Analizie poddane zostały dane dotyczące ilości nadawanych transmisji na żywo oraz liczby odbiorców. Do każdego z nich zostały oddzielne wygenerowane wykresy. Kod służący do stworzenia wykresów został napisany w języku R. Na podstawie wygenerowanych wykresów wyciągnięte zostały wnioski.

Słowa kluczowe: transmisje na żywo; twitch.tv; COVID-19; twórca internetowy

*Corresponding author

Email address: **kamil.jezowski@pollub.edu.pl** (K. Jeżowski)

1. Introduction

In the times of the COVID-19 pandemic, the Internet strongly entered in every area of human life. Due to the closure of society at home, the Internet has become indispensable in everyday life. For this reason, the phenomenon of remote work has significantly increased and online entertainment has become something much more common than before.

The lack of entertainment in everyday life has resulted in an increase in the broadly understood interest in computer games among the society of all ages. Even before COVID-19, the gaming and streaming industry grew rapidly, and the global outbreak of the pandemic significantly increased the attraction to gaming. The internet has made it possible for people not only to play games, but also to constantly watch people play them. Multiplayer games have been a staple of the gaming community for many years. There are many platforms that allow millions of players around the world to play the same game with each other. During the COVID-19 pandemic, the online social game Among Us saw its popularity burst. It is a multiplayer and multiplatform game that gained its peak of popularity in September 2020. It was also a hit of popularity on the Twitch.tv streaming service.

In addition to games, content creators who provide entertainment in real time have also gained popularity. Anyone who decides to devote their time to creators faces a difficult choice, because there are different streaming platforms, and the choice itself is just the beginning of the adventure. Currently, the recipient can choose from several platforms, including Facebook Gaming, YouTube or Twitch. After choosing a platform, the user looking for entertainment must define their expectations, such as the language of broadcasting and the game. Each profile includes a name, title of the broadcast, type of activity or profile picture [1]. More and more often we can meet big companies entering the streaming world. With the development of the COVID-19 pandemic and the growing interest in streaming among people, many creators who had previously given up live broadcasts to maintain better contact with their audience returned to the Twitch.tv platform. The appearance of completely new faces is noticeable, which, thanks to their charisma, gained popularity very quickly.

The loss of contact with the fans was felt by the musicians who, due to the lack of concerts, do not have direct contact with the fans. The same is the case with athletes, the lack of contact pushed both these social groups towards streaming, so now fans can watch livestreams how their idols are doing in online games, and additionally they can ask them their questions and exchange opinions with other online chat participants.

2. Materials and methods

2.1. Twitch.tv

Twitch started in 2011 and has quickly become one of the most important digital video entertainment platforms

[2]. In 2014, Twitch was bought by Amazon for $ 970 million. This platform is currently used to broadcast private live broadcasts of video games, e-sports tournaments, press conferences and various special events.

However, most of the broadcasts are private, and streamers, thanks to their audience, can fully devote themselves to their work, which is regular live streaming. Viewers can support their favorite creators through voluntary cash donations, bits or subscriptions. In addition, the streamer has the ability to run ads during the broadcast, thanks to which he can also earn money. Streamers very often form groups within which they cooperate and share a large crowd of viewers. These types of creators start broadcasting right after each other so that the content is conveyed to the recipient all the time, without break.

Streamers currently have a lot of influence on computer game companies, because they are the people who review games in some way and decide whether they will promote a game [3].

Twitch in 2020 could boast of such results as:

- an average of 30,000,000 unique visitors per day;
- 7,000,000 unique streamers per month;
- 1,000,000,000,000 minutes viewed [4];

2.2. Research group

The research was conducted on ten carefully selected streamers. The selection criteria were regularity of streaming and the time from which the creator has been broadcasting live. In terms of regularity, the creators could not have too large discrepancies in the number of transmission hours in the following months and could not take breaks from streaming, which would be indicated by low activity on the platform, because such a situation could lead to errors in the analysis. In the case of the time from which the creator is streaming, it had to be at least January 1, 2019. 10 streamers who met these requirements were selected.

2.3. R (programming language)

The R language was used for the research. It is also an environment for statistical calculations and their graphical presentation. It was developed at Bell Laboratories by John Chambers and his associates. The strength of the R language is the ease with which you can create high-quality graphs containing mathematical symbols and formulas where needed. Every effort has been made to ensure that the default values when creating the chart generate a transparent result, but the user still retains full control over the generated image and can modify it freely.

2.4. RStudio

RStudio is an integrated development environment for the R programming language. It has been carefully considered and anticipates the needs of new R users, and provides high-quality tools for the more advanced. RStudio simplifies many of the functionalities available in R and is the perfect combination of an R text interface with a graphical user interface [5-7].

3. Results

3.1. Epidemic impact on the number of unique viewers

The first case of COVID-19 in Poland was detected on March 4, 2020. March 16, 2020, an epidemic threat was introduced throughout the country, and remote teaching entered into force on March 25. Due to this fact, March 2020 was adopted as the month of the outbreak of the pandemic in Poland [8].

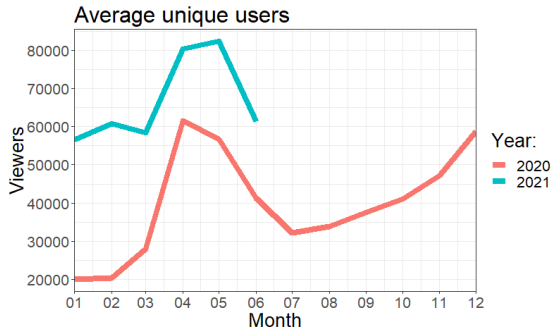


Figure 1: Monthly average unique users - chart

Figure 1 presents a graph of the average number of unique users since the beginning of 2020. A unique user is a person who visited the twitch.tv platform at least once a day.

The chart above shows that the average number of unique viewers in the first months of 2020 (January, February) fluctuated around 20,000. In the first month of COVID-19 coronavirus case in Poland, it increased by almost 10,000. April was the month of the greatest increase in unique users. Then, along with lifting the restrictions, the number of unique viewers decreased, and the introduction of new restrictions caused it to grow. Therefore, it can be concluded that the pandemic had a significant impact on the average number of unique viewers on the twitch.tv platform and caused their large increase.

3.2. Epidemic impact on the number of streamers

Figure 2 presents a chart showing the monthly average number of unique

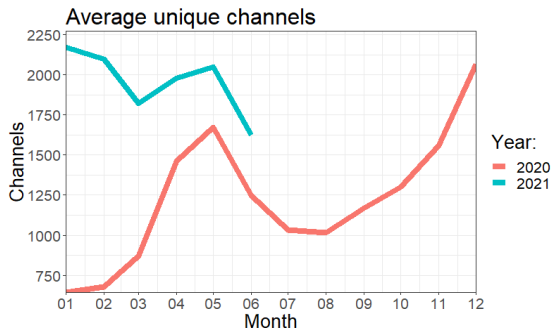


Figure 2: Monthly average unique streamers – chart

Polish channels on which live broadcasts took place. The chart shows that in April 2020 there was a signifi-

cant increase in the number of unique channels transmitting broadcasts, and in the following month there was also an increase, although it was not as large as in April. The number of channels in May increased by almost 100% compared to March.

On this basis, it can be concluded that the outbreak of the pandemic in Poland had a significant impact on the number of streamers, which has visibly increased.

3.3. The impact of introducing a red zone on the number of viewers

The increase in COVID-19 coronavirus infections, successive records of patients and deaths, have forced the government to introduce new sanitary restrictions. At the conference, which was organized on October 23, 2020, the introduction of the red zone throughout the country was announced. Restrictions included remote teaching, sports events without audience participation, the operation of swimming pools, aqua parks and gyms was suspended, and young people under the age of 16 were not allowed to move outside without adult supervision between 8:00 and 16:00. The easing of the restrictions did not begin until the following year, at the beginning of February [8, 9].

In order to determine the impact of the introduction of the red zone on the average number of viewers, two months before the introduction and two months after the introduction of the red zone will be taken into account. It will be the period from August to December.

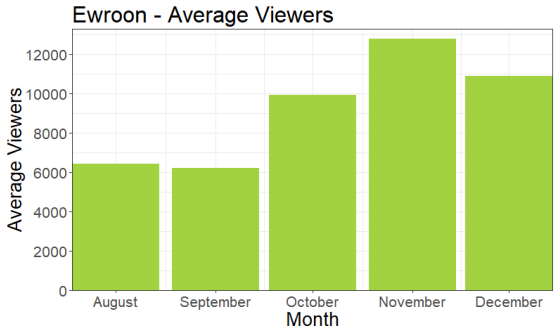


Figure 3: Monthly average viewers – Ewroon chart

Figure 3 above shows a chart of a creator who in the months following October achieved a higher average number of viewers than in the months before the introduction of the red zone across the country.

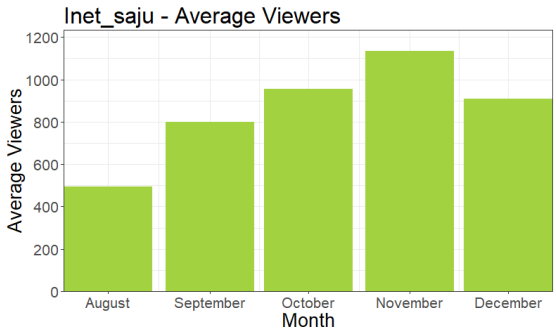


Figure 4: Monthly average viewers – Inet_saju chart

The charts presented on Figures 4 and 5 show the content creators who also achieved a noticeably better result in the average number of viewers in the period after the introduction of the red zone in Poland.

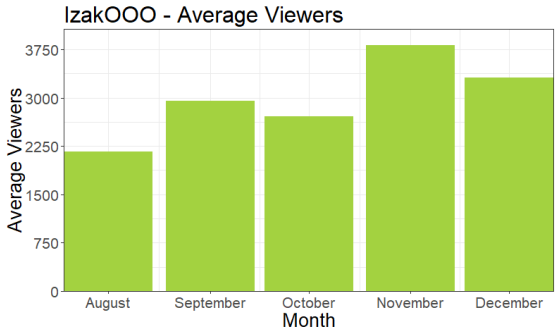


Figure 5: Monthly average viewers – IzakOOO chart

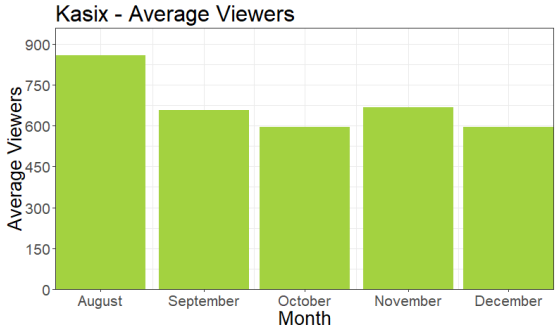


Figure 6: Monthly average viewers – Kasix chart

Creators whose charts are presented in Figures 6 and 7 also, as in the case of previous streamers, recorded an increase in the average number of viewers in the month following the introduction of the red zone.

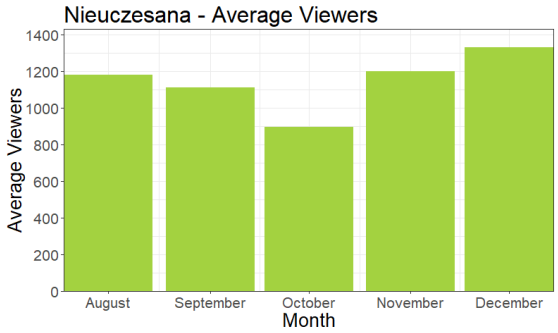


Figure 7: Monthly average viewers – Nieuczesana chart

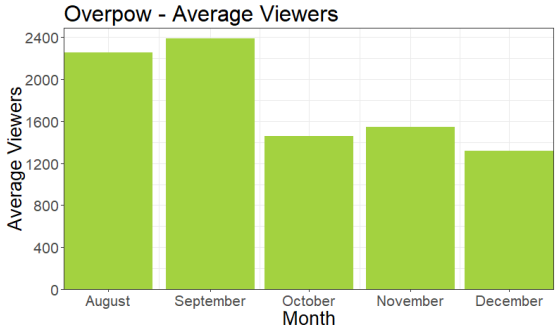


Figure 8: Monthly average viewers – Overpow chart

Figure 8 shows the creator who outperformed the average number of viewers in November but did not sustain this rising trend in the following month.

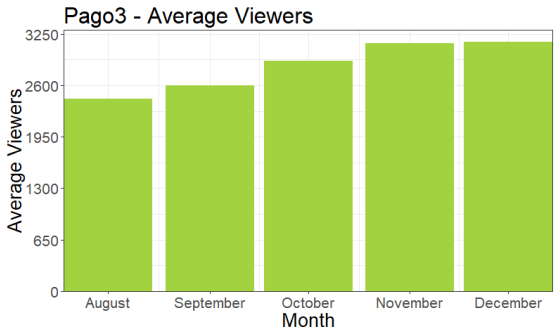


Figure 9: Monthly average viewers – Pago3 chart

The graph of the creator in Figure 9 shows the increase in the average number of viewers after the introduction of the red zone.

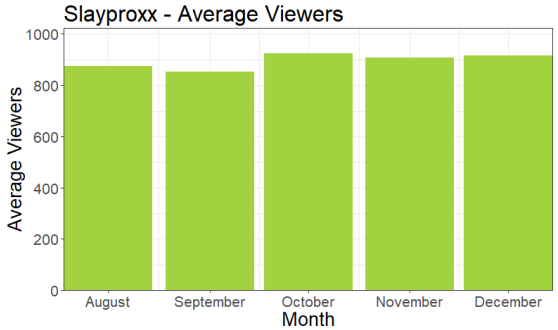


Figure 10: Monthly average viewers – Slayproxx chart

The graph of the creator presented above (Figure 10) shows that the introduction of the red zone has no influence on the achieved result of the average number of viewers.

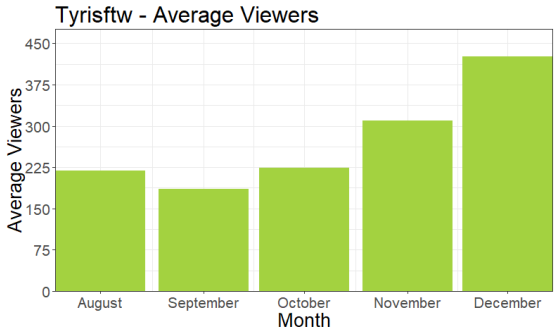


Figure 11: Monthly average viewers – Tyrisftw chart

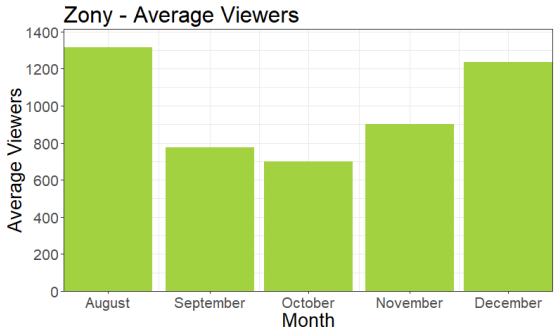


Figure 12: Monthly average viewers – Zony chart

Figures 11 and 12 show the graphs of content creators who obtained significantly better results in terms of the average number of viewers in the months after the introduction of the red zone in Poland.

4. Conclusions

The aim of the study was to analyze the data of the Polish community on the Twitch.tv streaming platform in the face of the COVID-19 pandemic.

The first issue concerned the unique number of visitors to the Twitch.tv platform during the COVID-19 coronavirus pandemic. It was possible to answer this question unequivocally. With the outbreak of the pandemic, the platform began to be visited by many more people looking for entertainment. In April 2020, the average number of unique viewers was around 60,000. Comparing this result to February 2020, when this number was around 20,000, it can be clearly concluded that the pandemic had a positive impact on the number of viewers. Additionally, it is worth mentioning that the average number of unique users in May 2021 reached over 80,000.

Another query concerned the impact of the pandemic on the number of people broadcasting live. After analyzing the data, it was possible to answer this question directly. The outbreak of the pandemic had a huge impact on the number of people broadcasting live. The number of people streaming in April 2020 was twice as high as in February 2020, and in February 2021 this number was almost three times higher than in the previous year.

The last issue was the impact of the introduction of the red zone throughout Poland in October 2020 on the number of viewers. In order to better look at this issue, two months before and two months after this event were taken into account. In this case, the months of August - December were the research period. An analysis of the data selected in this way showed that the majority of the surveyed content creators obtained a greater average number of viewers in the months following October than in the months preceding it.

References

[1] S. Anderson, Watching People Is Not a Game: Interactive Online Corporeality, Twitch.tv and Videogame Streams, Games Studies 17(1) (2017) 1-16.

[2] E. Gandolfi, To watch or to play, it is in the game: The game culture on Twitch.tv among performers, plays and audiences, Journal of Gaming & Virtual Worlds 8(1) (2016) 63-82.

[3] M. Johnson, J. Woodcock, The impacts of live streaming and Twitch.tv on the video game industry, Media Culture & Society 41(5) (2018) 670-688.

[4] Twitch advertising, https://twitchadvertising.tv/audience/, [31.07.2021].

[5] J. Racine, RStudio: a platform-independent IDE for R and Sweave, Journal of Applied Econometrics 27(1) (2012) 167-172.

[6] Ch. Gandrud, Reproducible Research with R and RStudio Second Edition, Taylor & Francis Group (2015) 12-14.

[7] J. Allaire, RStudio: Integrated Development Environment for R, useR! 2011, Coventry, 2011.

[8] COVID-19 pandemic in Poland https://www.medonet.pl/koronawirus-pytania-i-odpowiedzi/sars-cov-2,koronawirus---aktualne-obostrzenia-w-polsce--aktualizacja-,artykul,98382723.html, [24.07.2021].

[9] Official profile of the Chancellery of the Prime Minister, https://twitter.com/PremierRP, [08.08.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 349–355
Received: 14 September 2021
Accepted: 20 September 2021

A study of the user experience when interacting with applications that work with sports armbands to monitor human activity

Badania doświadczenia użytkownika podczas interakcji z aplikacjami współpracującymi z opaskami sportowymi do monitorowania aktywności człowieka

Mateusz Kiryczuk*, Paweł Kocyła*, Mariusz Dzieńkowski

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

This paper concerns the study of user experience and focuses on two aspects i.e. usability and user satisfaction. Two mobile applications for monitoring human activity, Mi Fit and Google Fit, were tested. Both applications work with sports armbands. Two methods were used for the study: a questionnaire and eye tracking. The comparison of the applications was made on the basis of the collected results from questionnaires, measurements of task completion times and the number and type of errors detected. Nine respondents participated in the study. The Google Fit application received a higher average score for user satisfaction, fewer errors and shorter task completion times.

Keywords: user experience; usability; sports armband; eye tracking

Streszczenie

Praca dotyczy badania doświadczenia użytkownika, koncentrując się na dwóch aspektach to jest: użyteczności i satysfakcji użytkownika. Testom poddano dwie aplikacje mobilne do monitorowania aktywności człowieka: Mi Fit oraz Google Fit. Obie aplikacje współpracują z opaskami sportowymi. Do badań wykorzystano dwie metody: kwestionariuszową i eyetrackingową. Porównania aplikacji dokonano na podstawie zebranych wyników z ankiet, pomiarów czasów realizacji zadań oraz liczby i rodzaju zidentyfikowanych błędów. W badaniach wzięło udział 9-ciu respondentów. Wyższą średnią ocenę satysfakcji użytkowników, mniejszą liczbę błędów przez nich popełnianą oraz krótszymi czasami wykonania zadań uzyskała aplikacja Google Fit.

Słowa kluczowe: doświadczenie użytkownika; aplikacje mobilne; opaska sportowa; eyetracking

*Corresponding author

Email address: mateusz.kiryczuk@pollub.edu.pl (M. Kiryczuk), pawel.kocyla@pollub.edu.pl (P. Kocyla)

1. Wstęp

Troska o swoje zdrowie połączona z dbałością o kondycję i sprawność fizyczną jest coraz powszechniej wspomagana przez oprogramowanie sportowe instalowane na urządzeniach mobilnych. Popularność zyskują aplikacje do wykonywania ćwiczeń w domu, biegania, robienia pompek czy ćwiczeń mięśni brzucha. Tego typu aplikacje mogą też mieć formę uniwersalną do wykonywania ćwiczeń ogólnorozwojowych i ogólnousprawniających. Zwykle aplikacje tego typu współpracują ze smartwatchem i opaską fitness. Stosowane są do systematycznego i rozsądnego uprawiania sportu oraz do utrzymywania ciała w dobrej kondycji fizycznej.

Aplikacje mobilne działające na urządzeniach przenośnych mają specyficzne cechy (ograniczona wielkość ekranu i wciąż jeszcze mała wydajność), które odróżniają je od aplikacji na komputery stacjonarne czy laptopy. Oprócz tego, że aplikacje stosowane w celu poprawy kondycji powinny poprawnie, szybko i wydajnie działać, wykorzystywać wbudowane podzespoły takie jak GPS czy żyroskop, to powinny także być łatwe i wygodne w użytkowaniu oraz w konsekwencji dawać satysfakcję użytkownikowi.

Użyteczność aplikacji mobilnych to klucz do sukcesu oprogramowania, ponieważ musi być ono postrzegane przez użytkowników jako łatwe do opanowania obsłudze, przyjazne dla użytkownika i mało czasochłonne podczas wykonywania zadań. Takie oprogramowanie zyskuje popularność i cieszy się akceptacją wśród użytkowników.

W testowaniu jakości interfejsów mobilnych oprócz klasycznych metod, wykorzystuje się technikę eyetrackingową. Pozwala ona precyzyjnie określić, na jakich elementach aplikacji użytkownik skupił swój wzrok oraz jak przebiega interakcja użytkownika z aplikacją. Za pomocą eyetrackera możliwa jest rejestracja wideo aktywności wzrokowej, punktów skupienia wzroku i ruchów oczu.

W ramach niniejszej pracy zostało przeprowadzone badanie eyetrackingowe oraz ankietowe na dwóch aplikacjach mobilnych współpracujących z opaskami i smartwatchem w celu sprawdzenia, a także porównania ich użyteczności oraz satysfakcji użytkowników.

2. Przegląd literatury

W ostatnich latach zauważalny jest przyspieszony rozwój technologii informatycznych, który wywołuje coraz częstsze dyskusje na temat jakości interfejsów aplikacji.

W artykule [1] autor twierdzi, że nie sama funkcjonalność aplikacji wpływa na jej użyteczność oraz zainteresowanie przez użytkowników. To w jaki sposób jest przedstawiona owa funkcjonalność jest równie ważne jak ona sama. Niezależnie od tego jak wiele daje perspektyw oraz jak bardzo zaawansowana jest aplikacja, trudność w jej użyciu może być barierą nie do pokonania. Autor artykułu wskazuje, że dodatkowo ważnym elementem w odczuciu użytkownika jest spójność całej aplikacji. Nie chodzi tu wyłącznie o samą szatę graficzną, ale również o spójne modele, przekazywane wiadomości czy też odpowiednie rozmieszczenie elementów na ekranie.

W odczuciu użytkownika bardzo ważny jest zarówno wygląd aplikacji jak i zrównoważona ilość przedstawianych informacji. W przypadku nadmiaru wyświetlanych funkcji oraz treści, następuje dezorientacja odbiorcy, która może prowadzić do niechęci z korzystania z programu. O wiele lepszym rozwiązaniem jest zaprojektowanie aplikacji o ściśle określonej funkcjonalności i zawężonym zastosowaniu. Niesie to również za sobą kolejną korzyść, jaką jest skrócony czas oczekiwania na uzyskanie odpowiedzi ze strony aplikacji. Długie i częste opóźnienia pracy skutkują spadkiem efektywności w interakcji użytkownika z urządzeniem, co również działa negatywnie na rozpowszechnianie się aplikacji [2].

Ważnym elementem w interfejsie inteligentnych opasek na rękę, podobnie jak w aplikacjach mobilnych, jest czytelność oraz odpowiednie rozmieszczenie ikon i przycisków. Istnieje jednak znacząca różnica w rozmiarze takich urządzeń, co wymaga jeszcze większej precyzji w dopasowywaniu elementów interfejsu urządzenia. Opaska powinna dostarczać użytkownikowi tylko niezbędnych informacji, które nie wymagają skomplikowanych ruchów ze strony użytkownika. Zaawansowane funkcje powinny być sterowane za pomocą urządzenia zewnętrznego, np. smartfona odpowiednio wcześniej skonfigurowanego z opaską. Takie rozwiązanie pozwala na czytelne i jasne przedstawienie wyników działania wszelkich funkcji [3].

W literaturze można znaleźć wiele artykułów dotyczących szeroko rozumianych badań narzędzi wspomagających uprawianie sportu. Artykuł [4] traktuje o dokładności pomiarów parametrów człowieka, takich jak liczba przebytych kroków czy liczba spalonych kalorii podczas wysiłku fizycznego. Okazuje się, że dane wprowadzone do aplikacji połączonej z urządzeniami monitorującymi aktywność użytkownika dają bardzo dobre wyniki, zgodne ze stanem rzeczywistym. W artykule zawarto wyniki z przeprowadzonych badań różnego sprzętu, a badania objęły użytkowników podzielonych na kilka grup wiekowych.

W kolejnej pracy [5] przedstawiono nowatorskie urządzenie typu smartband do monitorowania faz snu. Przeprowadzone zostały testy zaprojektowanego urządzenia, a następnie jego wyniki porównano z wynikami specjalistycznego sprzętu klinicznego. Autor sugeruje, że zaprojektowane urządzenie może odnieść sukces komercyjny, ponieważ uzyskane wyniki nieznacznie różnią się od wyników pomiarów na sprzęcie specjalistycznym.

Zespół z Korei Południowej wykonał badania dotyczące rozpoznawania ruchów rąk człowieka za pomocą opasek sportowych. Wyniki eksperymentu przedstawiono w artykule [6]. Wykazano w nim, że tani, komercyjny sprzęt jest w stanie rozróżnić wiele ruchów rąk podczas używania opaski. Autorzy przewidują, że dzięki tak wysokiej precyzji możliwy będzie w przyszłości rozwój i powstanie nowych funkcjonalności dla opasek w zakresie monitoringu aktywności użytkownika.

Irańscy naukowcy w artykule [7] opisują wyniki badań nad nowatorskim czujnikiem pozwalającym na określenie spalonych kalorii monitorowanych w czasie rzeczywistym za pomocą smartfona połączonego przez Bluetooth. Wyniki badań przeprowadzonych na grupie osób niezwiązanych ze sportem wykazały, że zaprojektowany czujnik cechuje się wysoką precyzją. Dzięki swoim niewielkim rozmiarom można go będzie wbudować w różnego rodzaju urządzenia takie jak smartband, smartwatch czy telefon komórkowy, wyposażone w moduł komunikacyjny Bluetooth.

W ostatnim czasie poszerzył się zakres badań okulograficznych. Powstaje wiele firm, które oferują badania ergonomii interfejsów. Eyetracking jest techniką umożliwiającą analizę ruchów gałki ocznej i stanów oczu osoby badanej. Ludzkie oko przeskakuje z miejsca na miejsce kilka razy na sekundę, ruch ten nazywa się sakadami. Informacje są wydobywane podczas fiksacji, czyli krótkich przerw pomiędzy sakadami, jest to moment w którym oczy są stosunkowo nieruchome. Dane uzyskane podczas badania mogą być przedstawiane w postaci ścieżek skanowania oraz map cieplnych. Ścieżki skanowania przedstawiają ruchy oczu osoby patrzącej na badany obiekt i składają się z fiksacji reprezentowanych jako kółka oraz z sakad, które są reprezentowane jako linie łączące fiksacje. Mapa cieplna za pomocą kolorów przedstawia miejsca, które najbardziej i najmniej przyciągają uwagę [8].

W pracy [9] autorzy przeprowadzili badania użyteczności aplikacji mobilnej, w których użyli techniki eyetrackingowej. Opracowano eksperyment, którego celem było przeprowadzenie testu użyteczności aplikacji „Sale Force Automation (SFA)" w warunkach zbliżonych do naturalnych. W badaniach tych przedstawiciel handlowy wykonywał przygotowane zadania na aplikacji SFA, zainstalowanej na rzeczywistym urządzeniu mobilnym. W badaniach wzięło udział 8 osób. Nie byli to przedstawiciele handlowi, a przyuczona grupa studentów, która wcześniej poznała codzienne czynności osoby wykonującej tego typu pracę. Badania użyteczności polegały na realizacji czterech scenariuszy. Głównym celem testowania użyteczności aplikacji była identyfikacja problemów dotyczących interfejsu, które były identyfikowane na podstawie analizy wideo ścieżek skanowania zarejestrowanych za pomocą mobilnego eyetrackera podczas wykonywania określonych zadań. Do każdego wykrytego błędu przypisywano określony priorytet, który zależał od częstotliwości występowania błędu oraz siły jego wpływu na realizację

zadania. W badaniach wzięto również pod uwagę czas wykonania poszczególnych zadań oraz wykryte błędy związane z gestami. Ostatnim elementem branym pod uwagę w badaniach była ocena satysfakcji użytkowników aplikacji przeprowadzona za pomocą ankiety składającej się z pięciu pytań [10].

3. Cel i teza badawcza oraz zakres pracy

Celem pracy jest ocena użyteczności i satysfakcji dwóch wybranych sportowych aplikacji mobilnych współpracujących z opaską i smartwatchem przeznaczonych do monitoringu aktywności sportowych i motywujących do systematycznego uprawiania sportu. Badania zrealizowano przy pomocy techniki eyetrackingowej oraz za pomocą ankiety, koncentrującej się na sprawdzeniu zadowolenia użytkowników podczas interakcji z daną aplikacją. Badania zostały zrealizowane stacjonarnie w laboratorium Katedry Informatyki Politechniki Lubelskiej.

Zakres pracy obejmował wybór obiektów do badań – dwóch aplikacji mobilnych, dobór metody badawczej, opracowanie zadań dla użytkowników, zaprojektowanie i przeprowadzenie eksperymentu, analizę i uogólnienie wyników oraz sformułowanie wniosków.

W ramach pracy została sformułowana teza mówiąca o tym, że „*szybkość wykonywania zadań na danej aplikacji oraz mniejsza liczba występujących przy tym problemów wpływa pozytywnie na doświadczenia użytkowników*”.

4. Metoda badawcza

Zaprojektowano eksperyment, w którym zastosowano dwie techniki oceny jakości interfejsów, biorąc pod uwagę użyteczność i satysfakcję użytkowników.
Etapy badań:

- eksperyment aktywny, polegający na realizacji scenariuszy, podczas których testowano interfejsy aplikacji mobilnych przez użytkowników
- analiza okulograficzna, podczas której użytkownicy mieli do wykonania po pięć zadań na każdej z aplikacji
- zbieranie opinii respondentów za pomocą ankiety do sprawdzenia zadowolenia użytkowników
- technika bierna, polegającą na obserwacji działań użytkownika w celu wykrycia błędów.

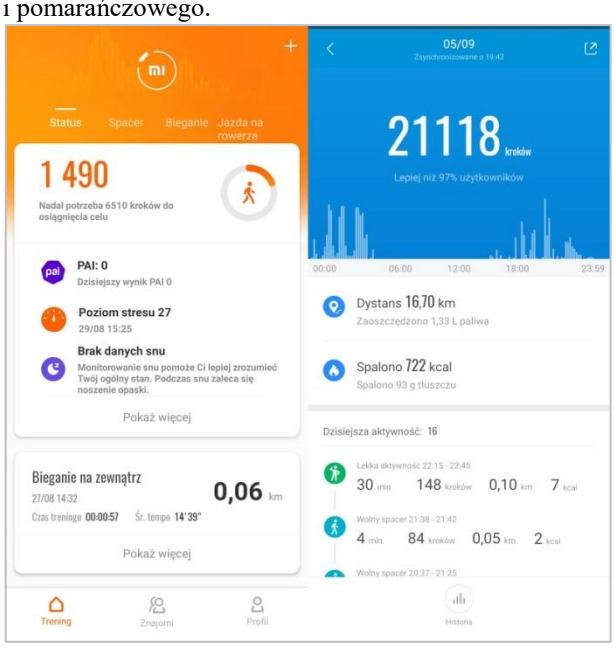
  Do porównań interfejsów wykorzystano:
- czas realizacji zadań,
- oceny użytkowników wybranych kryteriów dotyczących satysfakcji dokonanych na podstawie ankiety,
- analizę błędów.

4.1. Obiekty badań

Jako obiekt badań zostały wybrane dwie aplikacje mobilne służące do monitorowania aktywności fizycznej użytkownika: Mi Fit oraz Google Fit. Obie aplikacje są w pełni darmowe oraz dostępne w sklepie Google Play oraz App Store. Posiadają wiele możliwości, które zostały porównane w Tabeli 1.

Pierwsza z nich jest produktem pochodzącym od firmy Xiaomi i służy do połączenia opaski sportowej Mi Band ze smartfonem. Aplikacja przedstawiona na Rysunku 1 pozwala śledzić aktywność fizyczną w ciągu dnia: odbyty trening, liczbę kroków, pokonany dystans, puls oraz dane dotyczące snu. Wszystkie dane archiwizowane są w aplikacji. Ponadto istnieje możliwość konfiguracji rodzaju powiadomień, które będą dostarczone do opaski. Może to być na przykład informacja o połączeniu telefonicznym, przychodzącej wiadomości SMS czy innym powiadomieniu pochodzącym od aplikacji. Na ekranie startowym aplikacji wyświetlane są najważniejsze informacje takie jak ilość kroków przebyta w ciągu dnia, poziom stresu, informacje o jakości snu, podsumowanie ostatniego treningu oraz wykres z ilością kroków wykonanych w przeciągu ostatnich siedmiu dni. Wybranie jednej z ikon związanych z powyższymi opcjami powoduje pokazanie szczegółowych danych. Przykładowo po kliknięciu w liczbę kroków na ekranie pokazany jest godzinowy wykres z zaznaczonymi aktywnościami użytkownika. Widoczna jest również informacja o przebytym dystansie oraz liczbie spalonych kalorii. Dodatkowo wyświetlana jest lista ze szczegółowym opisem aktywności w danym dniu zawierająca ich czas trwania, liczbę wykonanych kroków, pokonany dystans i ilość spalonych kalorii. Istnieje również możliwość zobaczenia historii aktywności konkretnego dnia, tygodnia czy miesiąca.

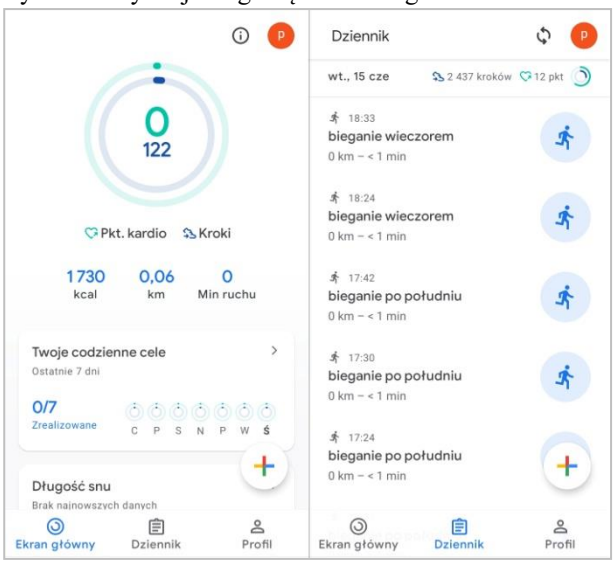
Z głównego ekranu aplikacji można w łatwy sposób, przesuwając palcem w lewo lub prawo, przejść do ekranu z aktywnościami takimi jak spacer, bieganie czy jazda na rowerze. Na dole okna znajdują się przyciski przenoszące do treningu, zakładki ze znajomymi oraz zakładka przenosząca do profilu, gdzie można skonfigurować ustawienia aplikacji czy zarządzać podłączonymi urządzeniami. Wygląd aplikacji Mi Fit zachowany jest w jasnej kolorystyce. Podobnie jak w innych produktach Xiaomi, dominuje kolor biały z dodatkiem niebieskiego i pomarańczowego.



Rysunek 1: Ekrany aplikacji Mi Fit.

Google Fit jest to platforma do monitorowania i mierzenia aktywności fizycznej dostarczona przez

firmę Google (Rysunek 2). Potrafi ona łączyć dane z wielu aplikacji i urządzeń. Do rejestrowania aktywności fizycznej wykorzystuje czujniki znajdujące się w urządzeniu mobilnym. Google Fit rejestruje każdą aktywność użytkownika: spacer, bieg, jazdę na rowerze czy inną aktywność, którą użytkownik wybierze. Ponadto aplikacja może zapisać przebytą trasę. Gogle Fit przechowuje wszystkie dane w chmurze. Na ekranie głównym wyświetlona jest liczba przebytych kroków, spalonych kalorii, zdobytych punktów kardio oraz przemierzony dystans i czas spędzony w ruchu. Ponadto dostępne są informacje dotyczące długości snu, tętna, wagi czy ciśnienia krwi. Po wybraniu danej statystyki, na wykresie wyświetlają się jej szczegółowe dane. Istnieje możliwość wyboru jednego z interwałów czasowych wyświetlanego wykresu: dzień, tydzień, miesiąc, a nawet rok. W dolnej części ekranu znajduje się menu umożliwiające wybór trzech opcji: ekran główny, dziennik oraz profil, w którym można zaktualizować informacje o użytkowniku takie jak waga, wzrost, wiek czy zdefiniować dzienne cele oraz zmienić ustawienia aplikacji. W dzienniku wyświetlana jest szczegółowa aktywność fizyczna użytkownika. Znajduje się tam informacja o rodzaju aktywności, długości jej trwania oraz o przebytym dystansie. Aplikacja kolorystycznie nie odbiega od pozostałych produktów dostarczanych przez Google. W jej ustawieniach istnieje możliwość wyboru motywu jasnego bądź ciemnego.



Rysunek 2. Ekrany aplikacji Google Fit.

Tabela 1: Porównanie możliwości obu aplikacji

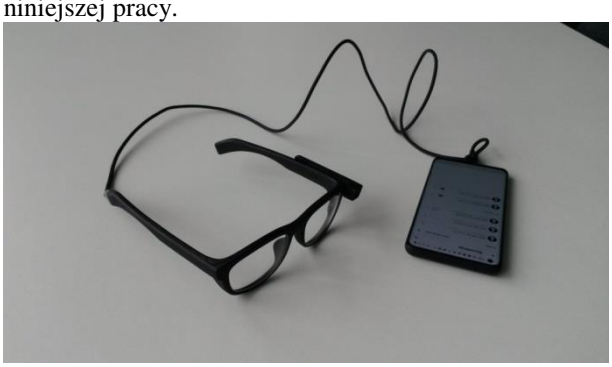
Lp.	Aplikacja	Funkcjonalności
1	Mi Fit	• monitorowanie BMI • porównywanie aktywności fizycznej i wagi z innymi użytkownikami • obserwacja aktywności znajomych • wysyłanie powiadomień do opaski • personalizacja urządzenia zewnętrznego • logowanie przez Facebook
2	Google Fit	• możliwość powiązania z innymi urządzeniami i aplikacjami • wprowadzanie pomiarów ciśnienia krwi • do używania wymaga jedynie smartfona • posiada samouczek wprowadzający do obsługi aplikacji

4.2. Grupa badawcza

W badaniach wzięło udział dziewięć osób, trzy kobiety i sześciu mężczyzn. Wiek badanych osób mieścił się w przedziale 21 - 30 lat. Przeważnie byli to studenci ostatniego roku kierunku Informatyka studiujący na Politechnice Lubelskiej. Wśród uczestników badań 77,8% zadeklarowało, że wcześniej korzystali z podobnych aplikacji, natomiast 55,6% miało do czynienia z opaską sportową, smartwatchem lub innym urządzeniem pozwalającym na monitorowanie aktywności fizycznej.

4.3. Stanowisko badawcze

Eksperyment został przeprowadzony w pomieszczeniu znajdującym się w Centrum Innowacji i Transferu Technologii Politechniki Lubelskiej należącym do Katedry Informatyki. Zapewnione były odpowiednie warunki oświetleniowe (sztuczne światło – świetlówki) bez dostępu światła słonecznego. Badania były prowadzone pod kontrolą moderatora, którym był jeden z autorów niniejszej pracy.


Rysunek 3: Eyetracker mobilny Pupil Invisible.

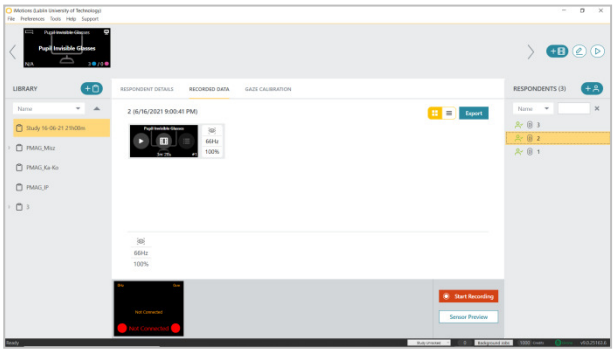
Urządzeniem, które zostało wykorzystane podczas badań był eyetracker mobilny Pupil Invisible (Rysunek 3) w postaci okularów wideo umożliwiających śledzenie obuoczne. Okulary są połączone ze smartfonem z zainstalowaną aplikacją do rejestracji obrazu sceny znajdującej się przed użytkownikiem oraz ruchów jego oczu. Szczegółowe parametry techniczne eyetrackera przedstawiono w Tabeli 2.

Tabela 2: Parametry techniczne eyetrackera Pupil Invisble [6]

Częstotliwość próbkowania	200Hz (śledzenie obuoczne)
Detekcja stanów oka	fiksacje, sakady, zmiany średnicy źrenicy
Technika śledzenia	ciemna źrenica
Dokładność w idealnych warunkach	0,6° obuocznie
Wbudowana kamera sceny	rozdzielczość 1088 x 1080, zakres widzenia 82 x 82°
Moduł do rejestracji danych	smartfon OnePlus 8 Android
Nagrywanie dźwięku	tak
Wyposażenie dodatkowe	zestaw soczewek korekcyjnych: -3 ÷ +3 dpt

Po zakończeniu rejestracji dane z rejestratora są wysyłane do chmury, skąd można je w każdej chwili pobrać. Do analizy danych okulograficznych wykorzystano oprogramowanie iMotions 9.0 (Rysunek 4), które jest kompatybilne z prawie wszystkimi typami eyetrackerów [11]. Platforma ta pozwala m.in. na:

- zaprojektowanie eksperymentu z wykorzystaniem różnych rodzajów bodźców: grafiki, wideo, plików audio, stron www, zrzutów ekranowych, kwestionariuszy i nagrań z kamery sceny,
- przeprowadzenie kalibracji,
- odtwarzanie i edycję nagrań,
- analizę i wizualizację wyników: mapy termiczne, mapy uwagowe, ścieżki fiksacji, obszary zainteresowań, tzw. „roje pszczół”,
- wyeksportowanie danych (surowych lub przetworzonych: fiksacji i sakad, obszarów zainteresowania (ang. Areas of Intrest - AOIs), macierzy przejść, mrugnięć).



Rysunek 4: Okno platformy iMotions 9.0 do analizy danych z eyetrackera mobilnego.

4.4. Eksperyment

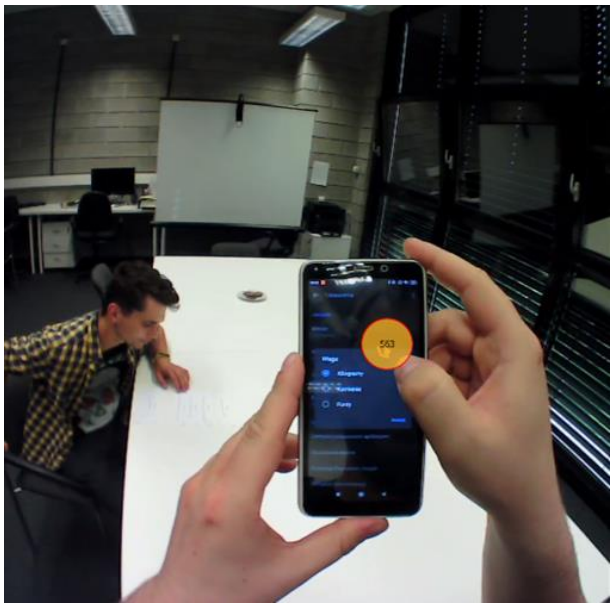
Do oceny użyteczności został opracowany eksperyment, który składał się z następujących etapów:

1. określenie celu badań,
2. wybór obiektów badań - aplikacji mobilnych,
3. przygotowanie zadań dla użytkowników,
4. pilotaż,
5. nagrywanie uczestników eksperymentu (Rysunek 4),
6. analiza zgromadzonych danych,
7. uogólnienie wyników, wnioski i rekomendacje.

Eksperyment składał się z pięciu zadań, które użytkownicy wykonywali na obu aplikacjach (Tabela 3).

Tabela 3: Zadania do wykonania dla uczestników

Lp.	Treść zadania
Z1	Sprawdź, ile kroków wykonano dnia 6 czerwca.
Z2	Sprawdź, ile spalił kalorii użytkownik podczas aktywności rozpoczętej 9 czerwca o godz. 10:00.
Z3	Zmień jednostkę miary wagi na jednostkę angielską (funty lbs).
Z4	Ustaw docelowy cel aktywności (liczba wykonanych kroków) o 1000 większy niż obecnie.
Z5	Rozpocznij trening biegu.



Rysunek 4: Przebieg eksperymentu.

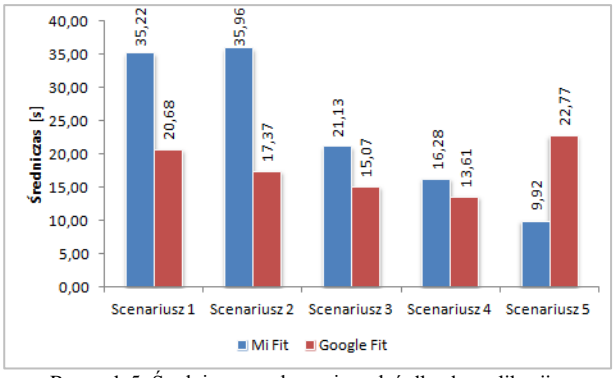
Przebieg sesji badawczej wyglądał następująco:

1. Poinformowanie uczestnika o celu badania oraz poinstruowanie go, jak ma się zachowywać podczas badania.
2. Wyrażenie zgody przez uczestnika na udział w eksperymencie.
3. Wypełnienie krótkiej ankiety dotyczącej znajomości testowanych aplikacji oraz zebranie danych metrycznych.
4. Przeprowadzenie kalibracji.
5. Nagrywanie uczestników podczas wykonywania zadań.
6. Wypełnienie ankiety diagnozującej zadowolenie użytkownika po interakcji z testowanymi aplikacjami.

5. Wyniki badań

5.1 Pomiar czasów realizacji zadań

Rysunek 3 przedstawia średnie czasy wykonywania zadań podczas obsługi dwóch testowanych aplikacji: Mi Fit oraz Google Fit. W przypadku zadań Z1 – Z4 uczestnicy szybciej wykonali polecenia posługując się aplikacją Google Fit. Natomiast obsługa oprogramowania Mi Fi wymagała krótszego czasu podczas realizacji zadania 5. Poniższy wykres ukazuje duże różnice w średnich czasach wykonania zadania pierwszego, drugiego i piątego dla obu aplikacji. Wyniosły one odpowiednio 14,54, 18,59 i 12,85 sekund.



Rysunek 5: Średni czas wykonania zadań dla obu aplikacji.

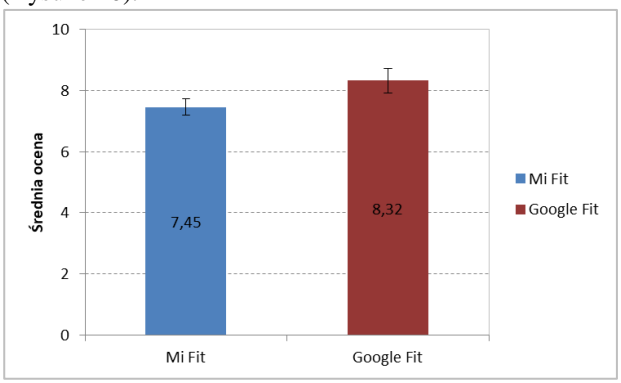
5.2 Badanie kwestionariuszowe – analiza satysfakcji

Po przeprowadzeniu badań eyetrackingowych uczestnicy wypełniali w skali 10-cio stopniowej ankietę, z której uśrednione wyniki zostały przedstawione w Tabeli 4.

Tabela 4: Średnia ocen badanych aplikacji

Pytanie	Średnia ocena	
	Mi Fit	Google Fit
Ocena interfejsu graficznego aplikacji	7,38	8,56
Ocena intuicyjności aplikacji	7,78	7,89
Ocena zrozumiałości aplikacji	7,78	8,22
Łatwość w obsłudze aplikacji	7,33	9,00
Komfort z używania aplikacji	7,33	8,00
Ocena aplikacji pod kątem oferowanych funkcji	7,11	8,22

Badana grupa jednomyślnie lepiej oceniła aplikację Google Fit. Wszystkie średnie oceny były wyższe dla tej aplikacji. Największe różnice w ocenie dotyczyły łatwości w obsłudze, wizualnej strony interfejsu oraz oferowanych funkcjonalności. Biorąc pod uwagę średnią wszystkich ocen, aplikacja Google Fit uzyskała wynik 8,32 (SD=0,41), natomiast Mi Fit 7,45(SD=0,27) (Rysunek 6).



Rysunek 6: Ocena satysfakcji użytkowników aplikacji Mi Fit i Google Fit.

5.3 Identyfikacja i analiza błędów

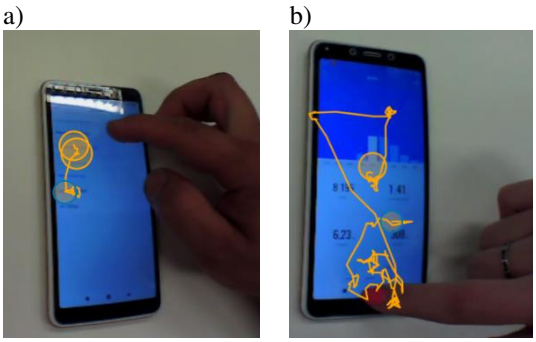
Wszyscy użytkownicy wykonali zadania poprawnie, jednak podczas ich realizacji nie ustrzegli się błędów. Tabela 5 zawiera listę błędów, które wystąpiły podczas realizacji zadań na obu testowanych aplikacjach.

Tabela 5: Identyfikacja błędów podczas interakcji z aplikacjami

Aplikacja Mi Fit	
Z1	• Zapomnienie lub niezrozumienie treści zadania • Trudność w podjęciu decyzji jaką opcję wybrać • Tapnięcie w niewłaściwym miejscu • Zgubienie się w aplikacji
Z2	• Zgubienie się w aplikacji • Trudność w podjęciu decyzji jaką opcję wybrać • Tapnięcie w niewłaściwym miejscu
Z3	• Zapomnienie lub niezrozumienie treści zadania • Wybór nieodpowiedniej opcji
Z4	• Wybór nieodpowiedniej opcji
Z5	-
Aplikacja Google Fit	
Z1	• Trudność w podjęciu decyzji jaką opcję wybrać
Z2	• Tapnięcie w niewłaściwym miejscu
Z3	• Zapomnienie lub niezrozumienie treści zadania
Z4	• Tapnięcie w niewłaściwym miejscu
Z5	• Trudność w podjęciu decyzji jaką opcję wybrać • Tapnięcie w niewłaściwym miejscu • Zgubienie się w aplikacji

Najwięcej błędów przydarzyło się uczestnikom badań podczas interakcji z aplikacją Mi Fit (10 błędów). Natomiast w przypadku Google Fit wystąpiło siedem błędów. Najczęściej pojawiającym się błędem było tapnięcie w niewłaściwym miejscu (pięć błędów: Mi Fit – dwa, Google Fit – trzy). Drugim najczęściej występującym problemem była trudność w podjęciu decyzji, którą opcję wybrać, żeby dojść do celu.

Rysunek 7 prezentuje dwa przykłady krótkich fragmentów nagrań wideo, które pokazują momenty, kiedy użytkownicy mieli problem z poprawnym wykonaniem zadań. W pierwszym przykładzie (a) było to tapnięcie w niewłaściwym miejscu, natomiast w drugim (b) zgubienie się w aplikacji.



Rysunek 7: Przykłady problemów z użytecznością.

6. Wnioski

Pozytywne doświadczenie użytkowników ma szczególnie duże znaczenie w przypadku aplikacji mobilnych, które komunikują się poprzez interfejs graficzny o niewielkich rozmiarach i dysponują ograniczonymi zasobami podczas przetwarzania danych. Dlatego też oprogramowanie tego typu powinno być dokładnie i wieloaspektowo przetestowane przed ukazaniem się na rynku. Jedną z technik, która jest obecnie coraz częściej stosowana jako technika badawcza jest eyetracking. Dostarcza ona dokładnych danych na temat doświadczeń użytkownika w pracy z aplikacją. Wgląd w działanie aplika-

cji i obserwację oprogramowania oczami użytkownika pozwala na wykrycie problemów związanych z użytecznością.

W ramach tej pracy do identyfikacji problemów podczas wykonywania zadań na aplikacjach testowych wykorzystano technologię eyetrackingową. Przy jej pomocy, na podstawie pomiarów czasów realizacji zadań, analizy problemów występujących podczas obsługi aplikacji oraz wyników ankiet aplikacja Google Fit została wyżej oceniona niż Mi Fit czyli druga z testowanych aplikacji. Uzyskane wyniki potwierdzają tezę, że większa szybkość obsługi aplikacji i mniejsza liczba popełnianych przy tym błędów wpływają pozytywnie na satysfakcję użytkowników.

Literatura

[1] S. Ambler, User Interface Design: Tips and Techniques, Toronto: Cambridge University Press, 2000.

[2] J. Gao, X. Bai, W.T. Tsai, T. Uehara, Mobile Application Testing: A tutorial, (2014) 46-55.

[3] J. Seo, Y. Choi, J. Kwak, H. Kim, Y. Yoon, Mobile device of bangle type, control method thereof and user interface (UI) display method, Google Patents, 2017.

[4] K. R. Evenson, M. M. Goto, R. D. Furberg , Systematic review of the validity and reliability of consumer-wearable activity trackers, International Journal of Behavioral Nutrition and Physical Activity 12(159) 2015, https://doi.org/10.1186/s12966-015-0314-1.

[5] B. Benet, Quantifying Sleep Quality with Smartband Technology, 2015. DOI: 10.13140/RG.2.1.2943.3448.

[6] J. Park, Hyun-Seo Hwang, Il-Young Moon, Study of Wearable Smart Band for a User Motion Recognition System, International Journal of Smart Home 8(5) (2014) 33-44. DOI: 10.14257/ijsh.2014.8.5.04.

[7] F. Fotouhi-Ghazvini, S. Abbaspour, Wearable Wireless Sensors for Measuring Calorie Consumption, Journal of Medical Signals and Sensors 10(1) (2020) 19-34. DOI: 10.4103/jmss.JMSS_15_18.

[8] A. Bojko, Eye tracking the user experience: A practical guide to research, Brooklyn, NY: Rosenfeld Media, 2013.

[9] M. Borys, M. Milosz, Mobile Application Usability Testing in Quasi-Real Conditions. A Case Study of a Mobile Eye Tracker, 8th International Conference on Human System Interactions, IEEE (2015) 381-387.

[10] Pupil Labs, Pupil Invisible. Technical Specs & Performance, https://pupil-labs.com/products/invisible/tech-specs/, [10.09.2021].

[11] iMotions, https://imotions.com/, [10.09.2021].

[12] M. Mussgnug, Q. Lohmeyer, M. Meboldt, Raising Designers' Awareness of User Experience by Mobile Eye Tracking Records, DS 78: Proceedings of the 16th International Conference on Engineering and Product Design Education (E&PDE14), University of Twente, The Netherlands (2014) 99-104.

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 356–361
Received: 15 September 2021
Accepted: 04 October 2021

Performance comparison of programming interfaces on the example of REST API, GraphQL and gRPC

Porównanie wydajności interfejsów programistycznych na przykładzie REST API, GraphQL i gRPC

Mariusz Śliwa*, Beata Pańczyk

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The article presents a comparison of the performance of three ways of implementing programming interfaces used in web applications - REST, GraphQL and gRPC. For the purposes of the research, three applications were developed, which were made in each of the indicated technologies and with the same functionalities. The applications were used for performance tests carried out with the use of the k6 tool. The applications are used to measure the execution time, performance and volume of processed data during display and adding operations. The obtained results allowed for the conclusion that the best interface in terms of performance (measured as the number of transactions per second) and server response time is REST. However, in terms of the smallest data volume, gRPC is the best choice.

Keywords: REST; gRPC; GraphQL; performance testing

Streszczenie

W artykule przedstawiono porównanie wydajności trzech sposobów realizacji interfejsów programistycznych stosowanych w aplikacjach webowych – REST, GraphQL oraz gRPC. Na potrzeby badań opracowano trzy aplikacje, które zostały wykonane w każdej ze wskazanych technologii i o takich samych funkcjonalnościach. Aplikacje wykorzystano do testów wydajnościowych, przeprowadzonych z użyciem narzędzia k6. Aplikacje zastosowano do zmierzenia czasu wykonania, wydajności i objętości przetwarzanych danych podczas operacji wyświetlania oraz dodawania rekordów. Uzyskane wyniki pozwoliły na sformułowanie wniosku, że najlepszym interfejsem pod względem wydajności (mierzonych jako liczba wykonywanych transakcji na sekundę) oraz czasu odpowiedzi serwera jest REST. Natomiast pod względem najmniejszej objętości danych, najlepszym wyborem jest gRPC.

Słowa kluczowe: REST; gRPC; GraphQL; testy wydajnościowe

*Corresponding author

Email address: mariusz.sliwa@pollub.edu.pl (M. Śliwa)

1. Wstęp

1.1. Interfejsy programistyczne

Najbardziej popularne są obecnie aplikacje, które korzystają z komunikacji przez sieć. Takie aplikacje potrzebują pośredników w celu komunikacji między klientem oraz serwerem. W celu zdefiniowania semantyki i składni komunikatów przekazywanych przez sieć, powstały różne interfejsy programowania aplikacji, realizowane przy użyciu protokołów oraz specyfikacji, szerzej znane jako API (ang. Application Programming Interface) [1]. Zaletą takiego podejścia do projektowania architektury aplikacji jest umożliwienie szerokiej gamie fizycznych urządzeń klienckich i typów aplikacji, interakcji z daną aplikacją. Jedno API może być używane nie tylko do komputerów PC, ale także do telefonów komórkowych i urządzeń IoT (ang. Internet of Things). Komunikacja nie ogranicza się do interakcji między ludźmi a aplikacjami. Przez ostatnie lata pojawiały się różne interfejsy programistyczne, każdy z nich posiadał własne wzorce standaryzacji wymiany danych. Jak dotąd najczęściej używanym API jest REST (ang. Representational state transfer) [2], który zastąpił SOAP (ang. Simple Object Access Protocol). REST posiada wiele zalet, ale mimo to nie nadaje się do każdego rozwiązania. Właśnie dlatego powstały alternatywy, które mają na celu zastąpienie niedoskonałego REST API w różnych zastosowaniach. W 2015 roku pojawiło się konkurencyjne rozwiązanie o nazwie GraphQL, stworzone przez programistów Facebooka, które w krótkim czasie zdobyło bardzo dużą popularność. Kolejnym rozwiązaniem jest gRPC [3] (ang. general-purpose Remote Procedure Calls), które również pojawiło się w 2015 roku, natomiast twórcami tego interfejsu są programiści z Google.

Każde z tych API działa w odmienny sposób oraz posiada swoje zalety w różnych zastosowaniach, niestety w niektórych przypadkach programiści niepoprawnie decydują o wyborze interfejsu obsługującego system złożony z aplikacji, a ma to istotne znaczenie w przypadku stabilności oraz wydajności systemu. Dlatego należy rozpatrzeć mocne i słabe strony każdej z opcji, którą można potencjalnie zastosować w tworzonej aplikacji.

1.2. Przegląd literatury

Celem artykułu "REST vs gRPC vs GraphQL" [4] było porównanie różnic między REST a gRPC i Gra-

phQL. Omówiono zalety i wady każdego podejścia wraz z przypadkami użycia, które pozwoliłyby wybrać, które podejście najlepiej odpowiada potrzebom programistów. Zalety, jakie według autora posiada REST, to łatwa skalowalność, łatwa integralność, wydajność; natomiast wadami są m.in. trudności w trzymaniu się koncepcji HATEOAS (ang. Hypermedia As The Engine Of Application State), czyli ograniczenie architektury aplikacji REST z pomocą hipermediów, brak jednolitego stylu dokumentacji oraz wersjonowanie API. W przypadku gRPC zaletami są silne typowanie, wbudowany generator kodu, używanie protokołu HTTP/2, automatycznie generowanie dokumentacji, natomiast wadami są brak możliwości używania cache HTTP ze względu na używanie jedynie metody POST oraz brak wbudowanego narzędzia do debugowania oraz testowania. Zaletami GraphQL są łatwość w testowaniu, możliwość dokładnego wskazania czego potrzebuje klient od serwera, możliwość pobierania danych z różnych źródeł z użyciem tylko jednego zapytania, natomiast wadami są m.in. brak wbudowanego cache, problemy z zapytaniami cyklicznymi, ryzyko ujawnienia modelu danych. W artykule wskazano, że wybór sposobu realizacji API jest uzależniony od wymagań stawianych przed taką aplikacją i nie jest możliwe jednoznaczne wskazanie, który sposób realizacji API jest najbardziej odpowiedni.

"APIs REST, GraphQL or gRPC – Who wins this game?" [5] - artykuł, w którym autor porównywał różne sposoby realizacji aplikacji definiując wstępnie trzy główne przypadki użycia interfejsów. Były to: "Experience API" - interfejs do użytku przez aplikacje i urządzenia klienckie na potrzeby cyfrowych doświadczeń np. aplikacja mobilna łącząca się z wewnętrznym serwerem odpowiedzialnym za przechowywanie danych , "Open API" - otwarty interfejs służący do integracji z zewnętrznymi aplikacjami oraz "Internal API" - interfejs wewnętrzny służący do komunikacji między mikroserwisami. W celu porównania interfejsów, autor stworzył tabele z kryteriami, w których przyznawał wartości punktowe dla każdego API zależnie od spełniania każdego z kryteriów. W przypadku Experience API największą liczbę punktów uzyskał GraphQL ze względu na szybkość odpowiedzi oraz małą liczbę wykonywanych zapytań, W Open API najwięcej punktów miał REST ze względu na dużą możliwość ponownego użycia części aplikacji oraz łatwość w używaniu. W Internal API najlepiej poradził sobie gRPC dzięki dobrej skalowalności oraz optymalizacji zapytań.

W pracy naukowej "Performance analysis of Web Services: Comparison between RESTful & GraphQL web services" [6] autor przetestował wydajność interfejsu REST API oraz GraphQL mierząc czas odpowiedzi na zapytanie i rozmiar odpowiedzi. W pilotażowym teście okazało się, że REST jest szybszy dla prostszych strukturalnie zapytań, takich jak pobieranie informacji tylko z jednego źródła lub tabeli. Różnica w wydajności w zakresie czasu odpowiedzi rosła wykładniczo wraz z rozmiarem bazy danych. Pobierając więcej informacji, GraphQL generował wolniejsze odpowiedzi. Różnica zmierzona w dwóch eksperymentach przeprowadzonych w tych badaniach wynosiła 64-115%. W przypadku większych baz danych z możliwością tworzenia bardziej złożonych zapytań, GraphQL uzyskało bardzo dobry wynik. W drugim eksperymencie tych badan różnica wynosiła już tylko 25% . Przeprowadzono drugi test w celu potwierdzenia tych ustaleń. Ten test implementował to samo pobieranie danych procedury, jednak zamiast 100 iteracji, test ten powtórzono 1000 razy, co pokazuje, ze GraphQL jest o 51,75% szybszy niż usługa REST. Dzięki możliwości wybierania danych z dużo większa precyzja, całkowite rozmiary pakietów przesyłanych miedzy klientem i serwer można było zmniejszyć, prawdopodobnie skracając czas ładowania. Według autora precyzyjny wybór danych mógł również zmniejszyć prace jaka musi wykonać klient, ponieważ nie wymaga tyle logiki przetwarzającej dużej ilości danych pochodzących z serwera, a raczej przetwarza tylko te dane, które są potrzebne w tym momencie.

1.3. Cel badań

W Internecie można znaleźć wiele artykułów na temat zalet i wad REST API, GraphQL i gRPC. Mimo to wciąż niewiele artykułów pokazuje jaki tak naprawdę wpływ na wydajność mają różne sposoby realizacji komunikacji między aplikacjami. Celem tej pracy jest przeprowadzenie dokładnej analizy porównawczej wydajności tytułowych rozwiązań komunikacji. Zakres badań obejmuje: opracowanie aplikacji testowych, przygotowanie eksperymentu, dobór narzędzia diagnostycznego, zrealizowanie zautomatyzowanych testów wydajnościowych oraz opracowanie i interpretacja wyników.

Postawiono następującą hipotezę badawczą:

„Aplikacje opierające się na gRPC zużywają najmniejszą ilość danych oraz są najbardziej wydajne”.

2. Metody badań

2.1. Przypadki testowe

W celu przetestowania różnic między interfejsami programistycznymi - utworzono aplikacje w REST, GraphQL oraz gRPC z użyciem platformy .NET 5.0 [7]. Każda z tych aplikacji posiadała takie same funkcjonalności i służyła w praktyce do ewidencji autorów oraz ich książek. Każda z aplikacji została przetestowana pod względem czasu przetwarzania operacji, liczby transakcji na sekundę oraz objętości przetwarzanych danych.

2.2. Środowisko testowe

Do przeprowadzenia badań wykorzystano dwa komputery. Ich parametry znajdują się w Tabeli 1.

Tabela 1: Parametry pierwszego komputera

Procesor	AMD Ryzen 7 3800X 8 rdzeni, 16 procesorów logicznych, @ 4,4 GHz
Pamięć RAM	32 GB
Karta sieciowa	TP-LINK 802.11ac Network Adapter
System operacyjny	Windows 10 Education N

Do testów wydajnościowych użyto narzędzia k6 v0.33.0 [8]. Jest to narzędzie typu open source napisane w języku Go. k6 osadza silnik ECMAScript umożliwiając użytkownikom pisanie testów w JavaScript. Oprogramowanie to umożliwia automatyczne analizowanie wydajności różnych usług pod dużym obciążeniem z symulowaniem równoczesnej pracy wielu klientów.

2.3. Scenariusze badawcze

Dla każdej aplikacji opracowano eksperymenty, podczas których serwer przetwarzał zróżnicowaną ilość danych. Do testów użyto operacje pobierania oraz dodawania, ze względu na najczęstsze wykorzystywanie tych operacji w aplikacjach. Badanie pobierania zostało przeprowadzone z użyciem trzech zmiennych:

- liczba symulowanych użytkowników wysyłających zapytania (1, 10, 50),
- liczba zapytań wysłanych przez jednego użytkownika (50),
  liczba rekordów pobieranych z serwera (11, 111, 3112) oraz ich rozmiar.

W przypadku operacji dodawania danych, badanie zostało przeprowadzone z wykorzystaniem różnej liczby użytkowników wykonujących po 50 zapytań do serwera.

Do porównania wydajności aplikacji, wykorzystujących różne interfejsy programistyczne posłużono się kryteriami:

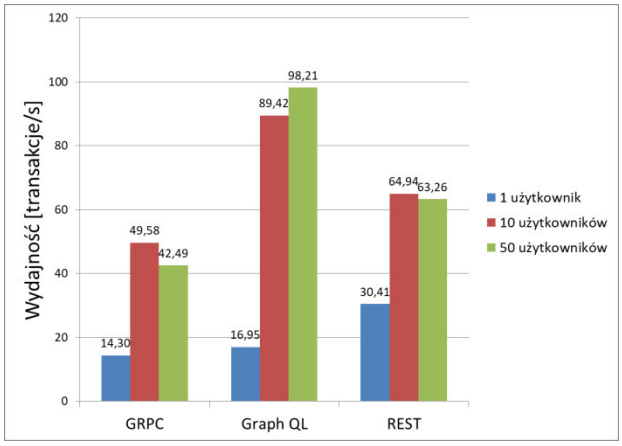
- średni czas przetworzenia żądań wyrażony w milisekundach,
- rozmiar danych w megabajtach (pobieranych lub wysyłanych),
- liczba transakcji na sekundę.

3. Wyniki badań

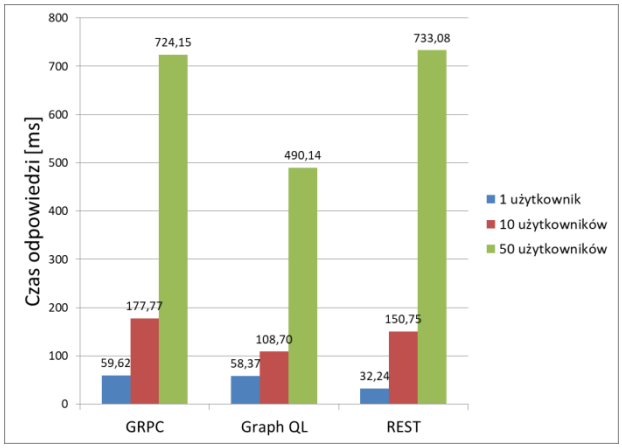
Badanie 1: *Pobranie 3112 rekordów autorów, u których w nazwie znajduje się fraza „Author - 1”*

W każdym z interfejsów, zostały przygotowane odpowiednie punkty końcowe do wysyłania zapytań o autorów, którzy w nazwie posiadali podaną przez użytkownika frazę. Zadaniem napisanych aplikacji było zwrócenie listy autorów z ich pełnymi danymi. Badanie zostało przeprowadzone dla trzech symulowanych grup użytkowników, wykonujących 50 zapytań każdy, a ich wynik został przedstawiony na rysunkach 1-3.

Na Rysunku 1 widoczne jest, że GraphQL najlepiej się sprawdza dla dużej liczby użytkowników wykonujących zapytania w jednym czasie. Porównując interfejs GraphQL z pozostałymi aplikacjami można zauważyć, że wraz ze wzrostem liczby użytkowników, zwiększa się również ilość transakcji na sekundę, co oznacza, że ten sposób realizacji dobrze się sprawdza przy pobieraniu dużej ilości danych równolegle przez wielu klientów. REST oraz gRPC w przeciwieństwie do GraphQL, w przypadku, w którym dane są pobierane przez 50 użytkowników, tracą na pod względem wydajności oraz czasu odpowiedzi.

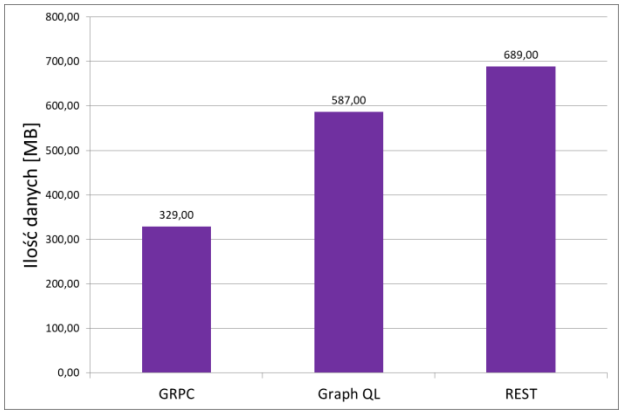


Rysunek 1: Wydajność interfejsów przy 3112 pobieranych rekordach.



Rysunek 2: Średni czas odpowiedzi interfejsów przy 3112 pobieranych rekordach.

Rysunek 3 przedstawia ilość łącznie otrzymanych danych dla 50 użytkowników wykonujących po 50 zapytań i pokazuje, że gRPC jest najbardziej optymalne pod względem jak najmniejszej objętości odebranych danych.

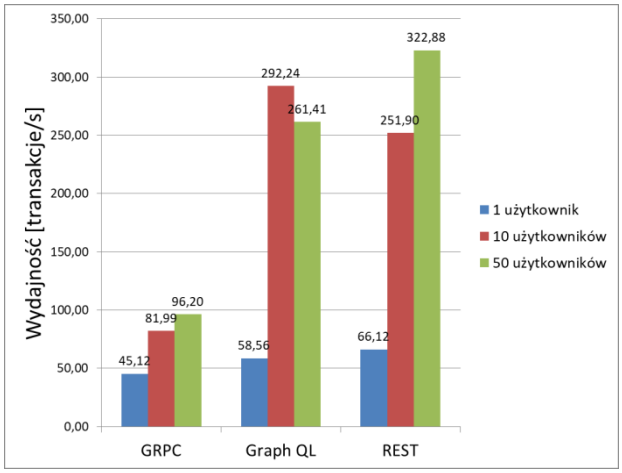


Rysunek 3: Ilość otrzymanych danych z punktów końcowych interfejsów przy 3112 pobieranych rekordach.

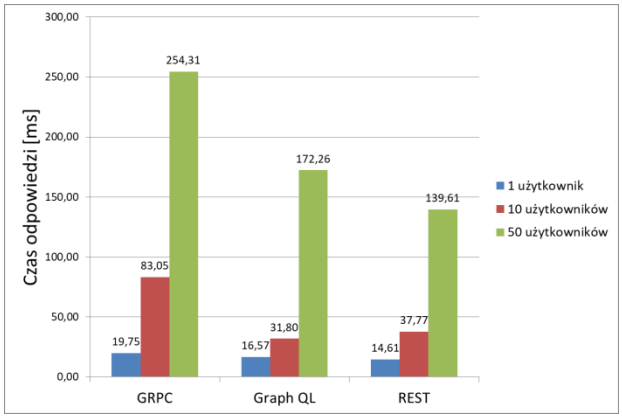
Badanie 2: *Pobranie 111 rekordów autorów, u których w nazwie znajduje się fraza „Author - 100”*

W tym badaniu, tak samo jak w poprzednim, zasymulowano wysyłanie zapytania o pełne dane autorów, których nazwa zawierała frazę „Author - 100”. Badanie zostało przeprowadzone dla trzech grup symulowanych użytkowników, a wyniki przedstawiają Rysunki 4-6.

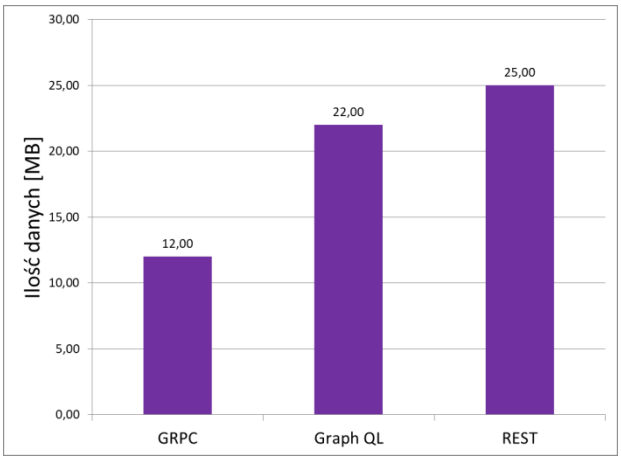
W tym teście widoczny jest wzrost wydajności interfejsu REST API względem poprzedniego badania, dzięki czemu wartości transakcji na sekundę oraz średniego czasu odpowiedzi zbliżyły się do wyników w GraphQL, a w przypadku 50 użytkowników wartość wydajności przekroczyła 300 transakcji na sekundę, czego nie udało się osiągnąć pozostałym aplikacjom. Rysunek dotyczący otrzymywanych danych ponownie wskazuje, że optymalnym rozwiązaniem pod tym względem jest gRPC, natomiast GraphQL oraz REST przesyłają dwukrotnie więcej danych.



Rysunek 4: Wydajność interfejsów przy 111 pobieranych rekordach.



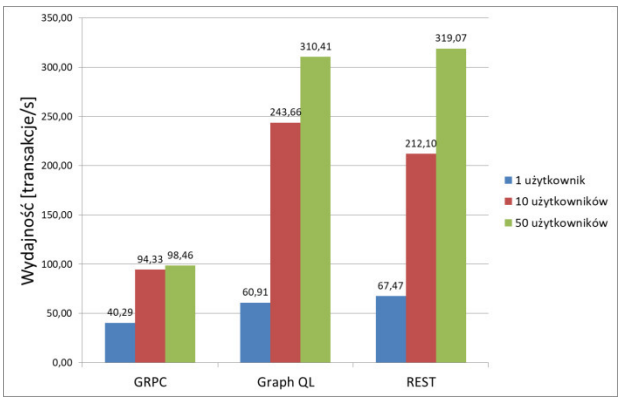
Rysunek 5: Średni czas odpowiedzi interfejsów przy 111 pobieranych rekordach.



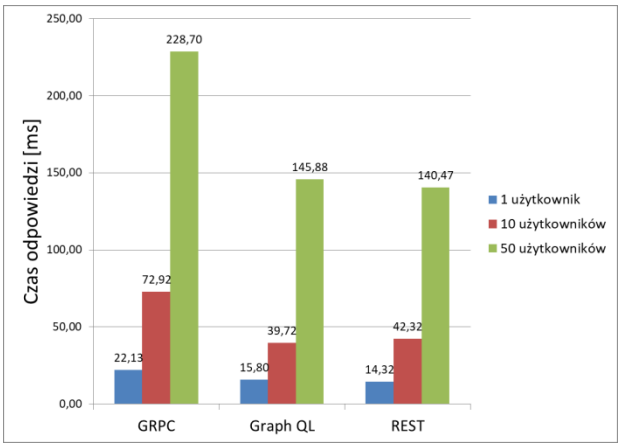
Rysunek 6: Ilość otrzymanych danych z punktów końcowych interfejsów przy 111 pobieranych rekordach.

Badanie 3: *Pobranie 11 rekordów autorów, u których w nazwie znajduje się fraza „Author - 1000”*

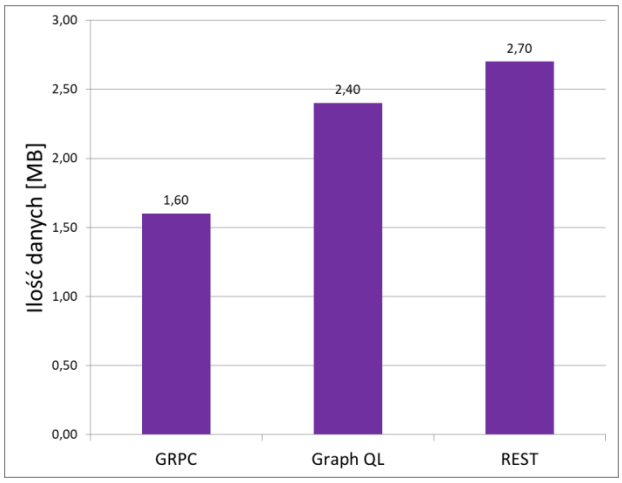
W ostatnim badaniu dotyczącym pobierania danych zasymulowano wysyłanie zapytania o pełne dane autorów, których nazwa zawierała frazę „Author - 1000”. Każda aplikacja miała zwrócić 11 rekordów z bazy danych. Badanie zostało przeprowadzone dla trzech grup symulowanych użytkowników, a wyniki pokazane są na Rysunkach 7-9.



Rysunek 7: Wydajność interfejsów przy 11 pobieranych rekordach.



Rysunek 8: Średni czas odpowiedzi interfejsów przy 11 pobieranych rekordach.

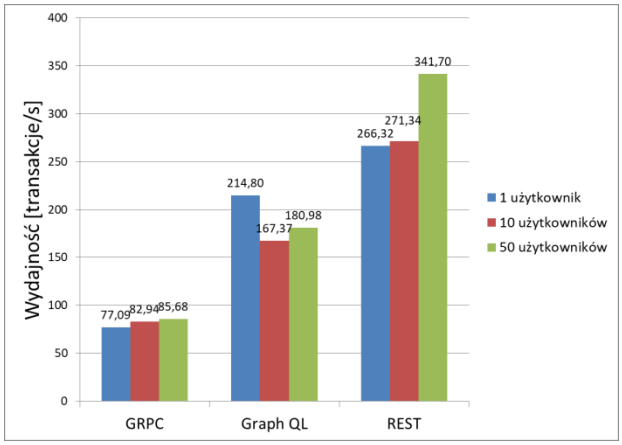


Rysunek 9: Ilość otrzymanych danych z punktów końcowych interfejsów przy 11 pobieranych rekordach.

Przy pobieraniu danych z serwera, po raz kolejny najgorsze wyniki uzyskał interfejs gRPC, który pod względem wydajności w sytuacjach wysyłania zapytania przez 10 oraz 50 użytkowników, jest słabszy 3 krotnie od pozostałych interfejsów. Aplikacje napisane z użyciem GraphQL oraz REST API ponownie zachowują zbliżone wartości wydajności oraz czasu odpowiedzi. Tak jak w poprzednich testach, rysunek 9 dotyczący otrzymanych danych, wskazuje na ciągłą przewagę gRPC nad pozostałymi interfejsami pod względem jak najmniejszą ilości otrzymanych danych.

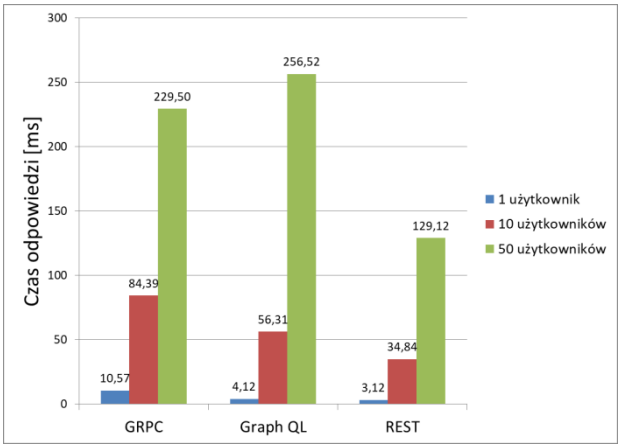
Badanie 4: *Dodanie autora do bazy danych*

Test dotyczący dodania autora do bazy danych zasymulowano poprzez wysyłanie danych zawierających nazwę oraz kraj autora. Każda z aplikacji po pomyślnym dodaniu autora zwracała odpowiedź zawierającą dodanego autora. Badanie zostało przeprowadzone dla trzech grup symulowanych użytkowników, a wyniki zostały przedstawione na rysunkach 10-12.

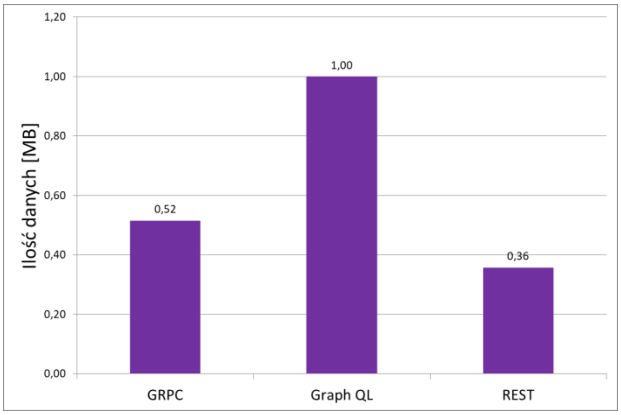


Rysunek 10: Wydajność interfejsów przy dodawaniu autora.

W przypadku dodawania autora do bazy danych, najlepszą wydajność uzyskał REST API, powiększając swoją przewagę nad pozostałymi interfejsami. Wraz ze wzrostem liczby użytkowników, uzyskał on dwukrotną przewagę nad GraphQL oraz czterokrotną nad gRPC. Pod względem czasu odpowiedzi, REST API uzyskał dwukrotnie lepszy wynik. Również w kwestii ilości wysłanych danych, REST API osiągnął wynik lepszy nawet od gRPC, który w testach pobierania danych był najbardziej optymalny.



Rysunek 11: Średni czas odpowiedzi interfejsów przy dodawaniu autora.



Rysunek 12: Ilość otrzymanych danych z punktów końcowych interfejsów przy dodawaniu autora.

4. Wnioski

Badania zrealizowane zostały na podstawie trzech aplikacji utworzonych w REST, GraphQL oraz gRPC, które posiadały maksymalnie zbliżone do siebie funkcjonalności. Pomiary czasu odpowiedzi, wydajności oraz ilości przesyłanych danych umożliwiło oprogramowanie k6. Zrealizowane badania aplikacji wskazały znaczne różnice między wykorzystanymi technologiami. Operacja pobierania danych, której w tym artykule poświęcono najwięcej uwagi wymagała pobrania wielu rekordów z bazy danych, dzięki temu można było sprawdzić jak każda z technologii radzi sobie z takim zadaniem. W przypadku dużej ilości pobieranych danych najlepiej sprawdzał się interfejs GraphQL, natomiast w przypadku średniej i małej wartości pobranych rekordów, wyniki wydajności REST API oraz GraphQL były do siebie bardzo zbliżone z delikatną przewagą REST API. Wyniki testów dotyczące REST oraz GraphQL pokrywają się z częścią pracy dotyczącej mniejszej ilości danych, którą wykonał A. F. Helgason [6], w której również REST miał lekką przewagę nad GraphQL w sytuacji,

gdy zapytania były prostsze strukturalnie. Niestety aplikacja napisana z użyciem interfejsu gRPC nie zdołała dorównać pozostałym przykładom, osiągając wyniki wydajności 2-4 krotnie słabsze pod względem wydajności oraz czasu odpowiedzi. W przypadku ilości danych pobranych podczas zapytań do serwera, gRPC wskazało swoją zaletę, jaką jest dwukrotnie mniejsza objętość danych w porównaniu do GraphQL oraz REST API. Testy poświęcone metodzie dodania autora do bazy danych podkreśliły przewagę REST API nad innymi rozwiązaniami. Interfejs ten osiągnął w kryterium wydajności oraz czasu odpowiedz najlepsze wyniki, osiągając dwukrotnie lepsze rezultaty od pozostałych interfejsów.

W Tabeli 2 podsumowano wyniki badań z wykorzystaniem punktacji w skali od 0 do 2 (im więcej punktów tym lepiej). W tabeli 2 zestawiono kryteria uwzględnione w badaniach, sumę punktów uzyskanych w danym kryterium oraz całkowitą sumę wszystkich punktów.

Tabela 2: Ocena punktowa technologii

Kryterium/Interfejs	gRPC	GraphQL	REST
Wydajność przy dużej ilości danych	0	2	1
Wydajność przy średniej ilości danych	0	1	2
Wydajność przy małej ilości danych	0	1	2
Wydajność przy dodaniu danych	0	1	2
Suma punktów kryterium wydajności	0	5	7
Czas odpowiedzi przy dużej ilości danych	0	1	0
Czas odpowiedzi przy średniej ilości danych	0	1	1
Czas odpowiedzi przy małej ilości danych	0	1	1
Czas odpowiedzi przy dodaniu danych	1	0	2
Suma punktów kryterium czasu odpowiedzi	1	3	4
Rozmiar odpowiedzi z serwera przy dużej ilości danych	2	1	0
Rozmiar odpowiedzi z serwera przy średniej ilości danych	2	1	0
Rozmiar odpowiedzi z serwera przy małej ilości danych	2	1	0
Rozmiar danych przy dodaniu danych	1	0	2
Suma punktów kryterium rozmiaru odpowiedzi	7	3	2
Suma punktów	8	11	13

W efekcie przeanalizowania wyników badań, otrzymane rezultaty udowadniają hipotezę na temat najmniejszego użycia danych przez gRPC, natomiast przypadek największej wydajności nie został potwierdzony.

Uwzględniając otrzymane wyniki badań, można stwierdzić, że ostateczna decyzja co do wyboru interfejsu programistycznego jest bardziej skomplikowana niż mogło by się wydawać. Przy wybieraniu sposobu w jakim będzie realizowana aplikacja musimy kierować się rozmiarem danych przesyłanych między użytkownikiem a aplikacją serwera, wydajnością urządzeń oraz liczbą użytkowników. Istnieją też kryteria, które wiążą się np. z wygodą tworzenia aplikacji przez programistów. Ze względu na swoją popularność i wygodę tworzenia kodu przez programistów, REST API jest najczęściej wybieranym rozwiązaniem, natomiast jak ukazały powyższe badania, nie zawsze jest to najlepsze rozwiązanie. W tym celu przedstawiono możliwości innych interfejsów programistycznych poprzez przetestowanie przykładowej aplikacji do ewidencji autorów oraz ich książek. Według artykułu Rafaela Rocha [5], aplikację tą można sklasyfikować jako "Experience API". Do tego typu aplikacji według autora najbardziej optymalnym rozwiązaniem byłby GraphQL, a najgorszym REST API, natomiast w pracy tej nie tylko aspekt wydajności był brany pod uwagę, ale również takie szczegóły jak możliwości monitorowania aplikacji. W przeciwieństwie do pracy Rocha, w powyższych badaniach wykazano, że REST API jest najlepsze, ale w przypadku dużej ilości danych równie dobrze sprawdza się GraphQL, natomiast w sytuacji wymagania małej objętości danych warto rozważyć użycie gRPC.

Literatura

[1] What is an API? (Application Programming Interface), https://www.mulesoft.com/resources/api/what-is-an-api, [10.09.2021]

[2] B. M. Balachandar, RESTful Java Web Services: A pragmatic guide to designing and building RESTful APIs using Java, 3rd Edition, Packt Publishing, 2017.

[3] Dokumentacja gRPC, https://grpc.io/docs, [01.03.2021]

[4] A. Tuban, REST vs gRPC vs GraphQL, https://technologyrivers.com/blog/rest-vs-grpc-vs-graphql, [01.03.2021]

[5] R. Rocha, APIs REST, GraphQL or gRPC – Who wins this game?, https://www.sensedia.com/post/apis-rest-graphql-or-grpc-who-wins-this-game, [01.03.2021]

[6] A. F. Helgason, Performance analysis of Web Services: Comparison between RESTful & GraphQL web services, University of Skövde, http://his.diva-portal.org/smash/record.jsf?pid=diva2:1107850, 2017, [01.03.2021]

[7] Wprowadzenie do .NET, https://docs.microsoft.com/en-us/dotnet/core/introduction, [10.09.2021]

[8] Dokumentacja JavaScript-owego API biblioteki k6, https://k6.io/docs/javascript-api, [10.09.2021]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 362–366
Received: 17 September 2021
Accepted: 20 Ocober 2021

Digital entertainment in the face of COVID-19

Rozrywka cyfrowa w obliczu COVID-19

Adam Jarszak*

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The work analyzed the impact of the Covid-19 pandemic on digital entertainment. The focus is on three types of digital entertainment: computer games, virtual reality and streaming, respectively. The first two types of entertainment were analyzed based on data from the Steam platform, while streaming was analyzed based on Twitch. Data were collected from 2019, i.e. the period before the pandemic, and 2020, i.e. the time of the pandemic. Then the data from both of these years were compiled for analysis. The study particularly focused on events related to the pandemic, such as the March declaration of the Covid-19 virus as a pandemic by the WHO or the holiday period in which the restrictions were reduced.

Keywords: digital entertainment; covid-19

Streszczenie

Praca zajęła się analizą wpływu pandemii Covid-19 na rozrywkę cyfrową. Skupiono się na trzech rodzajach rozrywki cyfrowej są nimi kolejno gry komputerowe, wirtualna rzeczywistość oraz streaming. Dwa pierwsze rodzaje rozrywki zostały przeanalizowane na podstawie danych z platformy Steam, natomiast streaming poddano analizie na podstawie Twitcha. Pobrano dane z lat 2019 czyli okres przed pandemią oraz 2020 czyli czas panowania pandemii. Następnie dane z obu tych lat zostały zestawione do analizy. W pracy w szczególności zwrócono uwagę na wydarzenia związane z pandemią, były to między innymi marcowe ogłoszenie wirusa Covid-19 jako pandemie przez WHO lub też okres wakacyjny w którym zmniejszone zostały obostrzenia.

Słowa kluczowe: rozrywka cyfrowa; covid-19

*Corresponding author

Email address: adam.jarszak@pollub.edu.pl (A. Jarszak)

1. Introduction

From year to year, more and more people choose digital entertainment as their leisure method. So it's also not surprising that during the COVID-19 pandemic, people are turning to this type of entertainment to a much greater extent. Public places at which you can spend free time are getting closed all over the world. Every person is looking for ways to spend their free time safely. They are trying to find a replacement for the old ways of spending free time, i.e. as mentioned in the first two references in the article [1, 2].

For example, cinemas and theaters are getting replaced by platforms like: Twitch, Youtube or Netflix [1]. Another activity that got replaced is spending time outside alone or with friends. It got replaced by video games which allow you to spend time alone and in company, they enable interaction with other users, even with those living on the other side of the globe [3, 4, 5]. Video games are a great way to spend free time and deal with the stress caused by Covid-19 as mentioned in references four and five [4, 5].

The whole situation in the world offers great growth opportunities for all kinds of digital entertainment. And if it continues, it will probably change the world of digital entertainment to a great extent, but only time will tell and we need to wait for the aftereffects of the pandemic as mentioned in the first reference of the article [1].

2. Hypotheses

There are three hypotheses for the article:

a) The pandemic contributed to the increase in popularity of computer games.
b) Virtual Reality has experienced the largest percentage increase in the number of users among the types of digital entertainment studied.
c) Watch time on Twitch increased significantly during the reign of the Covid-19 virus.

3. Materials and methods

The author's work analyzes three types of digital entertainment on the global market and compares their results before and during the pandemic, i.e. year 2019 and 2020.

The first of them, which is also the main focus of work, i.e. computer games. The analysis of computer games was performed on the basis of the most popular digital video game distribution service on personal computers created by Valve, ie Steam [6]. This platform was chosen because it is the only one to keep its current results available. The statistical data was collected from two websites, ie SteamCharts [7] and SteamDB [8]. These pages were selected because Steam only provides up-to-date data, but does not save past results anywhere. However, both of these sites collect and save the results from the Steam platform, with the difference that the

SteamDB site additionally compares the results of the Steam platform with the Twitch platform [9].

The work focuses not only on ordinary computer games, it also analyzes the popularity of virtual reality, i.e. VR. It allows you to experience more than a simple video call. The user feels as if he is in the same room as other users, this gives vr a tremendous opportunity to grow during a pandemic reign [10]. In this case, as before, the analysis was performed on the basis of the Steam platform. However, this time, we have not found any website that would collect historical data from the steam platform on virtual reality. The Internet Archive [11] website was used to obtain statistical data, which allows you to view archival versions of selected websites, which allowed to download the results of virtual reality at the turn of 2019 and 2020.

The last type of digital entertainment analyzed are streaming platforms. The results from the Twitch platform were analyzed [9]. The data was collected from two pages, the first of which is TwitchTracker [12], which contains data only about Twitch. The second page, the same as in the case of Steam, i.e. SteamDB, contains a summary of Steam and Twitch data.

In addition to analyzing the results obtained directly from both of the above platforms, the analysis also included the results of the Google Trends website [13], which, as the name suggests, provides information on search trends in the Google search engine.

Short few days deviations were ignored during the analysis, but they are still present in tables and figures. This decision was made because in most cases these were regular tournaments, events or updates that only attracted users for a short time and were not caused by a pandemic.

4. Results

Results are divided into three subsections each corresponding to one of the types of analyzed digital entertainment.

4.1. Computer games

In the first subsection, the first type of analyzed digital entertainment, i.e. computer games, was analyzed. As mentioned before, the focus here is on the Steam platform because it makes most of its data available to the public.

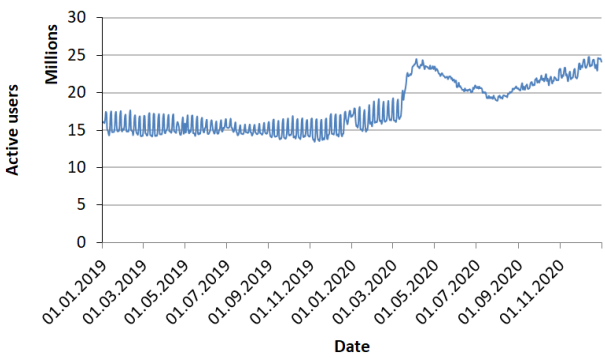


Figure 1: Active Steam Users.

Figure 1 shows two spikes in player numbers during the reign of the pandemic.

The first of them, namely the period from February to May 2020. During this period restrictions related to COVID-19 began to appear mainly in Asia. Slow growth at first, followed by a drastic spike due to WHO's declaration of COVID-19 as a pandemic. The announcement was made on March 11 [14]. This resulted in the emergence of new, more drastic restrictions around the world. The period from May to September saw a decline, it was caused by the removal and the reduction of many restrictions, which made many people go on vacation or return to work. Then, from September to December 2020, the restrictions began to return because after the holidays and after returning to school, the number of infections increased rapidly.

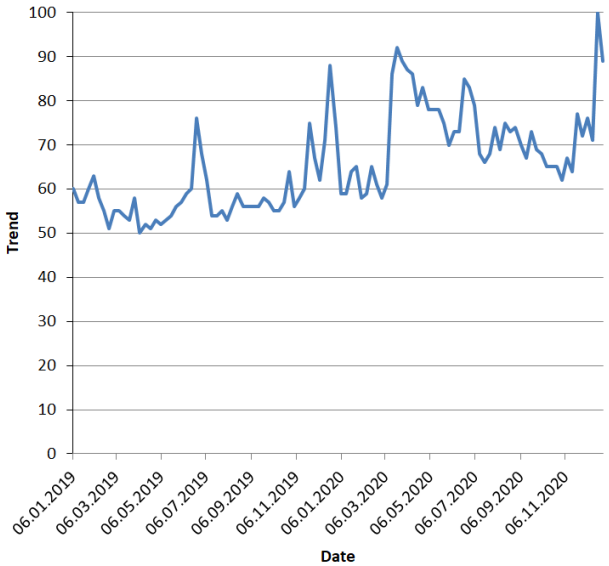


Figure 2: The trend of searching for "Steam" in the Google search engine.

Figure 2 shows the trend of searching for "Steam" in the Google search engine. The data was obtained from the Google Trends website. As with Figure 1, there are several spikes related to the pandemic. First in March, when the WHO announced the virus, followed by a downward trend until December. Besides the first there are two more jumps, one in June and one in December. Both of these months have jumps every year because the school year ends in June and many children then get a gift. Increasingly, parents choose various types of electronics, including consoles and computers. In addition, the end of the school year means holidays for young people, thanks to which they have more time to spend in front of the computer. The same applies to December because then it is Christmas, but in this case almost everyone has days off and gets gifts not only children. Therefore, the December jump is greater than the June jump.

4.2. Virtual reality

The second subsection is intended for virtual reality which is part of computer games. We also use the Steam platform to obtain the data here.

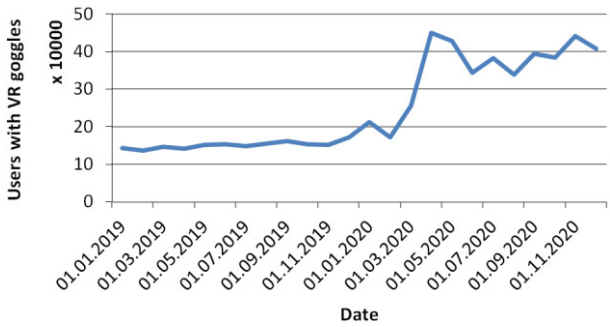


Figure 3: Estimated number of users with VR goggles.

Figure 3 shows the estimated number of Steam users with VR headsets. It was calculated by adding up the percentage of individual goggle models and then the percentage of the total number of platform users was calculated. It is estimated because Steam publishes only the percentage of users with goggles who took part in its regular monthly survey on computer hardware [15, 16].

On Figure 3 you can see one significant jump in the estimated number of users with VR goggles on Steam - March 2020. This jump is threefold, but in this case, unlike the previous figures, it is caused not only by the declaration of the Covid-19 virus as a pandemic by WHO, but also with the premiere of the long-awaited game Half-Life: Alyx exclusive for VR [16, 17]. The jump was followed by a slight decline until December. It is caused by the lack of availability of virtual reality goggles and the general decrease in the number of active players on Steam during this period, which is visible in Chart 1. At the end of 2020, a second, but much smaller, jump is visible. It was caused by the return of the availability of the goggles at the end of 2020. It could have been much larger because on October 13, 2020, new goggles from Oculus, namely Oculus Quest 2, were released for sale, which were a great success. Unfortunately, Valve began to count the number of Quest 2 goggles only from January 2021, so they were not included in the work.

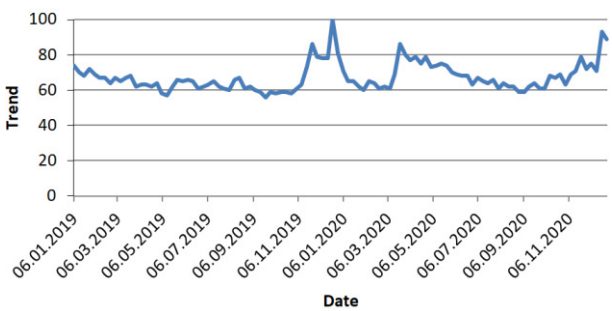


Figure 4: The trend of searching for the term "Virtual Reality" in the Google search engine.

Chart 4 shows the trend of searching for the term "Virtual reality". 3 big jumps can be noticed. The first is in November and December. It was caused by the announcement of the previously mentioned Half-Life: Alyx [16]. It is a continuation of one of the largest series in the history of computer games. The second jump took place in March then the aforementioned game was released and then there was an announcement from WHO. Then, from March to October, there was a decline and at the end of October the trend began to increase again. The re-growth was caused by the previously mentioned launch of the Quest 2 goggles. These goggles were not included in the analysis for the aforementioned reason, namely Steam did not collect data on new goggles until early 2021.

4.3. Streaming

The last subsection contains Streaming analysis.

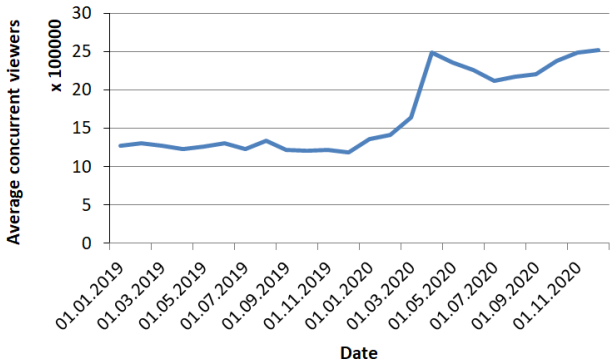


Figure 5: Average concurrent viewers for the entire Twitch platform.

The first figure in the Streaming chapter, i.e. Figure 5, contains information on the average number of simultaneous viewers on the Twitch platform. The entire year 2019 was around 1,300,000 viewers, after which in March 2020 there was a giant leap almost doubling this number. This jump, as already mentioned in many other figures, was caused by the declaration of a Pandemic by the WHO and the resulting restrictions and isolations. Then you can see a slight decrease during the holiday season. On the other hand, an upward trend has been visible since September.

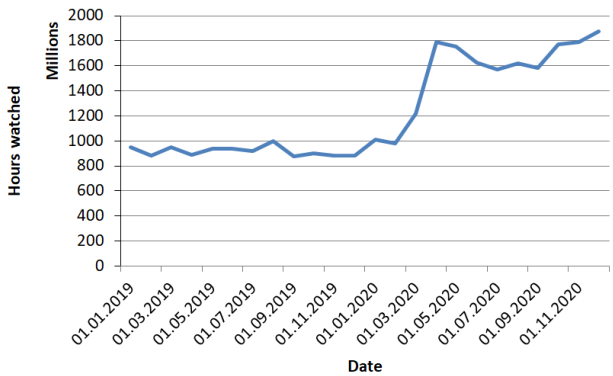


Figure 6: Twitch watch time.

Figure 6 shows Twitch watch time and is almost identical to Figure 5. In both charts we see a constant value in 2019. Then a big jump in September followed by a drop during the holiday season and a re-increase from around September.

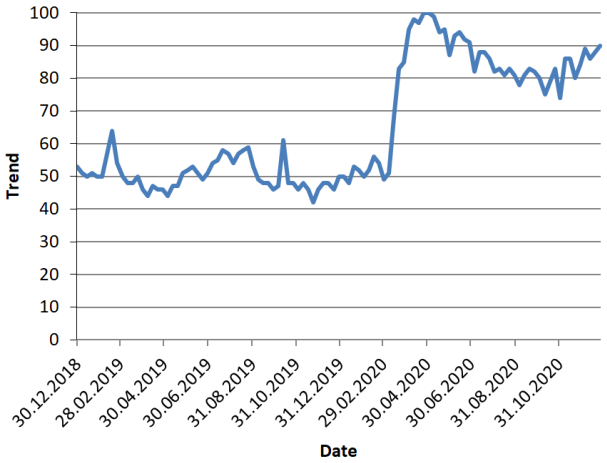


Figure 7: It shows the trend of searching for the term "Twitch" in the Google search engine.

Figure 7 is very similar to figures 5 and 6. Likewise, 2019 had a steady trend of around 50 with a sudden jump to 100 in March. Then you can see a decline during the holiday season, followed by an increase from September 2020.

The reason why the search trend figure is similar to the number of viewers and viewing hours is that Twitch is a website, so each time a user wants to watch something, they must enter this term into the browser. Of course, there are applications that allow you to watch Twitch without using a browser, but most users use the browser version.

5. Conclusions

The analysis of both computer games, virtual reality and streaming highlighted three events that had the greatest impact on digital entertainment. All these events are related to the pandemic namely the first was March 11, 2020 and this is the announcement of the Covid-19 virus as a pandemic by the World Health Organization, WHO. After this announcement, isolation began in many countries and numerous restrictions were imposed, leaving many people locked up in their homes with too much free time. As you can see in almost every chart, there was a huge jump for each of the analyzed types of entertainment during this time, during this time there was an upward trend.

After the March announcement, during the holiday season, ie from June to August, the results were in a horizontal or slightly downward trend.

This was due to the lifting of many restrictions or their reduction after the first wave of the pandemic in March, which meant that many people decided to go on vacation despite the pandemic still prevailing.

Another event that significantly influenced the analyzed data was the turn of September and October, i.e. the return to school and universities. This moment is important because it changes the trend for many charts from a horizontal or downtrend to an uptrend. The reason for such a change is that after the return of schoolchildren and students around the world, the number of infected people increased rapidly, which was not helped by the fact that during this period many people returned from holidays during which they had contact with many people from many parts of the world. Following the increase in infections, most countries have returned to remote teaching and the restoration of pre-vacation restrictions. Which positively influenced digital entertainment. The upward trend continued until the end of 2020.

As mentioned earlier, when analyzing in the case of virtual reality, you need to take into account the margin of error resulting from the fact that the data was collected only from users who participated in the survey and that Steam did not take into account the results of the Oculus Quest 2 goggles until 2021.

The analysis of search trends showed the same changes as described above, i.e. those with three significant events related to the pandemic. Likewise, a big jump in September with a horizontal or a downtrend in the holiday season and a renewed upward trend towards the end of 2020.

Summing up the period of the pandemic, it turned out to be very beneficial and important for the development of every type of analyzed digital entertainment, it allowed to achieve twice as much results compared to 2019 and even more. For example, in the figure of the number of users with VR goggles, the result is around 300% of the one from 2019 which is the highest percentage increase out of all analyzed data. The pandemic continues to extend, which will likely allow digital entertainment to reach even higher records in 2021.

References

[1] S. Mahendher, A. Sharma, P. Chhibber, A. Hans, Impact of COVID-19 on digital entertainment industry, UGC Care Journal 44 (2021) 148-161.

[2] S. Kohli, B. Timelin, V. Fabius, S. M. Veranen, How COVID-19 is changing consumer behavior–now and forever, McKinsey & Company (2020) 1-2.

[3] K. Salen, The Ecology of Games: Connecting Youth, Games, and Learning, MIT Press, Cambridge, 2008.

[4] W. Kriz, Gaming in the Time of COVID-19, SAGE Publications (2020) 1-3.

[5] Playing video games during quarantine, https://time.com/5824415/video-games-quarantine/, [01.08.2021].

[6] Main page of the steam platform, https://store.steampowered.com/, [01.08.2021].

[7] Steamcharts about page giving overall explanation of what it is, https://steamcharts.com/about, [01.08.2021].

[8] SteamDB page giving answers to most frequently asked questions, https://steamdb.info/faq/, [01.08.2021].

[9] What Is Twitch? How to Use the Live-Streaming Platform, https://www.makeuseof.com/what-is-twitch-live-streaming/, [01.08.2021].

[10] Post debating about potential of VR during pandemic, https://www.scmp.com/abacus/tech/article/3076992/will-pandemic-give-boost-virtual-reality, [01.08.2021].

[11] Main page of a nonprofit library of free movies, books and more, https://archive.org/, [01.08.2021].

[12] Main page of TwitchTracker site that collects data about Twitch, https://twitchtracker.com/, [01.08.2021].

[13] Page Google Trends allows the analysis of trends for terms searched in the google search engine, https://trends.google.pl/trends/, [01.08.2021].

[14] WHO has declared the COVID-19 pandemic. What does it mean?, https://pulsmedycyny.pl/who-oglosilo-pandemie-covid-19-co-to-oznacza-984790, [01.08.2021].

[15] Main page of Steam hardware survey containing explanation on what is it and most recent data collected, https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam, [01.08.2021].

[16] Post about influence of game called Half-life: Alyx on steam VR, https://www.roadtovr.com/steam-survey-vr-headset-growth-april-2020-half-life-alyx/, [01.08.2021].

[17] Post about growth of VR on steam in 2020, https://www.roadtovr.com/valve-steam-vr-2020-new-users-revenue/, [01.08.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 367–372
Received: 19 September 2021
Accepted: 14 Ocober 2021

Symfony and Laravel – a comparative analysis of PHP programming frameworks

Symfony i Laravel – analiza porównawcza szkieletów programistycznych języka PHP

Krzysztof Kuflewski*, Mariusz Dzieńkowski

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

This paper is a comparative analysis of PHP programming frameworks - Symfony and Laravel. The analysis was conducted on two test applications prepared for this purpose, based on the latest versions of the following technologies: Symfony 5.2 and Laravel 8. Both applications, being simple auction systems, have the same set of functionalities. They were compared in terms of selected criteria. Their implementation process, software metrics, performance and amount of community support were compared. Apache jMeter was used for performance testing. With its help, tests of several operations on databases were performed. The operations were as follows: adding auctions, retrieving auction details, editing, deleting auctions, bidding on an auction and simultaneous closing 1,000 auctions. The test results for the selected criteria were proved to be better for the Laravel framework based application.

Keywords: web applications; PHP frameworks; Symfony; Laravel

Streszczenie

Przedmiotem pracy jest analiza porównawcza szkieletów programistycznych języka PHP - Symfony i Laravela. Przeprowadzono ją na dwóch przygotowanych do tego celu aplikacjach testowych, opartych na najnowszych wersjach badanych technologii: Symfony 5.2 oraz Laravel 8. Obie aplikacje, będące prostym systemem aukcyjnym, posiadają ten sam zestaw funkcjonalności. Zostały one porównane pod względem wybranych kryteriów. Porównywano proces ich implementacji, metryki oprogramowania, wydajność oraz wielkość wsparcia społeczności. Do testowania wydajności wykorzystano program Apache jMeter. Z jego pomocą wykonano testy kilku operacji na bazie danych takich jak: dodawanie aukcji, pobieranie szczegółów aukcji, edycja, usunięcie aukcji, złożenie oferty na aukcję oraz jednoczesne zamknięcie 1000 aukcji. Wyniki badań dla wybranych kryteriów okazały się lepsze dla aplikacji opartej na szkielecie Laravel.

Słowa kluczowe: aplikacje internetowe; szkielety programistyczne języka PHP; Symfony; Laravel

*Corresponding author

Email address: **krzysztof.kuflewski@pollub.edu.pl** (K. Kuflewski)

1. Wstęp

1.1. Aplikacje internetowe i szkielety programistyczne

Ostatnie lata przyniosły zdecydowany wzrost zapotrzebowania na oprogramowanie komputerowe. Dużą popularnością cieszą się aplikacje internetowe - programy działające na serwerze internetowym, z których użytkownicy korzystają poprzez przeglądarkę internetową. Obecnie takie aplikacje stawiane są na równi z desktopowymi - powinny dostarczać jednakowych funkcjonalności, być wszechstronne, bezpieczne i szybko działające. Niewątpliwą zaletą aplikacji internetowych jest ich łatwa przyswajalność. Do użytkowania wystarczy połączenie z siecią i przeglądarka internetowa – nie ma więc potrzeby instalowania dodatkowego oprogramowania na urządzeniu. Postęp tego typu aplikacji przyczynił się do rozwoju technologii wykorzystywanych do ich tworzenia. Programiści mogą wybierać z wielu języków programowania pozwalających budować coraz bardziej zaawansowane aplikacje internetowe. W celu usprawnienia i przyspieszenia pracy nad tworzeniem programów dla wielu języków powstały szkielety programistyczne (ang. frameworks).

Szkielet programistyczny jest platformą do konstruowania aplikacji. Definiuje on jej strukturę oraz mechanizmy działania. Pomimo, że narzuca on sposób tworzenia aplikacji, w zamian programista dostaje gotowe biblioteki i rozwiązania, co znacznie przyspiesza pracę nad programem. Deweloper nie musi trudzić się rozwiązywaniem wielu powtarzalnych problemów i może w pełni skupić się na implementacji funkcjonalności. Rynek jest obecnie bardzo bogaty w szkielety programistyczne, a programiści stają przed dylematem, który z nich wybrać. Pomimo tego, że priorytetem wszystkich szkieletów jest usprawnienie implementacji, zapewnienie bezpieczeństwa aplikacji i wsparcie społeczności, każdy szkielet oferuje inne funkcje, rozwiązania i narzędzia. Dlatego też kluczową kwestią jest wybór odpowiedniego szkieletu do budowy planowanego projektu. Kierując się tą myślą, w niniejszej pracy przeprowadzono analizę porównawczą dwóch najbardziej popularnych szkieletów programistycznych języka PHP: Symfony i Laravel.

1.2. Przegląd literatury

Popularność aplikacji internetowych oraz duża liczba narzędzi wykorzystywanych do ich tworzenia doprowadziły do licznych rozważań nad wyborem najbardziej odpowiedniej technologii. Powstało wiele publikacji naukowych traktujących o tworzeniu aplikacji internetowych, technologiach stosowanych podczas ich programowania, czy narzędziach do testowania takich aplikacji. W wielu pracach, ich autorzy dążą do wskazania najlepszego – w danym aspekcie – rozwiązania.

Szkielety programistyczne Symfony i Laravel były już ze sobą porównywane. W publikacji *A comparative study of laravel and symfony PHP frameworks* autorzy stawiają tezę, że Symfony jest lepszym rozwiązaniem [1]. Podczas badań wzięto pod uwagę następujące kryteria: popularność, architekturę, budowę i cechy szkieletu, organizację kodu oraz wymagania systemowe. Twórcy na potrzeby analizy nie tworzyli aplikacji w badanych technologiach, a co się z tym wiąże, nie brali pod uwagę ich wydajności. Również nie są wskazane wersje badanych szkieletów, co może być istotną kwestią podczas implementacji. We wnioskach autorzy stwierdzają, iż wybór technologii zależy od potrzeb: Symfony jest lepszym rozwiązaniem do tworzenia dużych projektów, natomiast Laravel ma niższy próg wejścia, przez co może być częściej wybierany przez osoby zaczynające swoją przygodę z szkieletami programistycznymi języka PHP.

W publikacji *Practical Application of PHP Frameworks in Development of Web Information Systems* autorzy opisują kilka szkieletów programistycznych języka PHP: CakePHP2, CodeIgniter, Symfony 2, Yii i PhalconPHP, wskazują główne różnice w ich budowie i charakterystyce, a także wykonują testy wydajnościowe aplikacji wykonanych w dwóch szkieletach: Symfony2 oraz PhalconPHP [2]. Przedmiotem analizy w przypadku wydajności jest prosty system rezerwacji biletów, na którym wykonano po cztery scenariusze testowe dla każdej z technologii, a narzędziem do wykonania testów wydajnościowych był Apache Benchmark. Dokonano pomiarów czasów odpowiedzi na wysyłane żądania, zliczono żądania poprawnie obsłużone, zmierzono zużycie pamięci oraz liczbę plików, z których składała się dana aplikacja. Wyniki analizy wykazały, że PhalconPHP jest dużo wydajniejszy niż Symfony 2. Natomiast szkielet Symfony doceniono za szeroką ofertę możliwości, które znacznie ułatwiają i przyśpieszają proces implementacji.

Autorzy pracy *A new model for the selection of web development frameworks: application to PHP frameworks* zbadali cztery technologie: Symfony, Laravel, Zend, CodeIgniter bez konkretnie określonych wersji [3]. Celem ich analizy była pomoc w wyborze - przez potencjalnego programistę - odpowiedniego narzędzia do wykonania danej aplikacji. Jako kryteria porównawcze wybrane zostały: ogólne informacje o szkielecie, jego cechy charakterystyczne i budowa, czas wykonania projektu opartego na danym szkielecie oraz wydajność rozumiana jako liczba obsługiwanych żądań na sekundę. W każdej z porównywanych technologii wykonana została odrębna aplikacja wykorzystana do testów. W pracy nie ma informacji, jakie to były aplikacje. Do badań zostało użyte narzędzie ApacheBench – specjalistyczny program do wykonywania testów wydajnościowych serwerów HTTP. W podsumowaniu autorzy stwierdzają, iż każdy ze szkieletów ma swoje zalety, a wybór zależy od indywidualnych potrzeb programistów oraz specyfiki realizowanego projektu.

Technologie wytwarzania aplikacji internetowych, w tym szkielety programistyczne języka PHP, były i ciągle są ze sobą porównywane. Autorzy prac w swoich badaniach skupiali się na różnych aspektach tych technologii. Badano m.in.: główne założenia technologii, ich budowę i popularność. Na potrzeby niektórych analiz tworzono aplikacje testowe, które służyły jako obiekty badań głównie wydajności. W swoich pracach autorzy dążą do wskazania lepszego – pod danym względem – narzędzia. Niniejsza praca również wpisuje się w ten trend, koncentrując się na badaniu procesu implementacji, wydajności oraz metryk kodu programu, wykorzystując do tego celu utworzoną na potrzeby badań prostą aplikację bazodanową realizującą operacje CRUD, czyli zapis, odczyt, aktualizacja, usuwanie (ang. Create, Read, Update, Delete). Analizom poddano również wielkość wsparcia społeczności danej technologii. Spośród innych publikacji pracę tą wyróżniają wersje badanych szkieletów programistycznych, zaimplementowana aplikacja testowa, opracowane scenariusze testowe, a także narzędzie użyte do testowania.

1.3. Cel badań

Celem badań jest porównanie dwóch szkieletów programistycznych języka PHP - Symfony oraz Laravel. Na potrzeby rozważań stworzono dwie jednakowe aplikacje oparte na najnowszych wersjach szkieletów - Symfony 5.2 oraz Laravel 8. Przeprowadzona na podstawie opracowanych aplikacji analiza pod kątem wybranych kryteriów pozwoli wyciągnąć odpowiednie wnioski i pomóc w wyborze właściwego narzędzia do stworzenia aplikacji internetowej.

2. Metodologia badań

2.1. Aplikacje testowe

Do wykonania badania niezbędne było przygotowanie aplikacji testowej w dwóch egzemplarzach - jednej zbudowanej na bazie szkieletu Laravel 8 oraz drugiej w Symfony 5.2. Obie aplikacje wykonano w jednakowy sposób - zawierają one taki sam zestaw funkcjonalności, korzystają z tej samej bazy danych, posiadają bardzo podobne widoki. Aplikacją jest prosty system aukcyjny, umożliwiający realizację podstawowych operacji CRUD na bazie danych oraz uwierzytelnianie użytkowników. System rozróżnia dwa rodzaje użytkowników: gościa oraz użytkownika zalogowanego.

Wymagania funkcjonalne aplikacji testowych:

- Użytkownik niezalogowany:
  - rejestracja w systemie,
  - logowanie do systemu,
  - przeglądanie listy aukcji,

- przeglądanie szczegółów aukcji.
- Użytkownik zalogowany:
  - tworzenie, edycja i usuwanie własnych aukcji,
  - składanie ofert do aukcji,
  - wylogowanie.

Struktura bazy danych
Baza danych składa się z trzech tabel:
- *users* - opisującej użytkowników zarejestrowanych w systemie,
- *auctions* - opisującej aukcje stworzone przez użytkowników,
- *offers* - opisującej złożone przez użytkowników oferty w aukcjach.

Tabele zostały powiązane odpowiednimi relacjami:
- *users i auctions* – **1:n** - jeden użytkownik może utworzyć wiele aukcji,
- *users i offers* – **1:n** - jeden użytkownik może złożyć wiele ofert,
- *auctions i offers* – **1:n** – jedna aukcja może posiadać wiele ofert.

2.2. Środowisko testowe

W celu zapewnienia jak najbardziej wiarygodnych wyników badań wydajnościowych dla obydwu technologii, aplikacje zostały uruchomione w jednym środowisku testowym, z wykorzystaniem pakietu XAMPP. Korzystają one również z jednakowej bazy danych MySQL. Dane środowiska testowego przedstawia Tabela 1.

Tabela 1: Konfiguracja środowiska testowego aplikacji

Procesor	Intel Core i5-5200U
System operacyjny	Windows 10 Home 64-bit
Pamięć RAM	8 GB
Przeglądarka internetowa	Google Chrome 89.0.4389.128
Pakiet XAMPP	7.4.10
Serwer Apache	2.4.46
MySQL	7.4.10
PHP	8
Narzędzie testowe	Apache JMeter 5.4.1

Narzędziem testowym do badania wydajności aplikacji był Apache JMeter. Jest to dedykowane oprogramowanie do wykonywania testów wydajnościowych, obciążeniowych i funkcjonalnych na aplikacjach internetowych. Jego przeznaczeniem jest badanie wydajności statycznych oraz dynamicznych zasobów: plików, dynamicznych języków programowania serwisów internetowych, np.: Java, PHP, ASP.NET, obiektów Java, kwerend, baz danych, serwerów FTP [4].

2.3. Kryteria porównawcze i scenariusze testowe

W celu wykonania badań oraz dokonania analizy porównawczej omawianych w pracy szkieletów programistycznych, niezbędne było wybranie kryteriów, pod względem których pośrednio szkielety, a bezpośrednio aplikacje zostały sprawdzone. Do badania wydajności aplikacji opracowano scenariusze testowe, które pozwoliły zweryfikować wydajność aplikacji w trakcie wykonywania określonych zadań.

Proces implementacji aplikacji
Sposób tworzenia aplikacji w danej technologii ma duże znaczenie przy jej wyborze do realizacji określonego projektu. Porównany został mechanizm tworzenia aplikacji bazodanowej przy użyciu analizowanych szkieletach programistycznych na przykładzie systemu aukcyjnego. Wynikiem tych porównań jest zestawienie podobieństw oraz różnic. Nie skupiono się na przebiegu wszystkich etapów budowy aplikacji testowych, ale na tych kluczowych - będących charakterystycznymi elementami w obrębie badanego szkieletu.

Metryki aplikacji - rozmiar projektu, objętość kodu źródłowego
Rozmiar projektu ma wpływ na jego wydajność, a objętość kodu źródłowego jest czynnikiem wpływającym na pracochłonność implementacji aplikacji. Mniejsza ilość kodu to także większa czytelność i krótszy czas niezbędny do wytworzenia oprogramowania, co w konsekwencji sprowadza się do niższego kosztu.

Wydajność
Wydajność jest bardzo istotnym elementem aplikacji internetowych. Twórcom oprogramowania często zależy na sprawnym działaniu aplikacji i szybkim wykonywaniu zadań. W celu zbadania wydajności przygotowano scenariusze testowe, które pozwolą sprawdzić, jak w tym zakresie radzą sobie szkielety Laravel 8 i Symfony 5.2.

Opracowane scenariusze testowe dotyczyły prostych działań realizowanych przez użytkowników na aplikacjach testowych, podczas których rejestrowany był czas ich wykonania. Scenariusze obejmowały następujące czynności:
- dodanie aukcji,
- pobranie szczegółów aukcji,
- edycja aukcji,
- usunięcie aukcji,
- złożenie oferty do aukcji,
- zamknięcie 1000 aukcji.

Ponadto przygotowano trzy scenariusze, które wykorzystano do analizy czasów wyświetlania stron zawierających dane:
- wyświetlenie listy 100 aukcji,
- wyświetlenie szczegółów pojedynczej aukcji,
- wyświetlenie formularza dodawania aukcji.

Wsparcie społeczności
Specjalistyczne forum skupiające wokół siebie dużą liczbę osób korzystających z danej technologii, to grupa która może pomóc w sytuacjach, gdy dokumentacja z jakichś powodów okaże się niewystarczająca. Osoby te, dzięki swojemu zaangażowaniu, mogą odpowiadać na nurtujące pytania, rozwiązywać trudne problemy oraz wyjaśniać różne wątpliwości. Badanie wsparcia społeczności polegało na sprawdzeniu liczby zapytań dotyczących porównywanych technologii. Wykorzystano do tego celu jedno z najbardziej popularnych forów

programistycznych – StackOverflow.com [5]. Dodatkowo pod uwagę wzięto liczbę pytań posiadających co najmniej jedną odpowiedź, co zwiększa prawdopodobieństwo znalezienia rozwiązania dla danego problemu.

2.4. Proces implementacji

Implementacja aplikacji opartych na szkieletach Symfony 5.2 i Laravel 8 odbywa się w podobny sposób - językiem bazowym obydwu z nich jest PHP, obydwa oparte są o wzorzec projektowy MVC (ang. Model-View-Controller), struktura projektu jest także analogiczna. Zarówno Symfony jak i Laravel posiadają aplikację konsolową umożliwiającą tworzenie m.in. kontrolerów, modeli i widoków. Obydwa szkielety korzystają z systemu migracji i posiadają swoje pakiety do uwierzytelniania użytkownika, które wystarczy zainstalować i dopasować do swoich potrzeb. Szablony Symfony (twig) są bardzo podobne do szablonów Laravel'a (blade). Definicja rutingów w pliku kontrolera bezpośrednio przy funkcji obsługującej daną akcję w Symfony wydaje się bardziej czytelna niż rutingi w oddzielnym pliku w przypadku Laravela. Laravel poprzez brak repozytoriów, bardziej zwięzły kod przy działaniach z bazą danych, oraz brak definicji formularzy jako obiektów PHP jest z pewnością lepszą opcją dla początkującego programisty rozpoczynającego pracę ze szkieletami języka PHP niż Symfony.

2.5. Metryki aplikacji

Rozmiar kodu projektów aplikacji testowych
Badanie rozmiaru projektów aplikacji testowych wykazało, że projekt Laravela jest o około 16% mniejszy niż Symfony (odpowiednio: 38Mb, 45Mb). Pod uwagę nie brano katalogu *node_modules* zawierającego pliki bibliotek strony klienta i potrzebnego przy użyciu technologii Tailwind.css, odpowiedzialnej za warstwę wizualną aplikacji testowych.

Objętość kodu źródłowego
Objętość kodu źródłowego aplikacji testowych zbadano przy pomocy metryki SLOC (ang. Source Lines of Code), używanej do pomiaru rozmiaru oprogramowania komputerowego. Polega ona na zliczeniu linii kodu źródłowego, co może się przełożyć na ilość pracy potrzebnej do wykonania programu, a także jego dalszego utrzymania [6]. Badanie zostało wykonane z pomocą biblioteki *npm* o nazwie *sloc*. Testom poddano tylko te obszary aplikacji, które zostały zaimplementowane przez programistę. Nie badano plików bibliotek i innych plików generowanych automatycznie.

Tabela 2: Liczba linii kodu w poszczególnych częściach aplikacji testowych w badanych szkieletach

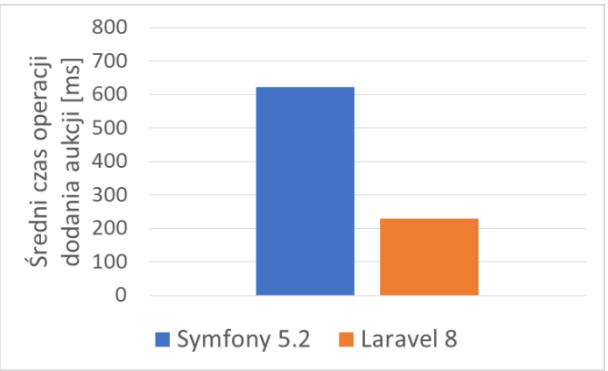
	Symfony 5.2	Laravel 8
Modele	362	75
Kontrolery	160	225
Widoki	431	456
Formularze	68	-
Rutingi	-	7
Migracje	-	91
Suma	**1021**	**854**

Wyniki zostały przedstawione w Tabeli 2. Zawiera ona liczbę linii kodu źródłowego w plikach odpowiedzialnych za poszczególne części stworzonej aplikacji. Brak plików z danego obszaru lub brak konieczności implementowania ich w danym szkielecie oznaczono znakiem "-".

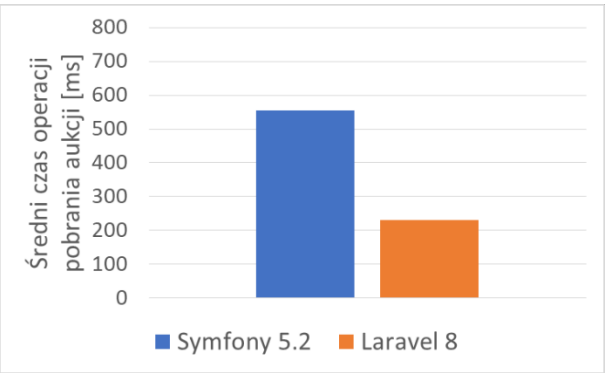
Badanie metryki SLOC wykazało, że aplikacja wykonana za pomocą szkieletu Laravel ma mniejszą ilość kodu o około 16%. Przy dużych projektach taka różnica może okazać się kluczową kwestią wpływającą na czas wykonania oprogramowania, a także jego późniejsze utrzymanie.

2.6. Wydajność

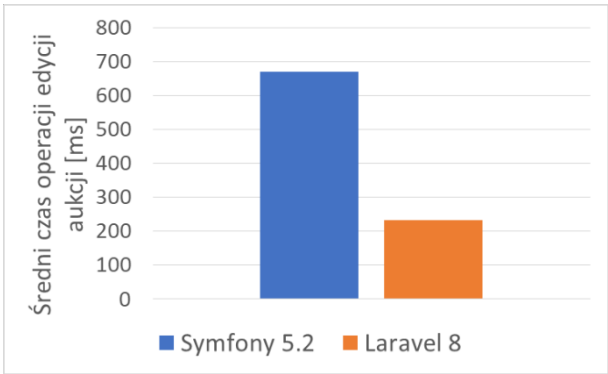
Każdy ze scenariuszy testowych został powtórzony 10 razy dla każdej aplikacji, na jednakowych zestawach danych. Na Rysunkach 1 - 6 przedstawiono wyniki w postaci średnich czasów wykonania operacji wyrażonych w milisekundach. Natomiast Rysunki 7 - 9 prezentują średnie czasy wyświetlania widoków zawierających dane (w sekundach).



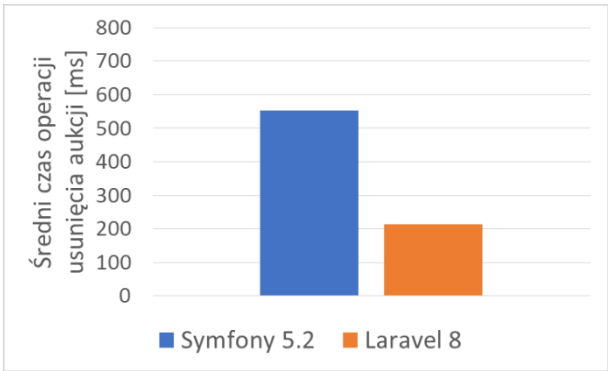
Rysunek 1: Wyniki dla scenariusza 1 – dodanie aukcji.



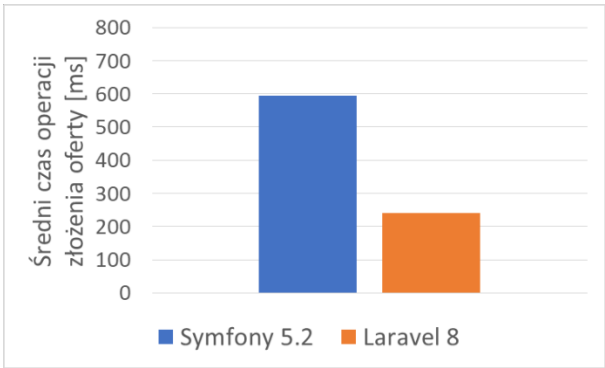
Rysunek 2: Wyniki dla scenariusza 2 – pobranie szczegółów aukcji.



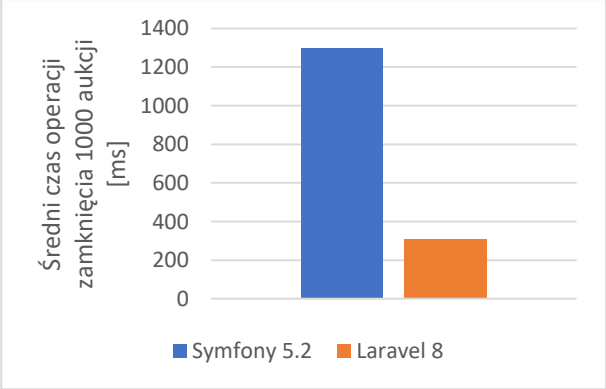
Rysunek 3: Wyniki dla scenariusza 3 – edycja aukcji.



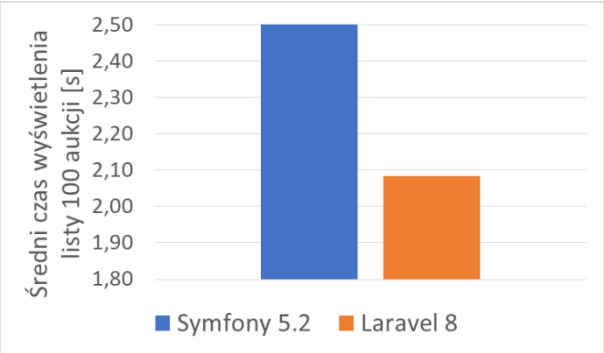
Rysunek 4: Wyniki dla scenariusza 4 – usunięcie aukcji.



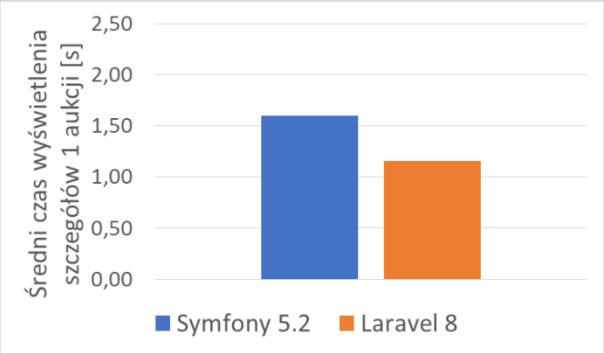
Rysunek 5: Wyniki dla scenariusza 5 – złożenie oferty na aukcję.



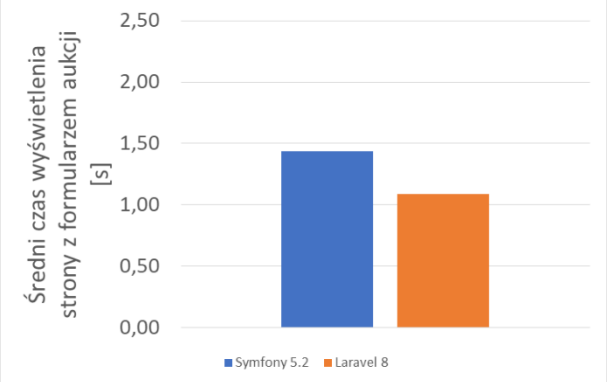
Rysunek 6: Wyniki dla scenariusza 6 – zamknięcie 1000 aukcji.



Rysunek 7: Wyniki dla scenariusza 7 – wyświetlenie strony z listą 100 aukcji.



Rysunek 8: Wyniki dla scenariusza 8 – wyświetlenie strony z pobranymi szczegółami jednej aukcji.

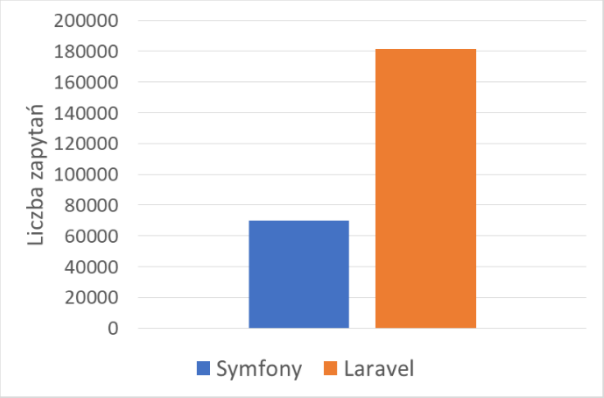


Rysunek 9: Wyniki dla scenariusza 9 – wyświetlenie strony z formularzem edycji aukcji.

W scenariuszach 1-5 dotyczących czasów wykonania operacji (dodanie aukcji, pobranie szczegółów aukcji, edycja aukcji, usunięcie aukcji, złożenie oferty na aukcję) Laravel był średnio 2,6 razy szybszy od Symfony. W przypadku scenariusza 6 (zamknięcie 1000 aukcji) różnica czasów była jeszcze większa – Laravel był ponad cztery razy szybszy od Symfony. Było to związane z brakiem w ORM (ang. Object-Relational Mapping) Doctrine, z którego korzysta Symfony, możliwości jednoczesnego zaktualizowania wielu rekordów (zamknięcie aukcji jest zmianą jej statusu w bazie danych). Z kolei taką możliwość daje ORM Eloquent, z którego korzysta Laravel. W przypadku wyświetlania stron z danymi (scenariusze 7 - 9) różnica pomiędzy badanymi szkieletami jest mniejsza, ale wciąż na korzyść Laravela. Wyświetlanie stron z danymi trwało w tym przypadku średnio o 30% krócej.

2.7. Wsparcie społeczności

W celu określenia wsparcia społeczności zbadano liczbę zapytań dotyczących obu narzędzi w serwisie StackOverflow.com. Wyniki przedstawiono na rysunku 10.



Rysunek 10: Liczba zapytań dla badanych szkieletów w serwisie StackOverflow.com.

Liczba wyników zapytań dla szkieletu Laravel (181341) była zdecydowanie wyższa niż w przypadku Symfony (69727). Było to około 2,6 razy więcej na korzyść Laravela w stosunku do Symfony. Odsetek pytań zawierających co najmniej jedną odpowiedź był na podobnym poziomie dla obydwu technologii: dla Laravela wynosił 84% (152326), a dla Symfony 85% (59268). Z dużo większej liczby pytań z co najmniej jedną odpowiedzią dla szkieletu Laravel można wysnuć wniosek, że platforma ta budzi większe zainteresowanie u programistów i stwarza większe prawdopodobieństwo znalezienia pomocy wśród bardzo licznej społeczności.

3. Wnioski

Rynek aplikacji internetowych rozwija się bardzo prężnie – zapotrzebowanie na aplikacje rośnie, a samym programom stawia się coraz wyższe wymagania. Zwiększa się także liczba technologii do tworzenia aplikacji www, a te istniejące, cały czas są rozwijane. Z tego względu programista staje przed trudnym wyborem – jaką technologię wybrać do swojego projektu.

Celem niniejszej pracy było porównanie dwóch najbardziej popularnych szkieletów programistycznych języka PHP: Symfony i Laravela. Do celów badawczych przygotowano dwie aplikacje testowe w najnowszych (w momencie rozpoczęcia badań) wersjach tych szkieletów. Aplikacje zostały zbudowane w podobny sposób, zgodnie z założeniami danego narzędzia. Posiadały one jednakowy zestaw funkcjonalności, a także korzystały z tych samych danych i zostały uruchomione w tym samym środowisku testowym. Aplikacje testowe poddano badaniom w obrębie wyselekcjonowanych kryteriów porównawczych i scenariuszy testowych.

Wyniki przeprowadzonych badań wypadają na korzyść szkieletu Laravel 8. Implementacja prostej aplikacji typu CRUD była łatwiejsza, a projekt zajmował mniej miejsca na dysku i składał się z mniejszej liczby linii kodu źródłowego. Analiza wydajności takiej aplikacji zrealizowana na podstawie czasów wykonania operacji CRUD oraz wyświetlania stron z danymi wykazała, że Laravel 8 okazał się szkieletem wydajniejszym niż Symfony 5.2. Kryterium wsparcia społeczności, wskazuje na szerokie zainteresowanie Laravelem. Należy przy tym dodać, że interpretacja tego kryterium może być także inna. Duża społeczność i jej aktywność może oznaczać, że platforma ta sprawia więcej problemów oraz zawiera wiele mankamentów i wymaga wielu zabiegów, aby je naprawiać.

Z przeprowadzonych w ramach pracy badań na prostych aplikacjach testowych według dobranych przez autorów kryteriów wynika, że Laravel jest lepszym rozwiązaniem do tworzenia prostych aplikacji współpracujących z bazą danych niż Symfony. Jednak, aby jednoznacznie stwierdzić, który szkielet jest lepszy należałoby wykonać testy na średnich i dużych systemach. Poza tym w badaniach należałoby uwzględnić więcej aspektów tych narzędzi, ponieważ są one bardzo rozbudowane.

Literatura

[1] M. Laaziri, K. Benmoussa, S. Khoulji, K. M. Larbi, A. El Yamami, A comparative study of laravel and symfony PHP frameworks, International Journal of Electrical and Computer Engineering (IJECE), 9(1) (2019), http://ijece.iaescore.com/index.php/IJECE/article/view/11601/11103, [01.09.2021].

[2] U. K. Latif, T. Kusumasari, Comparison Between Yii Frameworks and Laravel In 3 Different Version For Viewing Large Data of Shipyard Industry in Indonesia, International Journal of Innovation in Enterprise System, 2(1) (2018) 02-13, https://ijies.sie.telkomuniversity.ac.id/index.php/IJIES/article/view/12/146, [01.09.2021].

[3] K. Benmoussa, M. Laaziri, S. Khoulji, K. M. Larbi, A. El Yamami, A new model for the selection of web development frameworks: application to PHP frameworks, International Journal of Electrical and Computer Engineering (IJECE), 9(1) (2019), http://ijece.iaescore.com/index.php/IJECE/article/view/11586/11102, [10.09.2021].

[4] Apache JMeter – opis, https://www.slideshare.net/sjsi/jmeter-narzdzie-testera, [20.06.2021].

[5] Stack Overflow – forum programistyczne, https://stackoverflow.com, [09.09.2021].

[6] Metryka SLOC – opis, https://pvs-studio.com/en/blog/terms/0086/, [20.06.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 373–377
Received: 21September 2021
Accepted: 10 Ocober 2021

A comparative analysis of cryptocurrency wallet management tools

Analiza porównawcza narzędzi do zarządzania portfelem kryptowalut

Kamil Biernacki*, Małgorzata Plechawska-Wójcik

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The work is devoted to a comparative analysis of tools for managing a cryptocurrency portfolio. The study aimed to find out which of the tools is currently the best solution for users. This analysis was carried out on the basis of two tools, first: a survey conducted of among 41 responders who are cryptocurrency users, and the second one: cognitive walkthrough. Conducted analysis confirmed the thesis that the Trust Wallet tool is currently the best solution for users.

Keywords: cryptocurrency; blockchain; user experience; usability

Streszczenie

Praca poświęcona jest analizie porównawczej narzędzi do zarządzania portfelem kryptowalut. Badanie miało na celu stwierdzić, który z portfeli stanowi obecnie najlepsze rozwiązanie dla użytkowników. Analiza ta została przeprowadzona na podstawie dwóch narzędzi, pierwszego: ankiety przeprowadzonej wśród 41 respondentów, którzy są użytkownikami kryptowalut, oraz drugiego: wędrówki poznawczej. Wykonana analiza potwierdziła tezę, że portfel Trust Wallet stanowi obecnie najlepsze rozwiązanie dla użytkowników.

Słowa kluczowe: kryptowaluty; łańcuch bloków; doświadczenie użytkownika; użyteczność

*Corresponding author

Email address: kamil_biernacki@pollub.edu.pl (K. P. Biernacki)

1. Introduction

Nowadays, cryptocurrencies have become an area of interest for many people in their daily lives. Cryptocurrencies allowed many people to get rich for others, they brought losses. Cryptocurrency basically speaking, it is a virtual money that does not have its physical form [1]. The first cryptocurrencies appeared in 2008, but their popularity was related to the creation of bitcoin, which was found in the last quarter of this year [2].

Most cryptocurrencies use a decentralized system based on blockchain technology [3]. Blockchain technology is a chain of blocks that are linked chronologically using cryptography. Blockchain is distributed and saved on a peer-to-peer network, without a centralized server, formed by computers connected to each other on the network using a consensus algorithm. Blockchain has many desirable values, including consistency and irreversibility. Each cryptocurrency must be mined, and there are various method dedicated to this process [4]. Mining is used to check the correctness of the transaction and for this effort miners receive a new cryptocurrency. Bitcoin is the most popular cryptocurrency, but there are other cryptocurrencies that serve different purposes.

Cryptocurrencies, for security purposes, use an asymmetric cryptography, including two: private and public keys. The public key can be freely accessed, it is used to send funds to the cryptocurrency wallet. It is used to encrypt information, while the private key is used to read and must not be shared. Sharing the private key may allow people to give up control over the cryptocurrency [5]. These keys are managed by tools called a cryptocurrecy wallet, which, through the interface, allow users to access cryptocurrencies. Therefore it is worth comparing them. According to article [6], payments made in crytocurrencies ensure minimal costs, because there are two nodes that carry out transactions directly with each other, there is no need to include third parties.

There are many research papers exploring the possibilities and limitations of cryptocurrencies and their functioning in the economy. The article [7] describes the use of the bitcoin cryptocurrency in the modern world and attempts to assess whether bitcoin can replace traditional currencies. Research has shown that bitcoin has the potential to become an essential element in the economy, but it lacks the features that will allow it to replace the world's major currencies.

In the article [8] a usability study was conducted on the basis of the analytical Cognitive Walkthrough method for three popular cryptocurrencies and their wallets. In the study, participants were asked to perform tasks that tasks that allow investigate common usability issues with desktop and mobile-based wallets. Research has shown that currency wallets used to use cryptocurrencies suffer from serious usability issues, in particular for ordinary users, which is also confirmed to some extent by the article [9].

Paper [10] defines how cryptocurrencies build a fully decentralized financial system, independent of governments. The study concluded that the development of a new virtual monetary system has its relevance as a service or payment for online purchases that are currently gaining strength. Therefore, people are inclined to buy and sell online with bitcoin, so this new financial system would be very well received.

Bitcoin, being the most popular currency, is often analyzed. The publication [11] presents a survey of 990 bitcoin users, where it was examined how users experienced bitcoin in terms of security, privacy, and anonymity. The study found that users still had problems with bitcoin management, including not using all the security features of bitcoin cryptocurrency.

A study [12] examined the security of commonly used cryptocurrency applications for Android, which found that conventional financial services applications are slightly better than cryptocurrency applications in terms of security, but they offer greater privacy.

2. Purpose of work

The aim of the work is to conduct a comparative analysis of selected cryptocurrency management tools. The study aims to find out which tool is currently the best solution for users. This analysis was carried out on the basis of two tools, first: a survey among cryptocurrency users, and the second one: cognitive walkthrough. The thesis was based on the current observation of the market situation.

Thesis: Trust Wallet is currently the best cryptocurrency management tool.

Detailed research questions:

1. Does the Trust Wallet cryptocurrency wallet tool have the best functionalities in the opinion of users?
2. Is the user interface of Trust Wallet useful?
3. Is Trust Wallet the most popular tool among users?

3. Materials and methods

Tools for managing cryptocurrencies, in other words cryptocurrency wallets, were created with the emergence of the first Bitcoin cryptocurrency. The first cryptocurrency management tool was Bitcoin Core for desktops, which was dedicated to manage only Bitcoin. Since then, many types of wallets have emerged. There are different types of cryptocurrency wallets that can be divided into groups.

Contrary to popular belief, cryptocurrency management tools do not actually store cryptocurrencies, but provide tools for interacting with the blockchain network. They generate the information necessary to send and receive cryptocurrencies via the blockchain. Cryptocurrency management tools contain an address, which is an alphanumeric identifier generated from public and private keys. An address is a specific location on the blockchain to which transactions can be performed.

Desktop Wallet: Desktop wallets are available on most operating systems such as Windows, Linux, Mac. In stationary wallets, private keys are stored in the computer on the hard drive, in order to use them, appropriate software must be installed. An example of a stationary wallet can be the Exodus wallet [13], which also has a mobile version.

Mobile Wallet: Mobile wallets are applications that can be installed on smartphones. They work well for users who use cryptocurrencies as a means of payment as they give quick access wherever you are. An example of a mobile wallet is Trust Wallet [14].

Paper Wallet: A paper wallet is a private key printed on paper, usually with a QR code. Private keys are stored offline, making cryptocurrencies safer from cyberattacks, malware, etc. Figure 1 shows an example of a paper wallet.

Figure 1: An example of an Etherum paper wallet.

Hardware Wallet: Hardware wallets are physical electronic devices designed to safely store cryptocurrencies. They use a random number generator to generate public and private keys. The keys are stored on a device that is not connected to the Internet. These wallets are considered to be one of the safest options for storing cryptocurrencies. In order to carry out transactions with the use of this wallet, it must be connected to a computer or a smartphone. An example of a hardware wallet is Ledger [15].

Web Wallet: Web wallets are tools that can be accessed through various web browsers such as Google Chrome, Firefox, Brave, Edge. There are two types of online wallets that can be accessed through a website and those that are installed through a browser plug-in, where the keys are stored on the computer. An example of a hardware wallet is MetaMask [16].

3.1. Surveys

The survey was aimed at collecting and checking basic information about the users preferences regarding the cryptocurrency management tool they use. In order to confirm the thesis and answer research questions, two-stage research was carried out. A survey and a cognitive walkthrough that both confirmed the thesis and answered research questions.

The study collected basic information about the respondents, such as age, gender, income and continent of residence. The data was collected in order to verify the profile of the study participants.

For the wallet which users chose as the one they use most often, they were asked questions: did they know what to do when they started using the tool, how they assess security and whether they feel safe when using the tool, assessment of the interface, understandability of the elements included in the tool, ease of using the wallet, is the tool offering a sufficiently large number of services/functions, are they satisfied with the tool, are they comfortable using the tool. The participants had to answer these questions on a scale of 1 to 10.

The questionnaire has been shared electronically. The surveys were sent via private messages to cryptocurrency users and users of cryptocurrency related

groups on social networks. The questionnaires were designed in Polish and translated into English in order to reach more respondents.

3.2. Cognitive walkthrough

There was a total of 5 people taking part in the cognitive walkthrough. All participants had the title of engineer and were students of the Lublin University of Technology at the Faculty of Electrical Engineering and Computer Science. For the purposes of the study, 4 main tasks were created, which were divided into 11 subtasks. With each task, participants had to answer standard yes or no questions and the number of errors made by the participant was counted.

In order to carry out the cognitive walkthrough study, the following tasks were selected.

Task 1. Set up the wallet.
1. Create a new account,
2. Find and save 12-24 word recovery phrase.

Task 2. Find address of the Litecoin cryptocurrency.
1. Find or add a cryptocurrency token, track it if needed,
2. Find a functionality option that allows you to receive a given cryptocurrency,
3. Find the address.

Task 3. Execute a transaction to the address provided.
1. Find a functionality option that allows you to send a given cryptocurrency,
2. Enter the recipient's address,
3. Enter the quantity,
4. Find the confirm button.

Task 4. Recover the wallet after loss (In-app option or must be installed).
1. Find the appropriate restore option,
2. Insert 12-24 word recovery phrase.

Five questions were asked for each subtask. For each subtask, you had to answer "yes" or "no". The total number of responses or steps for each tool was 55.

P1. Will the user understand how to start the task? (understanding that the action is needed)
P2. Is the action clearly visible?
P3. Will the user associate the correct action with the outcome they expect to achieve?
P4. If the task has been successfully completed, will the user see progress towards the intended outcome?
P5. Did the user successfully complete the task?

The tools selected for the study are those that are one of the most commonly used for storing multiple cryptocurrencies. The research was conducted for cryptocurrency management tools such as Coinbase Wallet, Trust Wallet, Exodus, Jaxx Liberty and Coinomi, all of these tools have mobile versions. The research was conducted on mobile versions of all tools.

Due to the global situation caused by COVID disease, the research was conducted electronically. Each participant was given 4 tasks to be performed. All participants performed tasks using the Android NoxPlayer emulator through the screen shared to them. Their activities were monitored and any possible comments were noted. The monitoring consisted in asking participants to express their impressions aloud on an ongoing basis while performing the task. The person conducting the experiment did not interfere in the tasks performed by the participants of the study.

4. Results

4.1. Survey

In the survey, after removing incorrectly completed questionnaires, 41 responses were finally obtained, of which the largest group of respondents were people aged 20-30. The exact age of the users was not collected, only the age ranges in which the respondents are located. Figure 2 shows which types of wallets are used by the respondents. Most of the respondents (75.61%) use a cryptocurrency wallet on a mobile device. Tools for stationary devices are used by 56.10% of the respondents. None of the respondents use the paper wallet solution. A small proportion of respondents (12.20%) use special hardware devices to access cryptocurrencies. Less than half of the respondents (43.90%) declared that they use the browser portfolio.

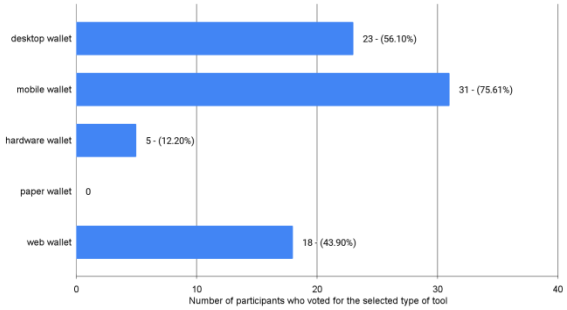


Figure 2: Bar chart showing the number of participants and the percentage of all participants who use a given type of wallet.

Figure 3 presents answers to the question which tools the respondents use most often. The survey participants most often chose the Trust Wallet tool, which was indicated by 29.30% of the respondents. In second place are three tools: MetaMask, Exodus, and Coinbase Wallet, each of them by 14.60% of respondents.

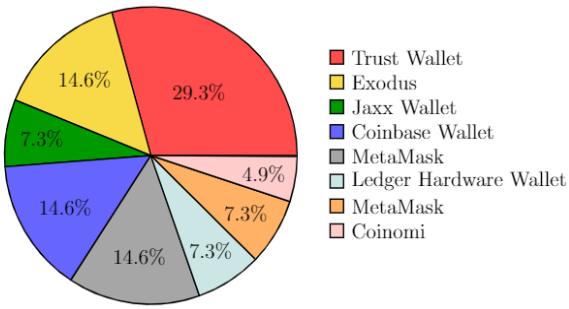


Figure 3: Pie chart showing the tool most respondents use.

For the tools selected in the question presented in Figure 3, users were asked about their impressions about the portfolio, its interface, security, and transpar-

ency. On the basis of the obtained results, the users, when assessing the level of safety and sense of security, gave most of the tools an average above 8 on a scale from 1 to 10.

It can be concluded that users feel safe using these solutions. In terms of safety, the worst tools was Exodus and MetaMask, with an average score of around 6.5 points (see Table 1). The results show that the respondents considered the Jaxx Liberty wallet as not having enough functionality, as it achieved an average of 5 points. Most of the tools in terms of assessing functionality or services received an average score of around 7 points. In terms of satisfaction, the Trust Wallet achieved the highest average. When assessing the comfort of using the application, the Trust Wallet and Coinomi tools were the best.

Of all the tools, Trust Wallet and Atomic Wallet performed best, having one of the highest averages of all the questions.

Table 1: User ratings for wallet questions on 10 point scale

Question / Tool	Trust Wallet	Coinomi	Atomic	Jaxx Liberty	MetaMask	Coinbase Wallet	Exodus
	Mean						
Did you know what to do when you started using your cryptocurrency wallet?	7.75	9.00	7.00	4.33	4.67	6.00	5.50
How do you rate the security level of the cryptocurrency management tool?	8.25	10.00	8.67	8.33	6.67	8.17	6.50
Do you feel safe when using a cryptocurrency wallet?	8.58	9.50	9.00	7.33	6.67	8.17	6.00
How do you rate the interface of the cryptocurrency management tool?	8.17	9.00	9.00	6.67	5.50	8.00	7.50
How do you rate the understandability of the elements contained in the cryptocurrency wallet? (do the elements have a specific purpose)	7.83	8.50	8.67	6.33	5.83	7.50	6.67
How do you rate the ease of use of the wallet?	8.25	9.50	8.33	6.67	6.83	6.67	8.17
Do you think the wallet offers enough services / features?	7.67	9.50	7.67	5.00	7.17	7.33	6.67
How satisfied are you with your wallet?	8.33	7.50	8.00	6.67	7.17	8.00	7.00
Do you feel comfortable using the application?	8.75	10.00	7.67	7.67	7.33	7.00	7.00

4.2. Cognitive walkthrough

The total number of failed steps for each tool is shown in Table 2. The table also shows the average number of failed steps committed by all participants out of 55 steps and their percentage. In each tool, at least one user has taken one failed step.

The users of the Coinomi tool faced the most problems. Most of the users reported having trouble with finding the option to restore the wallet. There were also problems finding the send and receive functionality, participants reported that they were not visible enough. The problematic functionality of sending and receiving is shown in Figure 4.

Mobile versions of selected tools were tested in the study. Each of the tools achieved a result of over 90% of correctly performed tasks.

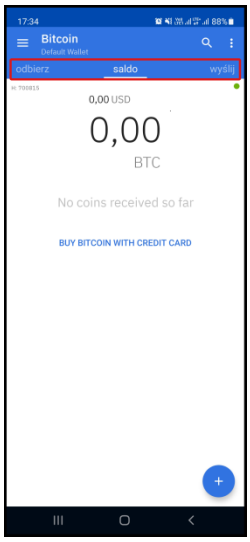


Figure 4: Faintly visible receive and send buttons in the Coinomi.

Table 2: Failed steps of the participants for each tool

Participant	P1	P2	P3	P4	P5	Avg. number of incorrect steps	Percentage of incorrect steps
Coinbase Wallet	0	0	1	0	0	0.2	0.36%
Trust Wallet	0	1	2	2	1	1.2	2.18%
Atomic Wallet	0	0	1	0	0	0.2	0.36%
Exodus	0	0	1	0	0	0.2	0.36%
Jaxx Liberty	2	0	0	2	2	1.2	2.18%
Coinomi	1	3	1	2	2	1.8	3.27%

All study participants were asked to choose two tools that they rated best after the study. Table 3 shows the tools chosen by each participant. The most votes were obtained by the Exodus, which obtained 5 votes, followed by the Trust Wallet, which received 4 votes from users, and the Coinbase Wallet tool with one vote.

Table 3: Two tools selected by each participant

Participant	Selected wallets	
1	Trust Wallet	Exodus
2	Coinbase Wallet	Exodus
3	Trust Wallet	Exodus
4	Exodus	Trust Wallet
5	Trust Wallet	Exodus

5. Conclusions

The main focus of the present research was put on determining which tool is currently the best solution for holding cryptocurrency. This analysis was carried out on the basis of surveys that were carried out among users of cryptocurrencies and the cognitive walkthrough.

The conducted research allowed to confirm the thesis and answer the research hypotheses.

One of the hypotheses was whether Trust Wallet is the most popular tool among users. After analyzing the survey, it can be concluded that it is one of the most popular tools because the largest number of respondents chose this tool as the one they use most often.

Another hypothesis was whether the Trust Wallet has the best functionalities according to the opinion of the users. When analyzing the survey, it can be noticed that Trust Wallet users gave a high rating in terms of the functionality offered by Trust Wallet, only Coinomi got a higher rating, but this may be due to the small number of respondents who chose the Coinomi tool.

The third research hypothesis was whether the user interface of the Trust Wallet tool is rated as useful. In the survey, the Trust Wallet tool performed very well, most of the users ratings were above the average of 7.5 points. The Cognitive Walkthrough study showed minor imperfections in the interface, but the overall result of correctly performed steps was 97.82%, which was not the best result that was obtained in the study. Additionally, the participants of the cognitive walkthrough, when asked which tool seemed to be the best for them, chose the Trust Wallet tool in second place. The Exodus tool was ranked first, with a majority of 1 vote.

The research was limited due to the small research group in the case of the questionnaires. More experienced cryptocurrency users generally want to remain anonymous in the decentralized world of cryptocurrencies, which may be the reason for a small research group. In the future, based on the data collected in the questionnaires, it is possible to conduct an in-depth analysis of the collected data.

References

[1] P. Marszałek, Kryptowaluty–pojęcie, cechy, kontrowersje. Studia BAS 1(57) (2019) 105-125.

[2] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review (2008) 21260.

[3] W.K Härdle, C.R Harvey, R.C. Reule, Understanding cryptocurrencies, Journal of Financial Econometrics 18, (2020) 181–208.

[4] U. Mukhopadhyay, A. Skjellum, O. Hambolu, J. Oakley, L. Yu, R. Brooks, A brief survey of cryptocurrency systems, 2016 14th annual conference on privacy, security and trust (PST) (2016) 745-752.

[5] A. Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder, Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction, Princeton University Press, 2016.

[6] S. Jokić, A. S. Cvetković, S. Adamović, N. Ristić, P. Spalević, Comparative analysis of cryptocurrency wallets vs traditional wallets, Ekonomika 65 (2019) 65-75.

[7] A. Sołoma, K. Spychalski, Zastosowanie bitcoinów we współczesnej gospodarce elektronicznej, Roczniki Kolegium Analiz Ekonomicznych Szkoły Głównej Handlowej 45 (2017) 227-240.

[8] M. Moniruzzaman, F. Chowdhury, M. S. Ferdous, Examining usability issues in blockchain-based cryptocurrency wallets, International Conference on Cyber Security and Computer Science (2020) 631-643.

[9] A. Kazerani, D. Rosati, B. Lesser, Determining the usability of bitcoin for beginners using change tip and coinbase, Proceedings of the 35th ACM International Conference on the Design of Communication (2017) 1-5.

[10] K. Cirstoiu, T. Guarda, L. Reyes, D. González, Cryptocurrencies, a new version of money, 2019 14th Iberian Conference on Information Systems and Technologies (CISTI) (2019) 1-5.

[11] K. Krombholz, A. Judmayer, M. Gusenbauer, E. Weippl, The other side of the coin: User experiences with bitcoin security and privacy, International conference on financial cryptography and data security (2016) 555-580.

[12] A. R. Sai, J. Buckley, A. Le Gear, Privacy and security analysis of cryptocurrency mobile applications, 2019 Fifth Conference on Mobile and Secure Services (MobiSecServ) (2019) 1-6.

[13] Exodus tool page, https://www.exodus.com/, [01.08.2021].

[14] Trust Wallet tool page, https://trustwallet.com/, [01.08.2021].

[15] Ledger tool page, https://https://www.ledger.com/, [01.08.2021].

[16] MetaMask tool page, https://metamask.io/, [01.08.2021].

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 378–382
Received: 21 September 2021
Accepted: 18 Ocober 2021

Comparison of Android data storage methods

Porównanie sposobów przechowywania danych w systemie Android

Dominika Kamila Kornaś*

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

This article presents a comparison of data storage methods available in the Android SDK. Analyzes the following information storage methods: SQLite, Room, Content Providers, SharedPreferences and Preferences DataStore. The aim of the study is to find the relationship between the complexity of the data structure and the cost and efficiency of data storage with the use of given methods. For the purposes of the test, an Android application was created which performed basic data operations and measured their duration. As a result of the performance test, average read and write times for the given data types and sizes were obtained. The summary contains conclusions on the most optimal method of data storage depending on the functional requirements of the application.

Keywords: Android SDK; data storage, mobile device

Streszczenie

Artykuł przedstawia porównanie metod przechowywania danych dostępnych w Android SDK. Poddaje analizie następujące sposoby składowania informacji: SQLite, Room, Dostawcy treści, SharedPreferences oraz Preferences DataStore. Celem przeprowadzonego badania jest znalezienie zależności między złożonością struktury danych, a kosztem i efektywnością ich przechowywania danymi metodami. Na potrzeby testu stworzono aplikację dla systemu Android, która wykonywała podstawowe operacje na danych oraz mierzyła czas ich trwania. W wyniku testu wydajności otrzymano średnie czasy wykonania zapisu i odczytu dla zadanych typów oraz rozmiarów danych. Podsumowanie zawiera wnioski na temat najbardziej optymalnego sposobu przechowywania danych w zależności od wymagań funkcjonalnych aplikacji.

Słowa kluczowe: Android SDK; przechowywanie danych; urządzenie mobilne

*Corresponding author

Email address: **dominika.kornas@outlook.com** (D. K. Kornaś)

1. Wstęp

Nieustanna ewolucja rynku mobilnego stała się przyczyną wykształcenia w społeczeństwie potrzeby ciągłego dostępu do informacji. Urządzenia mobilne będąc w stanie zapewnić dostęp do Internetu, a tym samym prawie nieograniczonej ilości danych, stały się narzędziem dystrybucji informacji. Funkcjonalność dostarczana za pośrednictwem aplikacji mobilnych obecnie znajduje zastosowanie w niemal każdej strefie życia społeczeństwa. Dzięki temu na przestrzeni dekady system Android osiągnął niepodważalny sukces.

Twórcy systemu Android, chcąc sprostać zmieniającym się potrzebom związanym z ciągłym rozwojem sektora aplikacji mobilnych w kolejnych dystrybucjach Android Software Development Kit proponowali udoskonalone lub całkiem nowe metody przechowywania danych. Współcześnie oferowane przez producentów urządzeń mobilnych technologie jedynie przyczyniły się do procesu przemian, pozwalając na wdrażanie wcześniej nieosiągalnych rozwiązań.

Wraz z rosnącą ilością danych, z których zaczęły korzystać aplikacje, poprawa wydajności metod przechowywania danych oraz skrócenie czasu wykonywania z ich użyciem podstawowych operacji było koniecznym dla twórców systemu Android celem, który należało osiągnąć. Jest to związane z bezpośrednim wpływem niniejszych czynników na ogólny odbiór aplikacji przez użytkowników końcowych.

2. Przegląd literatury

Zapewnienie optymalnego czasu operacji na danych stanowi temat rozważań Autorów publikacji „Porównanie możliwości wykorzystania oraz analiza wydajności baz danych na systemach mobilnych". Badaniu poddano następujące metody przechowywania informacji: SQLite, zewnętrzne serwery bazodanowe (MySQL, PostgreSQL, MicrosoftSQL Server), pliki płaskie, a także dla systemu Android – SharedPreferences, natomiast dla systemu Windows Mobile – Local Settings [1]. W pracy wykazano, że najkrótsze czasy dostępu do danych uzyskał Microsoft SQL Server spośród zewnętrznych serwerów. Dla danych typu prostego najlepszą wydajność osiągnęły ustawienia współdzielone oraz dla danych złożonych – SQLite.

W artykule „AndroBench: Benchmarking the Storage Performance of Android-Based Mobile Devices" przedstawiono wyniki testów wydajnościowych dla operacji zapisu, edycji i usuwania rekordów w bazie danych SQLite [2]. Rezultaty przeprowadzonego badania zostały wyrażone w liczbie transakcji na sekundę (ang. TPS – transactions per second). Czynność wstawiania danych posiada najniższy współczynnik TPS. Między uzyskanymi dla pozostałych operacji wynikami

wystąpiła silna zależność pod względem średnich czasów wykonania.

Autorzy publikacji „Performance analysis on Android SQLite database" zbadali wydajność przetwarzania danych niezaszyfrowanych oraz zaszyfrowanych przez lokalną bazę SQLite [3]. Stwierdzono, że najdłuższy czas wykonania dla rekordów niezaszyfrowanych posiada operacja wstawiania oraz pobierania danych. W przypadku informacji zaszyfrowanych niską efektywnością charakteryzuje się modyfikowanie i usuwanie elementów. Dodatkowo autorzy ustalili, że czas przetwarzania współbieżnego kilku zapytań w porównaniu do czasu niezbędnego na wykonanie pojedynczego zapytania, jest większy średnio o 40 – 44%.

Szczegółowe porównanie rozwiązań odpowiedzialnych za mapowanie obiektowo – relacyjne w systemie Android było przedmiotem publikacji „Porównanie systemów mapowania obiektowo relacyjnego greenDAO
i Room" [4]. Autor wykazał, że Room uzyskuje najkrótsze średnie czasy wykonania zapytań podczas operacji wstawiania danych. Natomiast znacząco mniejszą wydajność osiąga dla czynności aktualizowania i usuwania rekordów. Ponadto korzystanie z biblioteki Room nie wpływa na negatywnie na rozmiar aplikacji w porównaniu do greenDAO, jednakże przyczynia się do zwiększenia użycie pamięci operacyjnej oraz procesora urządzenia. Dodatkowo autor zwrócił uwagę na aspekty związane ze sposobem implementacji omawianych rozwiązań, podkreślając, że użycie Room wymaga podstawowej znajomości języka SQL.

3. Opis obiektu badań

Platforma Android jest przedstawicielem systemów z rodziny UNIX, mających u podstaw jądro Linuxa. Za pośrednictwem interfejsów bazujących na języku Java pozwala komunikować się z systemem operacyjnym
i warstwą sprzętową urządzenia. Zazwyczaj dane można przechowywać w pamięci wewnętrznej lub zewnętrznej urządzenia, przy czym standardowym podejściem jest przechowywanie danych poufnych aplikacji w pamięci wewnętrznej, ze względu na zmniejszone ryzyko utraty dostępu do nich [5 – 7].

Obecnie Android Software Development Kit posiada szereg rozwiązań pozwalających na przetwarzanie i magazynowanie informacji. W tabeli 1 przedstawiono metody przechowywania danych analizowane w ramach niniejszej publikacji.

Tabela 1: Zestawienie metod przechowywania danych

Metoda przechowywania	Typ danych
SQLite	dane ustrukturyzowane
Room	dane ustrukturyzowane
Dostawcy treści	dane ustrukturyzowane
	dane nieustrukturyzowane, tj. pliki audio, wideo, obrazy
Ustawienia współdzielone	pary klucz - wartość
Preferences DataStore	pary klucz - wartość

Biblioteka SQLite znalazła powszechne zastosowanie w aplikacjach dla systemu Android. Jako lekka wersja transakcyjnego silnika bazy danych SQL nie wymaga specjalnej konfiguracji ani serwera, dodatkowo może pracować na urządzeniach o niewielkich zasobach sprzętowych, powodując jedynie niewielkie zużycie pamięci oraz energii. Pliki, zawierające definicję bazy danych, tabel oraz dane są przechowywane w pamięci urządzenia. SQLite wspiera relacyjny model danych i obsługę dynamicznych, złożonych zapytań [8].

Niskopoziomowe operacje konieczne do obsługi bazy danych SQLite wyeliminowano wprowadzając bibliotekę Room, która pozwoliła na wydzielenie warstwy logiki biznesowej oraz bezpośrednie odwzorowanie obiektów do postaci tabel relacyjnej bazy danych za pomocą techniki mapowania obiektowo – relacyjnego. Połączenie z bazą danych, reprezentacja tabel oraz obsługa zapytań jest realizowana za pośrednictwem trzech głównych komponentów, kolejno: *@Database, @Entity* oraz *@Dao*. Ponadto wykorzystanie Room wprowadziło możliwość statycznej analizy kodu oraz weryfikacji poprawności zapytań w czasie kompilacji [9].

Zarówno SQLite, jak i Room, nie pozwalają na dostęp do danych aplikacji zewnętrznym procesom. Możliwość przetwarzania informacji, składowanych przez inne aplikacje na urządzeniu, oferują w systemie Android dostawcy treści. Dane mogą posiadać różną postać, np. tabele bazodanowe, obrazy, pliki audio i wideo oraz pochodzić z kilku źródeł. Jedynym wymogiem jest statyczna, trwała lokalizacja, do której aplikacja będzie miała dostęp, który odbywa się za pośrednictwem schematu URI (Uniform Resource Identifier). Na jego podstawie aplikacja może uzyskać ogólny lub czasowy dostęp do wybranych zasobów. Dostawcy treści stanowili warstwę abstrakcji dla bazy danych SQLite przed wprowadzeniem biblioteki Room. Dodatkowo, jako jedyny
z analizowanych sposobów przechowywania danych, pozwalają na zachowanie danych nawet po odinstalowaniu aplikacji z pamięci urządzenia [10].

Przechowywanie prostych typów danych w postaci klucz – wartość w systemie Android może zostać zrealizowane za pomocą dwóch mechanizmów: ustawień współdzielonych (ang. SharedPreferences) lub wprowadzonego w 2020 roku jako ich następca – Preferences DataStore. Oba rozwiązania dostarczają możliwość wykonania jedynie operacji zapisu i odczytu danych. Operacja modyfikacji polega na ponownym zapisie nowej wartości o tym samym kluczu. Natomiast całkowite usunięcie danych z pamięci urządzenia nie jest możliwe, dopóki aplikacja nie zostanie całkowicie odinstalowana. Ustawienia współdzielone opierają się głównie na synchronicznych operacjach na danych, w przeciwieństwie do Preferences DataStore, który dzięki wykorzystaniu komponentów języka Kotlin, zapewnia ich asynchroniczne wykonanie. Dodatkowo ustawienia współdzielone wspierają dostęp zewnętrznych procesów do składowanych informacji [11 – 12].

4. Metoda badań

Celem badań było przetestowanie wydajności wybranych metod przechowywania danych dostępnych w Android Software Development Kit. Skupiono sią na analizie efektywności wykonania operacji wstawiania oraz pobierania informacji, ponieważ jako jedyne są w sposób bezpośredni dostarczane przez wszystkie analizowane technologie.

Do przeprowadzenia testów konieczne było opracowanie struktur danych, które zostaną poddane analizie oraz skonstruowanie generatora, który pozwoli na ich uzyskanie. Rezultatem jego działania była kolekcja elementów w postaci listy o zadanym rozmiarze. Przetestowano wydajność aplikacji podczas przetwarzania 100, 1000, 10000 oraz 50000 rekordów. Natomiast typy i zakresy danych wykorzystane podczas badania znalazły się w tabeli 2.

Tabela 2: Typy i zakresy danych biorące udział w badaniu

Typ danych	Zakres danych
liczba całkowita	32 bity
liczba rzeczywista	64 bity
ciąg znakowy	500 znaków
obiekt	liczba całkowita 32 – bitowa liczba rzeczywista 64 – bitowa ciąg 500 – znakowy pole typu boolean
para klucz – wartość	klucz typu liczbowego wartość w postaci ciągu 100 – znakowego

W przypadku SQLite, Room oraz dostawców treści, będących pośrednikiem dla SQLite, dane zostały odwzorowane na rekordy bazodanowe, np. pola obiektów odtworzono jako odpowiadające im kolumny w tabeli. Metodami, które wymagały parsowania danych okazały się SharedPreferences oraz Preferences DataStore. Jako rezultat przyjęto średnią z uzyskanych podczas 5 pomiarów wyników.

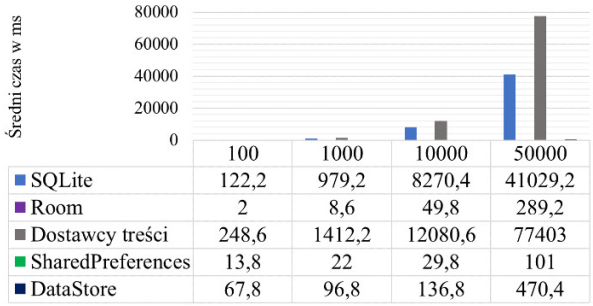
Każdy scenariusz badawczy został zrealizowany na podstawie ogólnego schematu:

1. Wybór parametrów generatora danych pseudolosowych: liczby rekordów, typu i zakresu danych.
2. Generowanie danych.
3. Wybór jednej z operacji: wstawianie lub odczyt rekordów.
4. Pomiar czasu wykonania wybranej operacji dla każdej z analizowanych metod przechowywania danych.

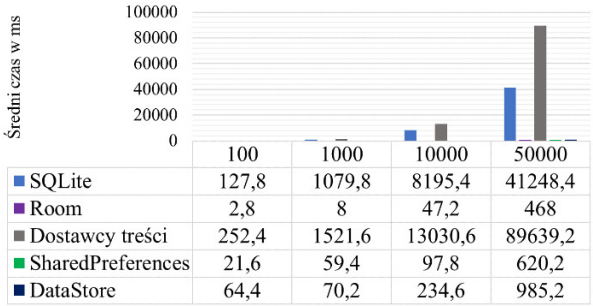
Czas wykonania wybranych operacji był mierzony w każdym przypadku od momentu wysłania żądania o jej przeprowadzeniu do otrzymania odpowiedzi o sukcesie wykonania. Dla operacji wymagających dodatkowego przygotowania danych, czas parsowania był wliczany do końcowego wyniku.

5. Wyniki

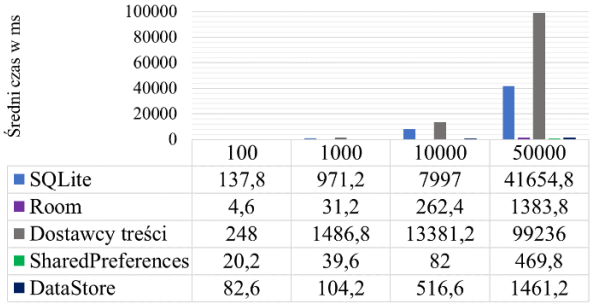
Poniżej przedstawiono wyniki przeprowadzonych badań. Wykresy widoczne na rys. 1 do rys. 5 prezentują kolejno rezultaty pomiarów czasu dla operacji wstawiania danych następujących typów: liczby całkowite, liczby rzeczywiste, ciągi znakowe, obiekty oraz pary klucz – wartość. Zaobserwowano, że liczba rekordów wpływa na kolejność analizowanych metod przechowywania danych pod względem wydajności operacji zapisu. Przypadki testowe, w których przetwarzano 100 oraz 1000 rekordów wykazały, że najlepszy średni czas zapisu osiągał Room dla wszystkich typów danych. Na kolejnym miejscu znalazły się ustawienia współdzielone. Jednakże po zwiększeniu liczby rekordów do 10000 ustawienia współdzielone uzyskały lepszy czas przetwarzania danych niż Room. Jedynym wyjątkiem były liczby rzeczywiste, gdzie Room uzyskał krótsze średnie czasy dla każdej analizowanej liczby rekordów. Podobna sytuacja miała miejsce dla 50000 elementów. Dla wszystkich analizowanych ilości oraz typów danych na kolejnych miejscach znalazły się DataStore, SQLite oraz dostawcy treści. Dodatkowo DataStore nie był w stanie wykonać zapisu 50000 obiektów oraz par klucz – wartość podczas żadnej z wykonanych prób. Tę sytuację zobrazowano na wykresach wynikiem 0 dla tego scenariusza testowego.



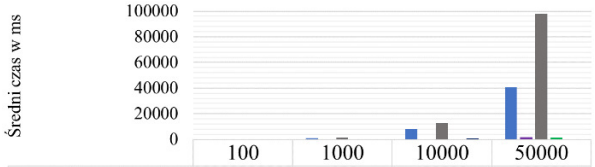
Rysunek 1: Porównanie średnich czasów operacji wstawiania danych dla liczb całkowitych w zależności od liczby rekordów.



Rysunek 2: Porównanie średnich czasów operacji wstawiania danych dla liczb rzeczywistych w zależności od liczby rekordów.

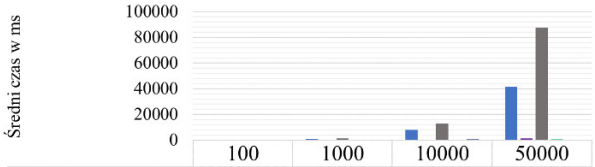


Rysunek 3: Porównanie średnich czasów operacji wstawiania danych dla ciągów znakowych w zależności od liczby rekordów.



	100	1000	10000	50000
SQLite	117,8	945,8	7953,4	40819,6
Room	6	30,8	323	1560,4
Dostawcy treści	229	1461	12866,8	97816,6
SharedPreferences	27,4	64,4	236,6	1314,4
DataStore	80,4	135,8	667	0

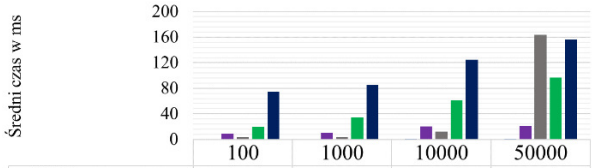
Rysunek 4: Porównanie średnich czasów operacji wstawiania danych dla obiektów w zależności od liczby rekordów.



	100	1000	10000	50000
SQLite	113	853,2	7971,2	41336
Room	4,4	26,6	284,6	1354,4
Dostawcy treści	242,6	1388,8	12601,4	87716,2
SharedPreferences	19,6	59,8	133,2	647,2
DataStore	76,8	163	617,8	0

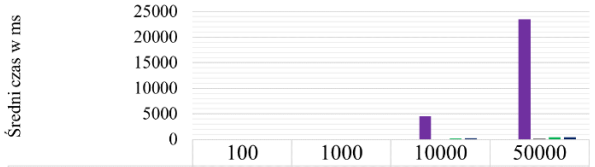
Rysunek 5: Porównanie średnich czasów operacji wstawiania danych dla par klucz – wartość w zależności od liczby rekordów.

Na wykresach (rys. 6 – rys. 10) znalazły się wyniki analizy operacji odczytu. W każdym przypadku testowym najkrótszy średni czas osiąga SQLite, na kolejnym miejscu klasyfikują się dostawcy treści. Dla liczb całkowitych oraz obiektów następne pozycje zajmują Room, SharedPreferences oraz DataStore. Taka sama kolejność zachowana jest dla pozostałych typów danych podczas wykonywania operacji pobierania 100 oraz 1000 rekordów. Natomiast dla 10000 i 50000 elementów Room uzyskuje najdłuższy średni czas.



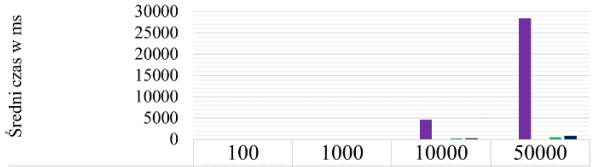
	100	1000	10000	50000
SQLite	0,56	0,75	1,26	1,32
Room	9,6	10	20,2	21,2
Dostawcy treści	3,2	3,4	12,4	164
SharedPreferences	19,6	34	61,8	97
DataStore	74,4	84,8	125	156,4

Rysunek 6: Porównanie średnich czasów operacji odczytywania danych dla liczb całkowitych w zależności od liczby rekordów.



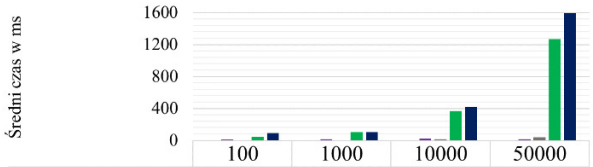
	100	1000	10000	50000
SQLite	0,49	0,55	1,17	1,1
Room	12	15,2	4541,2	23465,8
Dostawcy treści	3,8	4	12,2	164,2
SharedPreferences	28	75	167,2	398,4
DataStore	57,4	85,4	208,8	410,6

Rysunek 7: Porównanie średnich czasów operacji odczytywania danych dla liczb rzeczywistych w zależności od liczby rekordów.



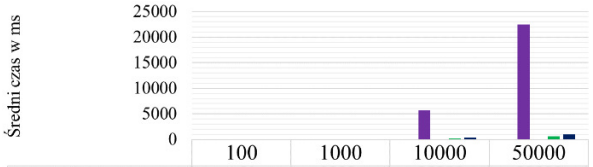
	100	1000	10000	50000
SQLite	0,45	0,56	1,31	1,25
Room	12,6	22,8	4555	28445,2
Dostawcy treści	3,6	6	12,6	31
SharedPreferences	24	41	130,4	491
DataStore	87,6	94,2	263,2	810

Rysunek 8: Porównanie średnich czasów operacji odczytywania danych dla ciągów znakowych w zależności od liczby rekordów.



	100	1000	10000	50000
SQLite	0,48	0,5	1,15	1,37
Room	11	14,2	21,6	18,2
Dostawcy treści	4	5,6	14,8	37,2
SharedPreferences	44	101,8	365,6	1265,6
DataStore	93,2	105,4	421,2	1593,6

Rysunek 9: Porównanie średnich czasów operacji odczytywania danych dla obiektów w zależności od liczby rekordów.



	100	1000	10000	50000
SQLite	0,42	0,68	1,07	0,93
Room	11,2	21	5621,6	22510,8
Dostawcy treści	3,4	4,6	14	34,2
SharedPreferences	25,8	54,2	185,4	628,8
DataStore	77	84,2	315,4	975,8

Rysunek 10: Porównanie średnich czasów operacji odczytywania danych dla par klucz – wartość w zależności od liczby rekordów.

6. Wnioski

Jednoznaczne wskazanie najbardziej uniwersalnej metody przechowywania danych spośród tych, które udostępnia Android SDK nie jest możliwe. Należy wziąć pod uwagę nie tylko kluczowe czynniki, jakie stanowią typ informacji oraz ich ilość, ale także szczegółowe wymagania biznesowe tworzonej aplikacji. Pozwoli to wyeliminowanie najmniej optymalnych dla danego oprogramowania rozwiązań już na etapie projektowania.

W związku z powyższym najbardziej optymalne metody dla aplikacji wykonujących częste operacje wstawiania danych stanowią Room oraz SharedPreferences. Dodatkowo należy zwrócić uwagę, że obecnie promowaną i zalecaną przez dokumentację metodą jest Room, będący jednym z przedstawicieli systemów ORM. Redukuje tym samym liczbę zapytań koniecznych do napisania przez programistę w języku SQL. Ustawienia współdzielone korzystają z danych typu klucz – wartość.

W związku z tym przechowywanie za ich pomocą złożonych struktur informacji wymaga dodatkowego nakładu pracy np. parsowania danych.

DataStore jako jedyna metoda przechowywania danych nie ukończył operacji zapisu wywołując błąd braku pamięci operacyjnej. Związane jest to z koniecznością utworzenia puli wątków asynchronicznych, z których każdy posiada zarezerwowaną ilość zasobów. Ponadto, aby skorzystać z tego rozwiązania w aplikacji tworzonej w języku Java, należy użyć biblioteki RxJava. Tylko w języku Kotlin jest to metoda dostarczana natywnie.

Podczas wykonania operacji odczytywania danych szczególnymi wynikami wykazał się SQLite, uzyskując istotnie krótsze średnie czasy przetwarzania zapytania niezależnie od typu i liczby rekordów. Kolejną pozycję ze względu na otrzymane wyniki zajmują dostawcy treści. Należy jednak zaznaczyć, że niniejsze metody przechowywania danych są jednymi ze starszych rozwiązań, które w nowych aplikacjach najczęściej są zastępowane nowszymi propozycjami.

Uzyskane dla operacji zapisu i odczytu danych wyniki są przeciwne. Metody uzyskujące najlepsze średnie czasy pobierania, otrzymały najgorsze rezultaty podczas wstawiania rekordów. Ostatecznie to do programisty należy wybór rozwiązania, które spełni wymagania funkcjonalne oraz zapewni maksymalne doświadczenie użytkownika podczas korzystania z aplikacji.

Literatura

[1] M. Grudzień, K. Korgol, D. Gutek, Porównanie możliwości wykorzystania oraz analiza wydajności baz danych na systemach mobilnych, Journal of Computer Sciences Institute 2 (2016) 133-139, https://doi.org/10.35784/jcsi.129.

[2] J. Kim, J. Kim, AndroBench: Benchmarking the Storage Performance of Android-Based Mobile Devices, Frontiers in Computer Education (2012) 667-674, https://doi.org/10.1007/978-3-642-27552-4_89.

[3] N. Obradovic, A. Kelec, I. Dujlovic, Performance analysis on Android SQLite database, 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH) (2019) 1-4, https://ieeexplore.ieee.org/document/8717652.

[4] M. Lewiński, Porównanie systemów mapowania obiektowo relacyjnego greenDao i Room, Journal of Computer Sciences Institute 14 (2020) 43-47, https://doi.org/10.35784/jcsi.1574.

[5] N. Gandhewar, R. Sheikh, Google Android: An Emerging Software Platform For Mobile Devices, International Journal on Computer Science and Engineering (2010) 12–17.

[6] K. Honcharenko, J. Smołka, Analiza rozwoju środowiska uruchomieniowego systemu Android. Journal of Computer Sciences Institute 12 (2019) 246–251, https://doi.org/10.35784/jcsi.504.

[7] Metody przechowywania danych i plików – dokumentacja Android, https://developer.android.com/training/data-storage, [04.05.2021]

[8] S. T. Bhosale, T. Patil, P. Patil, SQLite: Light Database System. International Journal of Computer Science and Mobile Computing 4 (2015) 882–885.

[9] Biblioteka Room – dokumentacja Android, https://developer.android.com/training/data-storage/room, [07.05.2021]

[10] Dostawcy treści – dokumentacja Android, https://developer.android.com/guide/topics/providers/content-providers, [04.05.2021]

[11] Ustawienia współdzielone – dokumentacja Android, https://developer.android.com/training/data-storage/shared-preferences, [07.05.2021]

[12] DataStore – dokumentacja Android, https://developer.android.com/topic/libraries/architecture/datastore, [04.05.2021]

JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 383–390
Received: 23 September 2021
Accepted: 26 October 2021

An analysis of the possibility of realization steganography in C#

Badanie możliwości realizacji steganografii w języku C#

Piotr Pawlak*, Jakub Bogdan Podgórniak*, Grzegorz Kozieł

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The computing power of modern computers is sufficient to break many cryptographic keys, therefore it is necessary to create an additional security layer which hides the very fact of transmitting a secret message. For this purpose, steganographic methods can be used. The article is devoted to the analysis of the possibility of implementing digital images steganography with the use of the C # programming language. Firstly, existing libraries and mathematical transformations which can help with performing steganography were found. Also, own code solutions were implemented. In order to objectively evaluate the methods of data hiding, the parameters describing the degree of distortion of transforms and hidden images were calculated. Subsequently, optimal solutions for specific problems were identified and demonstrational data hiding was performed. Based on the obtained results, it can be concluded that it is possible to successfully implement steganography in the C # language. There are many ready-made libraries and tools, the effectiveness of which has been verified in the conducted analysis. Due to the contradictory of stenographic requirements, it is not possible to meet all of them optimally, i.e. undetectability, resistance to destruction and information capacity. For this reason, it is not possible to clearly indicate the best solutions. In order to achieve satisfactory results, one should look for compromises between the set requirements.

Keywords: steganography; C# programming language; data hiding; digital image processing

Streszczenie

Moc obliczeniowa współczesnych komputerów jest wystarczająca do łamania wielu zabezpieczeń kryptograficznych, w związku z powyższym konieczne jest utworzenie dodatkowej warstwy bezpieczeństwa polegającej na ukryciu samego faktu przekazywania tajnej wiadomości. W tym celu mogą zostać wykorzystane metody steganograficzne. Artykuł poświęcono analizie możliwości realizacji steganografii w obrazach cyfrowych przy wykorzystaniu języka programowania C#. Wytypowane zostały istniejące biblioteki, przekształcenia matematyczne, a także zaimplementowane zostały własne rozwiązania. W celu dokonania obiektywnej oceny metod ukrywania danych obliczono parametry opisujące stopień zniekształceń transformat oraz ukrywanych obrazów. Następnie wyłoniono optymalne rozwiązania dla konkretnych problemów oraz przeprowadzono demonstracyjne ukrycie danych. Na podstawie otrzymanych rezultatów można stwierdzić, że możliwe jest kompleksowe zrealizowanie steganografii w języku C#. Istnieje wiele gotowych bibliotek i narzędzi, których skuteczność została zweryfikowana w przeprowadzonej analizie. Z racji sprzeczności wymagań stenograficznych nie jest możliwe optymalne spełnienie ich wszystkich tj.: niewykrywalności, odporności na zniszczenie i pojemności informacyjnej. Z tego powodu nie jest możliwe jednoznaczne wskazanie najlepszych rozwiązań. Aby osiągnąć zadowalające rezultaty należy szukać kompromisów pomiędzy stawianymi wymaganiami.

Słowa kluczowe: steganografia; język C#; ukrywanie danych; przekształcanie obrazów cyfrowych

*Corresponding author

Email address: piotr.pawlak3@pollub.edu.pl (P. Pawlak), jakub.podgorniak@pollub.edu.pl (J. B. Podgórniak)

1. Wstęp

Steganografia jest nauką zajmującą się ochroną cennej informacji poprzez ukrywanie samego faktu jej przekazywania. Dzięki temu osoby postronne nie są w stanie wykryć obecności przekazu, co jest szczególnie istotne w obliczu obecnego postępu technologicznego. Narzędzia łamiące szyfry mają przewagę nad mechanizmami zabezpieczającymi przepływ danych. W momencie, gdy osoby nieupoważnione nie są świadome faktu przekazywania wiadomości, nie mogą podjąć się próby jej przechwycenia. Skutkuje to wypełnieniem braków dotychczasowych systemów zabezpieczeń. Istnieje wiele technik steganografii, co jest jej istotnym atutem w kontekście doboru odpowiedniej techniki do zaistniałych okoliczności, tak aby spełniała ona wymagania ukrywającego i pozwoliła na skuteczną i tajną komunikację [13].

Wraz z rosnącą ilością danych przekazywanych przez Internet, a także z ciągłym rozpowszechnianiem się metod transmisji wszelkich rodzajów informacji, zwiększyła się liczba sposobów na jakie można realizować ukrywanie informacji, a także sama potrzeba niejawnego przekazywania danych. Wszystkie te czynniki przyczyniły się do wzrostu zainteresowania steganografią, dzięki czemu dostępnych jest wiele narzędzi i rozwiązań pozwalających na jej przeprowadzenie. Wzrost popularności tej dziedziny nauki poskutkował publikacją wielu pozycji naukowych traktujących o problemie zachowania bezpieczeństwa w tajnej komunikacji [10][12][18].

Obecna sytuacja pokazuje, że steganografia i związane z nią ukryte kanały transmisji danych będą zyskiwać na znaczeniu w kontekście bezpieczeństwa przekazywanych informacji. Jest to związane ze słabościami istniejących systemów bezpieczeństwa. W związku z tym ważne jest ciągłe przeprowadzanie badań dotyczących steganografii, tak aby lepiej poznać tę dziedzinę nauki.

W artykule przedstawiono i przeanalizowano wybrane metody realizacji steganografii przy wykorzystaniu obiektowego języka programistycznego C#. Dodatkowo określono skuteczność wybranych narzędzi z wykorzystaniem obiektywnych jak również subiektywnych form oceny, tak aby wyznaczyć optymalne rozwiązania dla poszczególnych problemów pod względem trójkąta sprzeczności wymagań.

2. Analiza literatury

W artykułach, zebranych podczas przeglądu literatury, poruszone zostały kwestie wykorzystania metod steganografii takich jak m. in. przetwarzanie obrazu w celu ukrycia danych. Informacje te, po analizie, zostaną wykorzystane jako jeden z obszarów badawczych.

W publikacji „Image steganography techniques: an overview" [12] autorzy skupili się na wykorzystaniu obrazu jako nośnika informacji, dlatego przedstawiona została taksonomia obecnych technik steganografii dla plików graficznych. Omówione techniki przetwarzania obrazów zostały przeanalizowane i omówione pod kątem zdolności ukrywania informacji w plikach obrazów, ilości informacji, które można ukryć, a także odporności na różne ataki, mające na celu zniszczenie przekazywanej informacji.

W przeglądzie „Reversible steganography techniques: A survey" [14] autorzy przygotowali kompleksowe zestawienie metod przeprowadzenia steganografii na bazie obrazów. Opisane techniki to rozszerzanie różnic (Difference Expansion), przesuwanie histogramów (Histogram-Shifting), porządkowanie zbioru próbek wektorowych (Pixel-Value-Ordering), schematy bazujące na dwóch obrazach oraz schematy bazujące na interpolacji. Każda z metod została szczegółowo opisana oraz zobrazowana przykładami jej wykorzystania, co czyni z tego przeglądu cenne źródło informacji podczas zagłębiania się w naukę jaką jest steganografia.

Autorzy artykułu „Analiza właściwości metod steganografii odwracalnej" [19] dokonali analizy trzech metod steganografii odwracalnej z wykorzystaniem obrazów cyfrowych jako nośnika. Wspomniane metody to: rozszerzanie różnić, obszar zainteresowania oraz modyfikacja histogramu. Każda ze zbadanych metod okazała się mieć nieco inną charakterystykę, co pozwoliło autorom wysnuć wniosek, że reagują one w różny sposób na cechy obrazu nośnego. Opracowanie przedstawia również szczegółowe wyniki pomiaru parametrów każdej z testowanych metod oraz prezentuje sposób porównania ich między sobą.

Artykuł "An overview of image steganography" [15] zawiera przegląd różnych technik steganografii z wykorzystaniem nośnika w postaci plików graficznych. Autorzy jako cel postawili sobie odnalezienie odpowiedzi na pytanie, która technika ukrywania danych jest najbardziej odpowiednia w zależności od różnych zastosowań oraz to jakie cechy pozwalają zidentyfikować dobry algorytm steganograficzny. Omawiane przez autorów techniki to: ukrywanie danych w najmniej znaczącym bicie bitmapy, steganografia z wykorzystaniem kompresji JPEG, metoda mozaiki oraz magazynowanie informacji w zdjęciu jako szum Gaussowski.

3. Metodyka badawcza

Na podstawie przeglądu literatury określono jakie przekształcenia są wykorzystywane w steganografii. Dla tychże przekształceń zidentyfikowano istniejące dla nich implementacje w języku C#. Następnie przy użyciu tych rozwiązań przeprowadzono transformacje na identycznych obrazach źródłowych. Obliczone zostały parametry służące porównaniu skuteczności danych metod.

Dla każdego przekształcenia wybrane zostały biblioteki dostępne w języku C#. Wykorzystane biblioteki to: Accord.NET [3], Aforge.NET [4], Math.NET [5], NWaves [6], UMapx[7], Universal.Common.Mathematics [8], TrentTobler.Algorithms.FourierTransform [9] oraz inne implementacje znalezione w repozytoriach GitHub. Jeżeli dla danej operacji nie była dostępna wystarczająca liczba gotowych rozwiązań, zaimplementowane zostało własne. Dzięki temu możliwe było wskazanie najlepszego narzędzia do rozwiązania konkretnego problemu. Z bazy flickr.com pobrano 100 publicznie dostępnych obrazów i na każdym z nich, przeprowadzono transformację z wykorzystaniem wybranych wcześniej rozwiązań. W kolejnym kroku, tak uzyskane transformaty poddano przekształceniu odwrotnemu do tego, z którego została ona uzyskana. Otrzymane obrazy wynikowe porównano z ich obrazami źródłowymi. Uzyskane wyniki uśredniono i porównano w obrębie metod, oraz pomiędzy każdą z metod.

Parametry wytypowane do przeprowadzenia oceny zastosowania danych bibliotek to: błąd średniokwadratowy (ang. Mean Square Error), znormalizowany błąd średniokwadratowy (ang. Normalized MSE) oraz szczytowy stosunek sygnału do szumu (ang. Peak SignaltoNoise Ratio). Pozwolą one na obiektywne porównanie obrazów wynikowych, uzyskanych poprzez zastosowanie poszczególnych bibliotek, z obrazami źródłowymi, a następnie określenie optymalnych rozwiązań.

Do przeprowadzenia analizy możliwości realizacji steganografii w języku C# wykorzystano komputer stacjonarny z zainstalowanym systemem Windows 10 64 bit w wersji 10.0.19042, posiadający procesor AMD Ryzen 5 2600 oraz pamięć RAM G.SKILL 16GB (2x8GB) 3000MHz CL16 Aegis. Do implementacji przekształceń matematycznych, a także algorytmów służących ukrywaniu informacji wykorzystano język C# w wersji 8 oraz środowisko programistyczne .NET w wersji 5.0.7

4. Analiza możliwości realizacji przekształceń obrazów w C#

W procesie realizacji steganografii wykorzystującej grafiki komputerowe jako nośnik ukrywanych danych, możliwe jest wykorzystanie całego spektrum przekształceń matematycznych, takich jak: dyskretna transformacja Fouriera, szybka transformacja Fouriera, dyskretna transformacja cosinusowa, transformacja falkowa oraz transformacji do nich odwrotnych. W przypadku formatów obrazów cyfrowych takich jak BMP, gdzie nie jest stosowana kompresja, możliwe jest również przeprowadzenie steganografii opierającej się na fundamentalnych właściwościach przetwarzania cyfrowego, czyli z użyciem operacji binarnych. Kluczowym aspektem dobrze zrealizowanego ukrycia danych jest ich niewykrywalność. W związku z tym należy zbadać, czy podane przekształcenia mogą zostać zrealizowane za pomocą języka programowania C#, tak aby otrzymany w ich wyniku błąd wprowadzał zniekształcenia trudne do wykrycia gołym okiem. Poza subiektywną oceną obrazów uzyskanych w wyniku transformacji, wyliczone zostały parametry określające stopień błędu.

4.1 Operacja podmiany bitów piksela

W grafice komputerowej wykorzystywane bitmapy zbudowane są z tablic liczb całkowitych przedstawionych w formie binarnej. Większość języków programowania, w tym C#, pozwala na manipulowanie wartościami pojedynczych bitów danych. Możliwość ta może zostać wykorzystana do przeprowadzenia steganografii. Jedną z najbardziej popularnych metod ukrywania danych jest metoda najmniej znaczącego bitu.

Aby zobrazować możliwość realizacji steganografii z użyciem manipulacji wartości bitów wykonano następujące czynności: wczytany został obraz źródłowy, wartość najbardziej znaczącego bitu została odwrócona przy wykorzystaniu operacji alternatywy rozłącznej, po uzyskaniu grafiki z przekłamanymi bitami przeprowadzono proces odwrotny, tak by odzyskać pierwotne dane. Badanie to ma na celu zweryfikowanie czy obraz został zniekształcony poprzez przeprowadzenie badanego przekształcenia, a następnie jego odwrócenie.

Zgodnie z oczekiwaniami obraz otrzymany po przekształceniu, a następnie przywróceniu wartości początkowych jest identyczny do obrazu źródłowego. Zatem w momencie wykonania manipulacji wartości poszczególnych bitów i późniejszych operacji odwrotnych przeprowadzonych na dokładnie tych samych bitach, uzyskany efekt jest taki jakby przekształcenie nigdy nie miało miejsca. Fakt ten potwierdzają otrzymane wyniki porównania nośnika oryginalnego z nośnikiem po przeprowadzeniu opisanej powyżej operacji, które przedstawiono w Tabeli 1.

Tabela 1: Wartości parametrów dla operacji bitowej XOR

Metoda	MSE	NMSE	PSNR [dB]
XOR	0	0%	-

4.2 Dyskretna Transformacja Fouriera

Transformacja Fouriera jest matematyczną transformacją, która transformuje sygnał z dziedziny czasu do dziedziny częstotliwości. Wynikiem tej transformacji jest transformata Fouriera i zawiera ona informacje o częstotliwościach składowych sygnału źródłowego [16]. Transformacja ta jest przydatna przy analizie częstotliwości sygnałów. W przypadku przetwarzania sygnałów cyfrowych koniecznie jest zastosowanie dyskretnej transformacji Fouriera.

Możliwe jest również przeprowadzenie odwrotnej transformacji Fouriera, która pozwala odzyskać oryginalne dane z transformaty. Zatem przekształca ona sygnał z dziedziny częstotliwości do dziedziny czasu.

Znalezione zostały dwie istniejące biblioteki dostarczające implementację dyskretnej transformacji Fouriera. Pierwszą z nich jest TrentTobler.Algorithms.FourierTransforms (TAF), a druga to Universal.Common.Mathematics (UCM). Dodatkowo algorytm DFT został zaimplementowany od podstaw jako trzecie rozwiązanie.

Aby dokonać obiektywnej oceny wybranych bibliotek dla każdej otrzymanej grafiki wyliczone zostały parametry: MSE, NMSE oraz PSNR. Otrzymane wyniki zostały przedstawione w Tabeli 2.

Tabela 2: Wartości parametrów dla poszczególnych metod

Metoda	MSE	NMSE	PSNR [dB]
TAF	0,000146	0,0379%	38,36
UCM	0,000154	0,0401%	38,11
Własne	0,000154	0,0401%	38,11

Z otrzymanych wartości wynika, że najmniejszy błąd został otrzymany przy użyciu biblioteki TAF. Pozostałe dwa rozwiązania wygenerowały identyczne wyniki, co może wskazywać na to, że wykorzystują one identyczny model matematyczny.

Wszystkie wykorzystane implementacje DFT oraz transformacji odwrotnej, mogą zostać z sukcesem wykorzystane do przeprowadzenia steganografii.

4.3 Szybka Transformacja Fouriera

Szybka transformacja Fouriera (ang. Fast Fourier Transform, FFT) to algorytm wykorzystywany do określenia dyskretnej transformaty Fouriera, a także transformaty do niej odwrotnej. Ma ona taką przewagę nad transformacją Fouriera, że w wyniku wykorzystania algorytmu Cooley-Tukeya, działającego zgodnie z maksymą „dziel i rządź", jej złożoność obliczeniowa jest mniejsza, a zatem przeprowadzenie transformacji będzie trwało krócej. Porównując, złożoność obliczeniową dyskretnej transformacji Fouriera to O(N2), a złożoność szybkiej transformacji Fouriera wynosi O(Nlog2N), widoczny jest duży zysk czasu w przypadku wybrania takiej implementacji [11].

Po przeprowadzeniu analizy dostępnych bibliotek wytypowano 3 rozwiązania, dostarczające implementację algorytmu FFT. Pierwszą z wytypowanych bibliotek

jest AForge.NET. Jej ogólnym przeznaczeniem jest dostarczenie szkieletu programistycznego dla programistów zajmujących się sztuczną inteligencją oraz cyfrowym rozpoznawaniem obrazów, przy których to FFT znajduje zastosowanie. Kolejną biblioteką jest Accord.NET. W tym przypadku kluczowym obszarem, dla którego wykorzystano algorytm FFT jest uczenie maszynowe. Ostatnią z testowych bibliotek jest Math.NET. Implementacja FFT w tej bibliotece pozwala na zastosowanie różnych trybów skalowania oraz określenie znaku wykładnika.

Aby obiektywnie porównać wybrane biblioteki dla każdej otrzymanej grafiki wyliczone zostały parametry MSE, NMSE oraz PSNR. Otrzymane wyniki zostały przedstawione w Tabeli 3.

Tabela 3: Wartości parametrów dla poszczególnych metod

Metoda	MSE	NMSE	PSNR [dB]
A.Forge.NET	0,000156	0,0404%	38,082
Accord.NET	0,00015	0,0391%	38,222
Math.NET domyślny	0,000149	0,0386%	38,275
Math.NET skalowanie	0,000151	0,0391%	38,222
Math.NET nieskalowany	0,319949	83,0941%	4,949

Ze wszystkich zbadanych metod najmniejszy błąd wprowadza algorytm biblioteki Math.NET wykorzystany w trybie domyślnym. Drugim najbardziej efektywnym rozwiązaniem jest to zaoferowane przez bibliotekę Accord.NET. Największy błąd, widoczny również na grafice, wystąpił przy wykorzystaniu biblioteki Math.NET w trybie bez skalowania. Porównując dyskretną transformację Fouriera do szybkiej transformacji Fouriera warto zauważyć, że najmniejszy poziom zniekształceń uzyskano przy wykorzystaniu pierwszej z nich. Jednakże, różnica pomiędzy najlepszym wynikiem DFT, a FFT jest znikoma i niezależnie od wybranej metody, nie powinna wprowadzać ona błędu, który znacząco obniżyłby jakość przeprowadzonej steganografii. Oznacza to, że z powodzeniem można zastosować obie te metody w celu ukrycia faktu przekazywania tajnych informacji

4.4 Dyskretna transformacja cosinusowa

Następnym przekształceniem, mogącym posłużyć do przeprowadzenia steganografii z wykorzystaniem obrazu jako nośnika danych jest dyskretna transformacja cosinusowa (ang. discrete cosine transform, DCT). DCT jest to rodzaj transformacji blokowej przeprowadzonej na wartościach dyskretnych [1][19]. Dzięki zastosowaniu transformacji odwrotnej do DCT możliwe jest przywrócenie pierwotnych danych.

DCT jest powszechnie używana do kompresji obrazów cyfrowych. Na jej podstawie opracowany został format JPEG. Struktura tego formatu pozwala na ukrycie danych w współczynnikach DCT, dlatego format ten jest odpowiednim kandydatem do bycia bazą operacji steganograficznej.

Spośród dostępnych rozwiązań, pozwalających na przeprowadzenie dyskretnej transformacji cosinusowej, wytypowano trzy następujące: Accord.NET, Universal.Common.Mathematics oraz autorską implementację udostępnioną przez nauczyciela akademickiego dr Vinyaka Bharadi (VB).

Dla przeprowadzenia obiektywnej oceny wyliczone zostały parametry MSE, NMSE oraz PSNR, umożliwiające porównanie skuteczności wytypowanych bibliotek. Wartości parametrów dla poszczególnych rozwiązań widoczne są w Tabeli 4.

Tabela 4: Wartości parametrów dla poszczególnych metod

Metoda	MSE	NMSE	PSNR [dB]
Accord.NET	0,000637	0,1652%	31,96
VB	0,000628	0,1628%	32,02
UCM	0,000639	0,1656%	31,95

Na podstawie wykonanej analizy, można wnioskować, że każde z badanych narzędzi może zostać z sukcesem użyte do przeprowadzenia steganografii. Najlepsze wyniki zostały uzyskane przy wykorzystaniu implementacji dr Vinayaka Bharadi. Jednakże wartości parametrów dla wszystkich trzech metod są do siebie zbliżone i optymalne.

4.5 Transformacja falkowa

Transformacja falkowa jest to przekształcenie, podobnie jak w przypadku transformacji Fouriera, oparte na operacji iloczynu skalarnego badanego sygnału. Istnieje wiele sposobów przeprowadzenia transformacji, uzyskanych w wyniku doboru różnych falek. Falką nazywamy jądro przekształcenia transformacji, czyli funkcję spełniającą wymagania analizy czasowo-częstotliwościowej. Funkcja ta jest natury sinusoidalnej [11].

Do przeprowadzenia analizy transformacji falkowej wytypowano trzy rodzaje falek: falka Haara, falka CDF97 oraz falka Daubechies rzędu 4. Falka Haara to najstarsza oraz najprostsza znana falka. Jej autorem jest Alfréd Haar, a utworzona została na początku XX wieku.

Następną wykorzystaną falką jest CDF97 (Cohen-Daubechies-Feauveau 9/7). Należy ona do rodziny falek biortogonalnych [2]. Jest ona wykorzystywana m. in. w standardzie kompresji JPEG 2000 do przeprowadzenia kompresji stratnej. Ostatnią wytypowaną falką jest falka Daubechies rzędu 4, należąca do rodziny falek Daubechies. Rodzina ta swą nazwę wywodzi od nazwiska francuskiej uczonej Ingrid Daubechies. Istnieją falki różnych rzędów, jednak dla każdej z nich otrzymane wyniki oscylowały wokół zbliżonych wartości, dlatego zdecydowano się na szczegółowe opisanie wyłącznie falki rzędu 4-go.

Do przeprowadzenia transformacji falkowej wybrano trzy biblioteki: UMapx, Accord.NET oraz NWaves. UMapx jest to biblioteka udostępniająca funkcje do

procesowania sygnałów cyfrowych. Ze wszystkich trzech rozwiązań, to ona zawiera największą liczbę obsługiwanych falek. Kolejne badane narzędzie – Accord.NET – zawiera interfejs wspierający jedynie transformację z użyciem falki Haara oraz CDF97. Biblioteka NWaves służy głównie do pracy na 1-wymiarowych sygnałach dźwiękowych, jednakże umożliwia również przeprowadzenie transformacji z użyciem falek Haara, Daubechies, Coiflet oraz Symlet.

Aby dokonać obiektywnej oceny, obliczone zostały parametry MSE, NMSE oraz PSNR, tak aby wyznaczyć optymalne rozwiązania dla konkretnych rodzajów falek. Wartości parametrów uzyskane z użyciem falki Haara przedstawiono w tabeli 5, falki CDF97 w Tabeli 6, natomiast wartości dla falki Daubechies 4-go rzędu w Tabeli 7. Dla wszystkich metod wartości parametrów wykazały wystarczająco niski stopień zniekształceń.

Tabela 5: Wartości parametrów (Haar)

Metoda	MSE	NMSE	PSNR [dB]
UMapx	0,000156	0,0404%	38,08
Accord.NET	0,000146	0,0379%	38,36
NWaves	0,000155	0,0402%	38,098

Tabela 6: Wartości parametrów (CDF97)

Metoda	MSE	NMSE	PSNR [dB]
UMapx	0,000155	0,0402%	38,101
Accord.NET	0,000151	0,0393%	38,198

Tabela 7: Wartości parametrów (Daubechies 4-go rzędu)

Metoda	MSE	NMSE	PSNR [dB]
UMapx	0,000156	0,0404%	38,081
NWaves	0,000155	0,0404%	38,086

Na podstawie analizy uzyskanych danych stwierdzono, iż możliwe jest zrealizowanie steganografii z powodzeniem dla każdego z testowanych narzędzi przy użyciu każdej z wybranych falek. Najlepsze wyniki uzyskano dla biblioteki Accord.NET, jednak wadą tego rozwiązania jest to, że wspiera ona jedynie dwie falki. Najgorszy rezultaty uzyskano przy zastosowaniu biblioteki UMapx, jednakże są to wyniki zbliżone do tych otrzymanych przy użyciu NWaves. Warto zaznaczyć, że na korzyść UMapx działa fakt, iż narzędzie to wspiera wiele rodzajów falek.

5. Metoda LSB

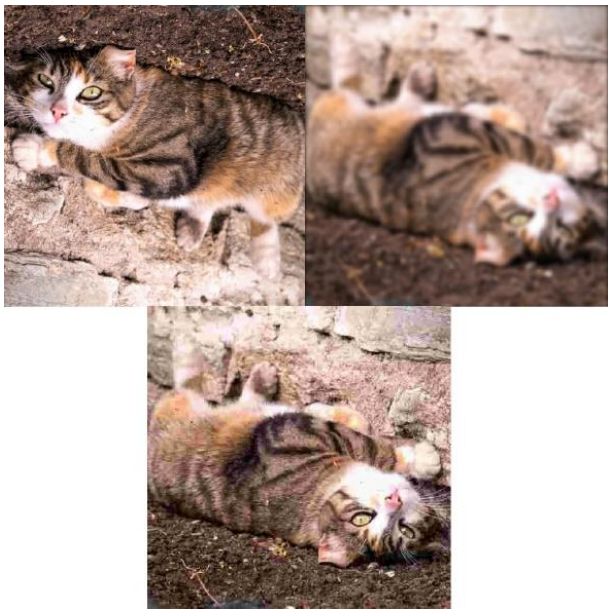
Z uwagi na łatwość realizacji steganografii z wykorzystaniem operacji bitowych zdecydowano się na implementację własnej metody mającej na celu ukrywanie tajnych informacji w obrazie. Zaimplementowany algorytm pozwala na określenie liczby najmniej znaczących bitów wykorzystanych do przechowania ukrywanej informacji. Im większa jest ich liczba, tym większa jest pojemność informacyjna, jednak wraz z jej wzrostem zmniejsza się niewykrywalność faktu przeprowadzania steganografii. Na Rysunku 1 przedstawiono przykładowe grafiki wykorzystane w procesie ukrywania danych. Grafika z kotem posłużyła za nośnik, natomiast obraz przedstawiający chmurę stanowił ukrywaną informację.

Rysunek 1: Obraz nośnika i obraz ukrywany

Przeprowadzono steganografię z użyciem różnych rozmiarów ukrywanej grafiki, a także za każdym razem iterowano liczbę podmienianych bitów danych nośnika. W każdym z procesów wykorzystano wszystkie trzy kanały kolorów: czerwony, zielony i niebieski. Im większa jest liczba zmanipulowanych bitów, a co za tym idzie również pojemność informacyjna, tym większe jest wygenerowane zniekształcenie. Obrazy wynikowe przedstawiono na Rysunku 2. Zniekształcenia występują w górnej części obrazów. Wynika to z logiki zaimplementowanej metody steganograficznej, która iteruje po kolejnych wierszach pikseli. Wielkość zniekształconego fragmentu obrazu zależna jest od liczby przekłamanych bitów, która jest pochodną rozmiaru tajnych danych.

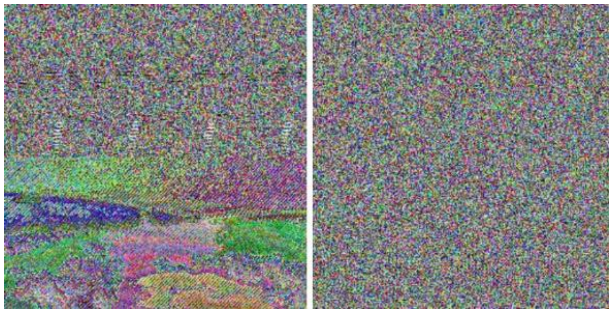
Rysunek 2: Obrazy wynikowe przy podmianie 5 oraz 7 bitów danych nośnika

Po otrzymaniu obrazów zawierających ukryte informacje przeprowadzono badanie odporności wykorzystanej metody na zniszczenie ukrywanych danych. Testowe zniekształcenia przeprowadzono na obrazie, w którym ukryty został obraz o rozmiarze 256x256. Dane zostały ukryte we wszystkich trzech kanałach kolorów w trzech najmniej znaczących bitach (Rys. 3). Wykonane zniekształcenia to: obrót, rozmycie oraz kompresja JPG. Rozmycie zastosowano w 5 różnych poziomach, a kompresję JPG w 10 stopniach jakości (od 100% do 10%). Zdecydowano się na zastosowanie takich zniekształceń i ich intensywności, które występują najczęściej podczas przekazywania i przetwarzania grafik cyfrowych.



Rysunek 3: Zniekształcone obrazy steganograficzne

Następnie podjęto się próby odzyskania ukrytych obrazów. Niektóre z otrzymanych rezultatów zostały przedstawione na rysunku 4. Jak można było oczekiwać, ukryte obrazy zostały niemal całkowicie zniszczone, co oznacza, iż metoda LSB nie jest odporna na zniekształcanie obrazu przechowującego ukryte informacje.

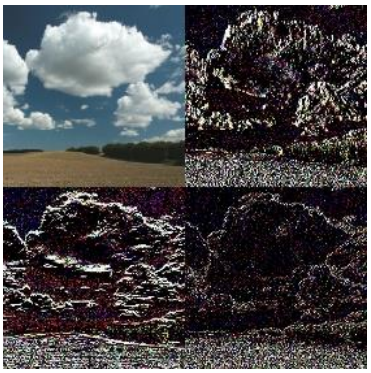


Rysunek 4: Odzyskane obrazy

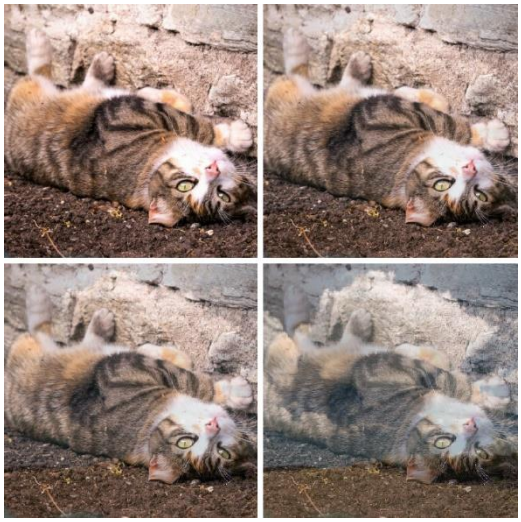
6. Ukrywanie danych przy wykorzystaniu transformacji falkowej

Z uwagi na satysfakcjonująco niski stopień zniekształceń wprowadzanych do grafiki źródłowej przy wykorzystaniu transformacji falkowej zdecydowano się na własną implementację metody bazującej na rozwiązaniu zamieszczonym w artykule „A Discrete Wavelet Transform: A Steganographic Method for Transmitting Images" [17]. Metoda ta bazuje na falce Haara. Tajna wiadomość ukrywana jest po przeprowadzeniu transformaty, której wynikiem jest grafika złożona z czterech obszarów (Rysunek 5), które kolejno reprezentują: LL (reprezentacja transformowanego obrazu po uproszczeniu), LH (ekspozycja krawędzi pionowych), HL (ekspozycja krawędzi poziomych), HH (ekspozycja krawędzi diagonalnych). Sekretne dane umieszczane są w obszarze LL.

Przeprowadzono ukrywanie danych z użyciem różnych rozmiarów tajnej grafiki. W każdym z procesów wykorzystano wszystkie trzy kanały kolorów: czerwony, zielony i niebieski. Ukrywana była grafika o rozmiarze 256x256. Zastosowana została falka Haara pierwszego rzędu. Wybrane grafiki zawierające ukryte dane zostały przedstawione na Rysunku 6. Na pierwszych dwóch grafikach ciężko jest odnaleźć różnice pomiędzy otrzymanymi grafikami, a obrazem źródłowym. Od trzeciej grafiki postępuje coraz bardziej widoczne zniekształcenie obrazu źródłowego. Im wyższa jest wartość współczynnika zagnieżdżenia, tym większy jest stopień zniekształceń wprowadzonych do grafiki wynikowej.

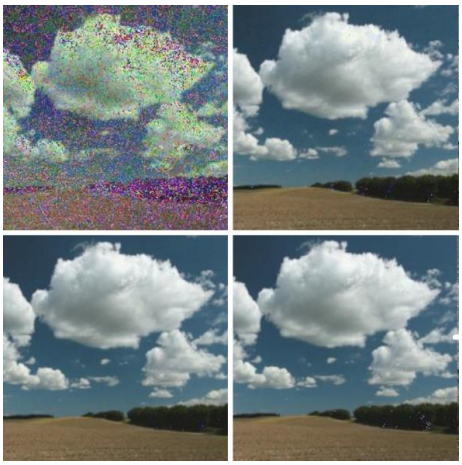


Rysunek 5: Transformata uzyskana z użyciem falki Haara



Rysunek 6: Obrazy wynikowe

Na Rysunku 7 widoczne są ukryte obrazy odzyskane z grafiki będącej nośnikiem tajnej informacji. Obrazy zachowały się najlepiej w przypadku wykorzystania współczynników zagnieżdżenia o wartościach 0,15 i 0,25. W przypadku wartości 0,02 widoczne jest znaczne uszkodzenie ukrywanej grafiki, jednakże sama jej struktura została zachowana. W momencie wykorzystania zagnieżdżenia na poziomie 0,5 zaczynają pojawiać się dodatkowe zniekształcenia ukrywanego obrazu. Jest to skutkiem samej logiki wykorzystywanej metody. Optymalne zatem jest korzystanie z zagnieżdżenia z zakresu od 0,15 do 0,25. Można zatem stwierdzić, iż proces ukrywania danych i ich ponownego odzyskania zakończył się powodzeniem dla współczynników o wartościach zagnieżdżenia 0,15 i 0,25.



Rysunek 7: Odzyskane obrazy dla współczynników zagnieżdżenia o wartościach 0,02, 0,15, 0,25 i 0,5

Aby dokonać obiektywnej oceny zniekształceń obrazów przechowujących ukrywaną informację względem obrazu źródłowego wyliczone zostały parametry: MSE, NMSE oraz PSNR. Otrzymane wyniki zostały przedstawione w Tabeli 8 i 9.

Tabela 8: Wartości parametrów dla różnych wartości współczynnika zagnieżdżenia

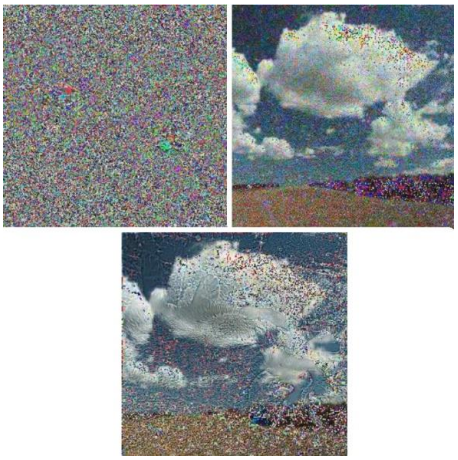
Współczynnik zagnieżdżenia	Obraz z ukrywaną informacją		
	MSE	NMSE	PSNR [dB]
0,02	0,000116	0,0331%	39,34
0,15	0,002217	0,6311%	26,54
0,25	0,005997	1,7066%	22,22
0,5	0,02362	6,722%	16,27

Tabela 9: Wartości parametrów dla różnych wartości współczynnika zagnieżdżenia

Współczynnik zagnieżdżenia	Odzyskany obraz		
	MSE	NMSE	PSNR [dB]
0,02	0,031519	11,7512%	15,01
0,15	0,000248	0,0926%	36,05
0,25	0,000093	0,0348%	40,3
0,5	0,000144	0,0536%	38,43

Dane wskazują na ujemną korelację pomiędzy jakością obrazu zawierającego ukryte dane, a jakością odzyskanego obrazu. Im większy stopień zniekształceń występuje w obrazie z ukrywaną informacją, tym mniej uszkodzony jest przekazywany, tajny komunikat. Analogicznie występuje to w drugą stronę. Wartości obliczonych parametrów potwierdzają przypuszczenia wynikające z naocznej analizy otrzymanych grafik.

Po otrzymaniu obrazów zawierających ukryte informacje przeprowadzono badanie odporności wykorzystanej metody na zniszczenie ukrywanych danych. Testowe zniekształcenia przeprowadzono na obrazie, w którym ukryty został obraz o rozmiarze 256x256. Kryterium doboru zniekształceń było identyczne jak w przypadku testowania metody LSB. Otrzymane rezultaty pozwolą na dokonanie oceny odporności badanej metody na zniszczenie ukrytych danych.



Rysunek 8: Odzyskane obrazy po obróceniu, kompresji JPG i rozmyciu nośnika

Następnie podjęto się próby odzyskania ukrytych obrazów. W przypadku obróconego obrazu ukryte dane zostały całkowicie uszkodzone. Dane można jednak odzyskać poprzez obrócenie obrazu do pierwotnej pozycji.

Kompresja JPG nośnika nie spowodowała zniekształcenia struktury ukrywanej grafiki. Im wyższa jakość kompresji, tym mniejsze zniekształcenie i uszkodzenie grafiki. Można postawić wniosek, że metoda ukrywania danych oparta na transformacji falkowej z wykorzystaniem falki Haara jest odporna na kompresję JPG.

Dla grafik odzyskanych po kompresji JPG obliczone zostały wartości parametrów MSE, NMSE oraz PSNR. Otrzymane rezultaty zostały przedstawione w Tabeli 10. Wraz ze wzrostem jakości kompresji spadają wartości parametrów MSE oraz NSME. Rośnie natomiast wartość parametru PSNR. Oznacza to, że im lepsza jest jakość kompresji, tym mniejsze jest zniekształcenie wprowadzone do grafiki wynikowej.

Tabela 10: Wartości parametrów dla różnych jakości kompresji JPG

Jakość	MSE	NMSE	PSNR [dB]
10%	0,061709	23,0079%	12,1
20%	0,036504	13,6098%	14,38
30%	0,025218	9,4021%	15,98
40%	0,019088	7,1166%	17,19
50%	0,015355	5,7246%	18,14
60%	0,011885	4,431%	19,25
70%	0,008629	3,2172%	20,64
80%	0,005521	2,0585%	22,58
90%	0,002567	0,9572%	25,91
100%	0,000591	0,2202%	32,29

Ostatnim przeprowadzonym zniekształceniem było rozmycie. Im wyższy jest poziom rozmycia, tym wyższy jest stopień zniekształceń ukrywanego obrazu. W

przypadku rozmycia 1-go poziomu odzyskany obraz zachował jedynie swoją pierwotną charakterystykę.

Aby otrzymać obiektywne dane obliczone zostały wartości parametrów MSE, NMSE oraz PSNR. Otrzymane rezultaty przedstawione zostały w Tabeli 11. Widoczne jest, że wraz ze wzrostem poziomu rozmycia wzrastają wartości parametrów MSE i NMSE, zaś wartości parametru PSNR maleją. Oznacza to postępujący wzrost wprowadzanych zniekształceń.

Tabela 11: Wartości parametrów dla różnych poziomów rozmycia

Poziom rozmycia	MSE	NMSE	PSNR [dB]
1	0,03918	14,6075%	14,07
2	0,072758	27,1263%	11,38
3	0,084515	31,5093%	10,73
4	0,091654	34,1709%	10,38
5	0,092999	34,6727%	10,32

7. Wnioski

Założeniem badania była analiza możliwości realizacji steganografii przy wykorzystaniu języka programowania C#. Dołożono szczególnych starań, aby badane narzędzia, biblioteki, a także samodzielnie zaimplementowany kod spełniały oczekiwania osoby chcącej zataić fakt przekazywania tajnych informacji. Wybrano popularne i skuteczne przekształcenia matematyczne, a następnie wyłoniono te, które w wyniku przekształcania obrazu źródłowego generują najmniejszy stopień zniekształceń.

Przeprowadzone zostało demonstracyjne ukrycie danych z wykorzystaniem samodzielnie zaimplementowanych narzędzi. Cały proces zakończył się sukcesem. Autorom udało się wykorzystać grafikę cyfrową jako nośnik tajnych informacji, umieścić w nim drugą, tajną grafikę, a na koniec odzyskać z powrotem ukryte dane z grafiki wynikowej. Wykorzystane metody steganograficzne zostały poddane testom odporności na zniekształcenia i niewykrywalności manipulacji obrazem pierwotnym.

Na podstawie przeprowadzonych działań można stwierdzić, że z powodzeniem możliwe jest zrealizowanie procesów steganograficznych przy wykorzystaniu języka programowania C#. Ponadto istnieje wiele gotowych bibliotek i narzędzi, których skuteczność została zweryfikowana w niniejszej pracy, które upraszczają proces ukrywania danych.

Z racji sprzeczności wymagań steganograficznych nie jest możliwe optymalne spełnienie ich wszystkich tj.: niewykrywalności, odporności na zniszczenie i pojemności informacyjnej. Z tego powodu nie jest możliwe jednoznaczne wskazanie najlepszych rozwiązań. Aby osiągnąć zadowalające rezultaty należy szukać kompromisów pomiędzy stawianymi wymaganiami, a także dopasowywać odpowiednie narzędzie do zaistniałego problemu.

Jako że steganografia dynamicznie się rozwija, a większość znanych metod bazuje na popularnych przekształceniach, ich implementacja w C# jest w pełni możliwa. Pozwala to wnioskować, iż język C# jest narzędziem wystarczającym do kompleksowej realizacji steganografii.

Literatura

[1] N. Ahmed, T. Natarajan, K.R. Rao, Discrete Cosine Transform, IEEE Transactions on Computers, Volume: C-23, 1 (1974) 90-93.

[2] J. Białasiewicz, Falki i aproksymacje, Wydawnictwa Naukowo-Techniczne, Warszawa, 2000.

[3] Biblioteka Accord.NET, github.com/accord-net/framework, [1.04.2021].

[4] Biblioteka AForge.NET, github.com/andrewkirillov/AForge.NET, [1.04.2021].

[5] Biblioteka Math.NET, github.com/mathnet/mathnet-numerics, [1.04.2021].

[6] Biblioteka NWaves, github.com/ar1st0crat/NWaves, [1.04.2021].

[7] Biblioteka UMapx, https://github.com/asiryan/UMapx, [1.04.2021].

[8] Biblioteka Universal.Common.Mathematics, nuget.org/packages/Universal.Common.Mathematics/, [1.04.2021].

[9] Biblioteka TrentTobler.Algorithms.FourierTransform, github.com/trenttobler/FourierTransform, [1.04.2021].

[10] S. Dhawan, R. Gupta, Analysis of various data security techniques of steganography: A survey, ISJ: A Global Perspective, 30(2) (2021) 63-87.

[11] Z. Fortuna, B. Macukow, J. Wąsowski, Metody numeryczne, Wydawnictwa Naukowo-Techniczne, Warszawa, 2015.

[12] N. Hamid, A. Yahya, R. B. Ahmad, O. M. Al-Qershi, Image steganography techniques: an overview, IJCSS, 6(3) (2012) 168-187.

[13] P. Kopniak, Metody cyfrowego przetwarzania sygnałów na potrzeby steganologii komputerowej, Politechnika Lubelska, Lublin, 2007.

[14] T. C. Lu, T. N. Vo, Reversible steganography techniques: A survey, In Digital Media Steganography, Elsevier (2021) 189-213.

[15] T. Morkel, J. H. Eloff, M. S. Olivier, An overview of image steganography, ISSA (2015).

[16] P. Strumiłło, M. Strzelecki, Przekształcenie Fouriera obrazów, Politechnika Łódzka, Łódź, 2006.

[17] M. A. Wakure, A. N. Holambe, A Discrete Wavelet Transform: A Steganographic Method for Transmitting Images, IJCA, 129 (2015) 26-29.

[18] Z. Yuan, D. Liu, X. Zhang, Q. Su, New image blind watermarking method based on two-dimensional discrete cosine transform, Optik, 204 (2020) 164152.

[19] P. Zimnicki, G. Kozieł, Analiza właściwości metod steganografii odwracalnej, JCSI, 8 (2018) 292-297.