ISSN 0555-2656

БИОНИКА ИНТЕЛЛЕКТА

ИНФОРМАЦИЯ, ЯЗЫК, ИНТЕЛЛЕКТ

№ 1 (96) **2021**

НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Основан в октябре 1967 г.

Учредитель и издатель
Харьковский национальный университет радиоэлектроники

Периодичность издания – *2 раза в год*

Харьков • ХНУРЭ • 2021

Научно-технический журнал

«БИОНИКА ИНТЕЛЛЕКТА»

ISSN 0555-2656

Основан Харьковским национальным университетом
радиоэлектроники в 1967 году

Реферирование и индексирование:

Google Scholar

BASE

INDEX COPERNICUS
INTERNATIONAL

Національна бібліотека України
імені В. І. Вернадського

Журнал включен в список научных специализированных изданий Украины
по техническим и физико-математическим наукам
согласно приказа Министерства образования и науки Украины № 820 от 11.07.2016

UDC 519.246.8+519.216.3+550.385.1 DOi 10.30837/ bi.2021.1(96).01

Vadim Romanuke
Polish Naval Academy, Gdynia, Poland
v.romanuke@amw.gdynia.pl

MAXIMUM-VERSUS-MEAN ABSOLUTE ERROR IN SELECTING CRITERIA OF TIME SERIES FORECASTING QUALITY

In time series forecasting, a commonly accepted criterion of the forecasting quality is the root-mean-square error (RMSE). Sometimes only RMSE is used. In other cases, another measure of forecasting accuracy is used along with RMSE. It is the mean absolute error (MAE). Although RMSE and MAE are the common criteria of time series forecasting quality, they both register information about averaged errors. However, averaging may remove information about volatility, which is typical for time series, in a few points (outliers) or narrow intervals. Information about outliers in time series forecasts (with respect to test data) can be registered by the maximum absolute error (MaxAE). The MaxAE criterion does not have any relation to averaging. It registers information about the worst outlier instead. Therefore, the goal is to ascertain the best criteria of time series forecasting quality, wherein the RMSE criterion is always present. First, 12 types of benchmark time series are defined to test and select criteria. The time series is of 168 points, whereas the last third of the series is forecasted. After having generated 200 times series for each of those 12 types, ARIMA forecasts are made at 56 points of every series. All the 2400 RMSEs are sorted in ascending order, whereupon the respective MAEs and MaxAEs are re-arranged as well. The interrelation between the RMSE and MAE/MaxAE is studied by their intercorrelation function. RMSEs and MaxAEs are "more different" than RMSEs and MAEs, because the correlation between the RMSE and MAE is stronger. Consequently, the MAE criterion is useless as it just nearly replicates information about the forecasting quality from the RMSE criterion. Inasmuch as the MaxAE criterion can import additional information about the forecasting quality, the best criteria are RMSE and MaxAE.

TIME SERIES FORECASTING, FORECASTING QUALITY, ROOT-MEAN-SQUARE ERROR, MEAN ABSOLUTE ERROR, MAXIMUM ABSOLUTE ERROR, OUTLIERS, ARIMA FORECASTING, INTERCORRELATION FUNCTION

Романюк В. В. Максимальна проти середньої абсолютної похибки при виборі критеріїв якості прогнозування часових рядів. При прогнозуванні часових рядів якість прогнозів загальноприйнято оцінювати за критерієм середньоквадратичної помилки (RMSE). Іноді тільки RMSE й використовують. В інших випадках, разом з RMSE використовується ще одна міра точності прогнозування. Такою мірою є середня абсолютна помилка (MAE). Хоча RMSE й MAE є загальноприйнятими критеріями якості прогнозування часового ряду, вони обидва фіксують інформацію про усереднені помилки. Однак усереднення може стирати інформацію про волатильність, яка є типовою для часових рядів у точках викидів або на вузьких інтервалах. Інформація про викиди у прогнозах часового ряду (відносно тестових даних) може бути зафіксована за допомогою максимуму абсолютної помилки (MaxAE). MaxAE-критерій не має жодного стосунку до усереднення. Натомість він фіксує інформацію про найгірший викид. Тому мета полягає у встановленні найкращих критеріїв якості прогнозування часових рядів, де, щоправда, RMSE-критерій завжди присутній. Спочатку визначаються 12 типів контрольних часових рядів для тестування і вибору критеріїв. Часовий ряд складається зі 168 точок, причому прогнозується остання третина цього ряду. Згенерувавши 200 часових рядів для кожного з 12 типів, виконуються ARIMA-прогнози у 56 точках кожного ряду. Всі 2400 значень RMSE сортуються у порядку зростання, після чого відповідні значення MAE й MaxAE також упорядковуються наново. Взаємоспіввідношення між RMSE й MAE/MaxAE вивчається за їх взаємокореляційною функцією. Значення RMSE й MaxAE є "більш різними", ніж значення RMSE й MAE, оскільки кореляція між RMSE й MAE є сильнішою. Отже, MAE-критерій не має сенсу, тому що він практично повторює інформацію про якість прогнозування з RMSE-критерію. Оскільки MaxAE-критерій може вносити додаткову інформацію про якість прогнозування, найкращими критеріями є RMSE й MaxAE.

ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ, ЯКІСТЬ ПРОГНОЗУВАННЯ, СЕРЕДНЬОКВАДРАТИЧНА ПОМИЛКА, СЕРЕДНЯ АБСОЛЮТНА ПОМИЛКА, МАКСИМУМ АБСОЛЮТНОЇ ПОМИЛКИ, ВИКИДИ, ARIMA-ПРОГНОЗУВАННЯ, ВЗАЄМОКОРЕЛЯЦІЙНА ФУНКЦІЯ

Романюк В. В. Максимальная против средней абсолютной ошибки при выборе критериев качества прогнозирования временных рядов. При прогнозировании временных рядов качество прогнозов общепринято оценивать по критерию среднеквадратической ошибки (RMSE). Иногда только RMSE и используют. В других случаях, вместе с RMSE используется ещё одна мера точности прогнозирования. Такой мерой является средняя абсолютная ошибка (MAE). Хотя RMSE и MAE являются общепринятыми критериями качества прогнозирования временного ряда, они оба фиксируют информацию об усреднённых ошибках. Однако усреднение может стирать информацию о волатильности, которая типична для временных рядов в точках выбросов или на узких интервалах. Информация о выбросах в прогнозах временного ряда (относительно тестовых данных) может быть зафиксирована при помощи максимума абсолютной ошибки (MaxAE). MaxAE-критерий не имеет никакого отношения к усреднению. Вместо этого он фиксирует информацию о наихудшем выбросе. Поэтому цель состоит в установлении наилучших критериев качества прогнозирования временных рядов, где, впрочем, RMSE-критерий присутствует всегда. Сначала определяются 12 типов контрольных временных рядов для тестирования и выбора критериев. Временной ряд состоит из 168 точек, причём прогнозируется последняя треть этого ряда.

Сгенерировав 200 временных рядов для каждого из 12 типов, осуществляются ARIMA-прогнозы в 56 точках каждого ряда. Все 2400 значений RMSE сортируются в порядке возрастания, после чего соответствующие значения MAE и MaxAE также упорядочиваются заново. Взаимосоотношение между RMSE и MAE/MaxAE изучается по их взаимокореляционной функции. Значения RMSE и MaxAE являются "более разными", чем значения RMSE и MAE, поскольку корреляция между RMSE и MAE сильнее. Итак, MAE-критерий бесполезен, потому что он практически дублирует информацию о качестве прогнозирования из RMSE-критерия. Поскольку MaxAE-критерий может привносить дополнительную информацию о качестве прогнозирования, наилучшими критериями являются RMSE и MaxAE.

ПРОГНОЗИРОВАНИЕ ВРЕМЕННЫХ РЯДОВ, КАЧЕСТВО ПРОГНОЗИРОВАНИЯ, СРЕДНЕКВАДРАТИЧЕСКАЯ ОШИБКА, СРЕДНЯЯ АБСОЛЮТНАЯ ОШИБКА, МАКСИМУМ АБСОЛЮТНОЙ ОШИБКИ, ВЫБРОСЫ, ARIMA-ПРОГНОЗИРОВАНИЕ, ВЗАИМОКОРЕЛЯЦИОННАЯ ФУНКЦИЯ

1. Common criteria of time series forecasting quality

Time series forecasting is applied to control and predict processes comprising sequences of data. The quality (accuracy) of forecasting depends on which approaches are used to forecast, length of forecast, and which criteria are used to estimate the quality. A commonly accepted criterion is the root-mean-square error (RMSE) [1, 2]. Sometimes only RMSE is used. In other cases, another measure of forecasting accuracy is used along with RMSE. It is the mean absolute error (MAE) [3]. Although RMSE and MAE are the common criteria of time series forecasting quality [4], they both register information about averaged errors. Meanwhile, time series forecasts are never guaranteed to be similarly scattered around test data points (with respect to which the forecasting accuracy is estimated). Volatility is typical for times series, and it becomes more intense for farther points forecasted. However, averaging may remove information about volatility in a few points (spikes) or narrow intervals.

Information about spikes (which also may be referred to as outliers) in time series forecasts (with respect to test data) can be registered by the maximum absolute error (MaxAE). The MaxAE criterion does not have any relation to averaging [5]. An open question is whether it is better to estimate forecasting accuracy by using only RMSE, or the pair of RMSE and MAE, or the pair of RMSE and MaxAE.

2. The goal and tasks to achieve it

The goal is to ascertain the best criteria of time series forecasting quality (accuracy). To achieve the goal, the following three tasks are to be completed. First, benchmark time series will be defined to test and select criteria. The four options are the single RMSE criterion, RMSE and MAE, RMSE and MaxAE, or RMSE by MAE and MaxAE. Second, an analysis of MaxAE versus MAE will be carried out after forecasts are made by the ARIMA approach [1, 2, 6, 7]. Finally, the selection of criteria should be justified followed by an appropriate conclusion.

3. MaxAE criterion

Consider a time series defined on a sequence of time points $t=\overline{1,T}$, where T is an available amount of data (not to be confused with the availability of data in real-world practice, where forecasting is "blind" and the factual accuracy of forecasts is principally indeterminable until time point T is reached). Hereinafter, it is easier to presume that, without losing generality, $t_i = i$. Let $\left\{y(t_i)\right\}_{i=1}^{T_0}$ by $T_0 < T$ be the data by which forecasts are made at $t_i = \overline{T_0+1, T}$. So, set $\left\{y(t_i)\right\}_{i=1}^{T_0}$ is the time series to be forecasted for $T - T_0$ time points ahead.

Data $\left\{y(t_i)\right\}_{i=T_0+1}^{T}$ are used for testing the forecasting accuracy. These data are normalized (standardized to the range from 0 to 1) as follows [8, 9]:

$$u(t_i) = \frac{y(t_i) - \min\limits_{k=\overline{T_0+1, T}} y(t_k)}{\max\limits_{k=\overline{T_0+1, T}} y(t_k) - \min\limits_{k=\overline{T_0+1, T}} y(t_k)}. \quad (1)$$

The standardization by (1) allows comparing the forecasting quality for different time series defined along the same number of points to be forecasted. Similarly to (1), if $\left\{\tilde{y}(t_i)\right\}_{i=T_0+1}^{T}$ are forecasted data (regardless of approaches used to forecast), they are normalized also (with respect to the initial data):

$$\tilde{u}(t_i) = \frac{\tilde{y}(t_i) - \min\limits_{k=\overline{T_0+1, T}} y(t_k)}{\max\limits_{k=\overline{T_0+1, T}} y(t_k) - \min\limits_{k=\overline{T_0+1, T}} y(t_k)}. \quad (2)$$

Then the RMSE is [1, 2]

$$\rho_{RMSE} = \sqrt{\frac{1}{T-T_0} \sum_{i=T_0+1}^{T} \left[u(t_i) - \tilde{u}(t_i)\right]^2} \quad (3)$$

and the MAE [3] is

$$\rho_{MAE} = \frac{1}{T-T_0} \sum_{i=T_0+1}^{T} \left|u(t_i) - \tilde{u}(t_i)\right|. \quad (4)$$

The difference between RMSE (3) and MAE (4) is not that much. Obviously, owing to the square, the RMSE criterion intensifies greater errors. This is why it is unconditionally used almost everywhere to compare data, functions, surfaces, etc. [5, 10]. The MAE criterion is sometimes claimed to be more suitable but reasons behind this are quite unclear. A very serious drawback of both RMSE and MAE is that they do not show outliers. This is so because outliers, if any, are lost due to averaging. On the contrary, the MaxAE calculated as

$$\rho_{MaxAE} = \max_{i=\overline{T_0+1, T}} \left|u(t_i) - \tilde{u}(t_i)\right| \quad (5)$$

registers information about the worst outlier [11, 12]. Therefore, whereas RMSE (3) is a compulsory criterion,

using MAE (4) or/and MaxAE (5) requires a thorough research and subsequent justification.

4. Benchmark time series

The benchmark time series are based on 12 random-like sequences with repeatability. Every sequence is generated by using pseudorandom numbers drawn from the standard normal distribution (with zero mean and unit variance) [8, 13, 14]. Apart from the repeating random "pure" sequence, a trend, seasonality, and extinction properties are embedded into the sequences by the following patterns [15]:

$$y_1(t)=\left[a_1+0.25\Theta_1(T)\right]r_1(t)+a_2\Theta_2(T), \quad (6)$$

$$y_2(t)=\left[a_1+0.25\Theta_3(T)\right]r_2(t)+a_2\Theta_4(T)+a_3t, \quad (7)$$

$$y_3(t)=\left[a_1+0.25\Theta_5(T)\right]r_3(t)+$$
$$+a_2\Theta_6(T)+\left[a_4+0.25\Theta_7(T)\right]a_5\cos(\upsilon t), \quad (8)$$

$$y_4(t)=\left[a_1+0.25\Theta_8(T)\right]r_4(t)+a_2\Theta_9(T)+$$
$$+a_3t+\left[a_4+0.25\Theta_{10}(T)\right]a_5\cos(\upsilon t), \quad (9)$$

$$y_5(t)=\left[a_1+0.25\Theta_{11}(T)\right]r_5(t)e^{-a_6t}+a_2\Theta_{12}(T), \quad (10)$$

$$y_6(t)=\left[a_1+0.25\Theta_{13}(T)\right]r_6(t)e^{a_6t}+a_2\Theta_{14}(T), \quad (11)$$

$$y_7(t)=\left[a_1+0.25\Theta_{15}(T)\right]r_7(t)e^{-a_6t}+$$
$$+a_2\Theta_{16}(T)+a_3t, \quad (12)$$

$$y_8(t)=\left[a_1+0.25\Theta_{17}(T)\right]r_8(t)e^{-a_6t}+$$
$$+a_2\Theta_{18}(T)+\left[a_4+0.25\Theta_{19}(T)\right]a_5\cos(\upsilon t)e^{-a_6t}, \quad (13)$$

$$y_9(t)=\left[a_1+0.25\Theta_{20}(T)\right]r_9(t)e^{-a_6t}+a_2\Theta_{21}(T)+$$
$$+a_3t+\left[a_4+0.25\Theta_{22}(T)\right]a_5\cos(\upsilon t)e^{-a_6t}, \quad (14)$$

$$y_{10}(t)=\left[a_1+0.25\Theta_{23}(T)\right]r_{10}(t)e^{a_6t}+$$
$$+a_2\Theta_{24}(T)+a_3t, \quad (15)$$

$$y_{11}(t)=\left[a_1+0.25\Theta_{25}(T)\right]r_{11}(t)e^{a_6t}+$$
$$+a_2\Theta_{26}(T)+\left[a_4+0.25\Theta_{27}(T)\right]a_5\cos(\upsilon t)e^{a_6t}, \quad (16)$$

$$y_{12}(t)=\left[a_1+0.25\Theta_{28}(T)\right]r_{12}(t)e^{a_6t}+a_2\Theta_{29}(T)+$$
$$+a_3t+\left[a_4+0.25\Theta_{30}(T)\right]a_5\cos(\upsilon t)e^{a_6t}, \quad (17)$$

where $r_g(t)$ is a sequence of identical randomly-structured subsequences (whose shape is not that random and it still may have some roughly-regular convexities/concavities) of type g, $\{\Theta_l(T)\}_{l=1}^{30}$ are vectors of T pseudorandom numbers (these vectors are used to simulate noise and volatility), $\{a_h>0\}_{h=1}^{6}$ is a set of adjustable coefficients, and factor $\upsilon>0$ indicates an oscillation frequency. Initially, a time series is generated by

$$a_1=2,\ a_2=0.175,\ a_3=0.01,\ a_4=5,\ a_5=0.18,$$
$$\upsilon=0.02,\ a_6=0.0005,\ T=1680,$$

where every sequence $r_g(t)$ is of 6, 7, or 8 subsequences, $g=\overline{1,12}$. Then the time series is equidistantly downsampled so that 168 time points remain. These points are smoothed producing thus the benchmark time series. Graphical examples of benchmark time series generated by (6) — (17) are presented in Figure 1.

It is worth noting that the benchmark series are intentionally generated in a way preventing from forecasting trivial time series (being "easy-to-forecast-with-high-accuracy" time series). Although the downsampled time series is smoothed, fluctuations in it are still present (due to every initial sequence of 1680 points has severe spikes which cannot be literally smoothed). Moreover, the smoothing itself may produce outliers at the starting and ending time points, i. e. at $t_1=1$ and $t_{168}=168$. For example, an outlier is seen in the top left subplot in Figure 1. Thus, despite this subplot represents the simplest case without trend and seasonality, the forecasts are not likely to be accurate. Another outlier example is in the second row middle subplot, where the ending time point has unexpectedly dropped down. Similar cases with outliers are seen in Figure 1 as well. Such benchmarking is made to obtain more significant differences in accuracy. This subsequently will allow making more effective decisions on the best criteria of time series forecasting quality.

5. MaxAE versus MAE

After having generated 200 times series for each of those 12 types by $T=168$, ARIMA forecasts are made at $t_i=\overline{113,168}$ (i. e, the forecast length is one third of the available data). The worst and best forecasts (whose RMSEs are the highest and the least for the given type, respectively) are presented in Figure 2. All the 2400 RMSEs are sorted in ascending order, whereupon the respective MAEs and MaxAEs are re-arranged as well. They are shown in Figure 3, where 10 worst RMSEs along with the respective 10 MAEs and MaxAEs are cut off due to they correspond to unacceptable forecasts (see Figure 2).

The interrelation between the RMSE and MAE/MaxAE can be studied by their intercorrelation function [16, 17]. If $\{c_j\}_{j=1}^{n}$ and $\{d_j\}_{j=1}^{n}$ are some real-valued data, where $c_j=0$ and $d_j=0$ by $j<1$ or $j>n$, their intercorrelation function is a sequence calculated as follows:

$$b(z)=\sum_{j=1}^{n}c_j\cdot d_{j-z} \quad \text{for} \quad z=\overline{-n+1,n-1}. \quad (18)$$

Inasmuch as the RMSEs, MAEs, and MaxAEs are differently scattered from minimum to maximum values, it is better to normalize them before calculating intercorrelation functions by (18). For this, every value of the respective criterion is divided by the maximum (corresponding to the worst forecast). The intercorrelation

functions calculated in this way are presented in Figure 4. The normalized intercorrelation functions are shown in Figure 5 allowing to see more distinctly that RMSEs and MaxAEs are less correlating than RMSEs and MAEs. This is so because the intercorrelation function of RMSEs and MaxAEs is closer to the intercorrelation function of RMSEs and noise, whereas correlation with noise is always the least.

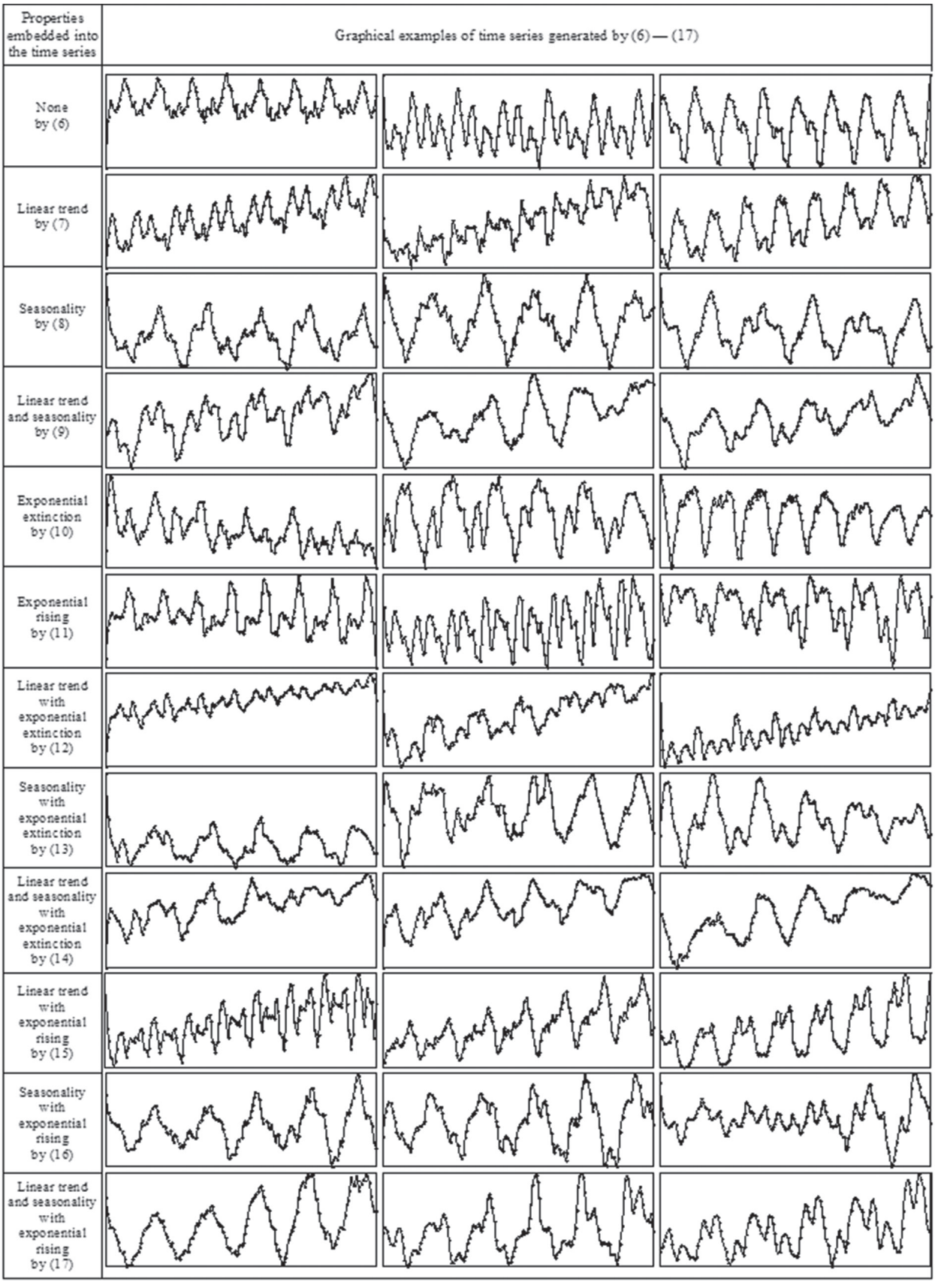


Fig. 1. Graphical examples of the 12 types of benchmark time series

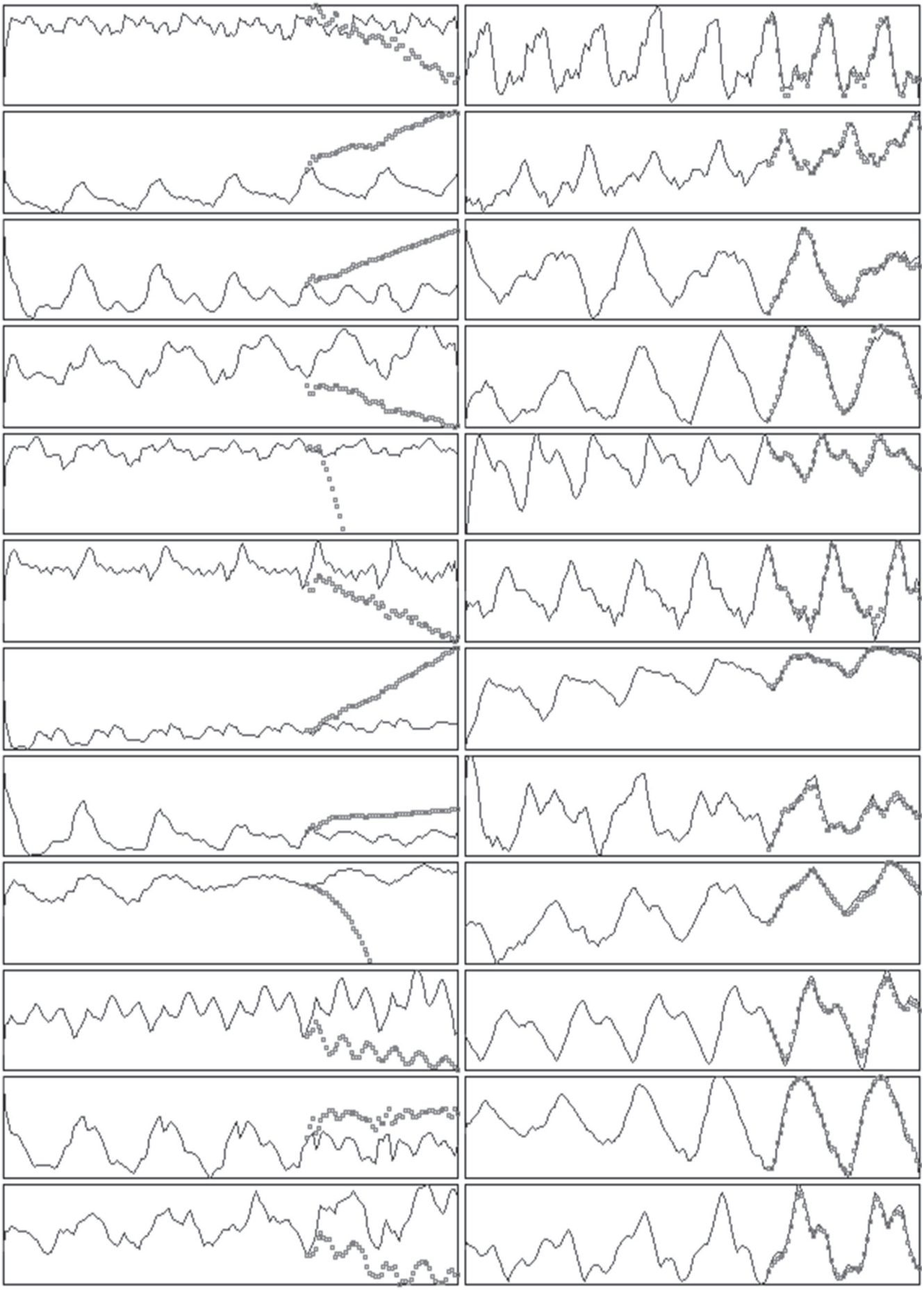


Fig. 2. The worst and best forecasts for each of the 12 types of benchmark time series

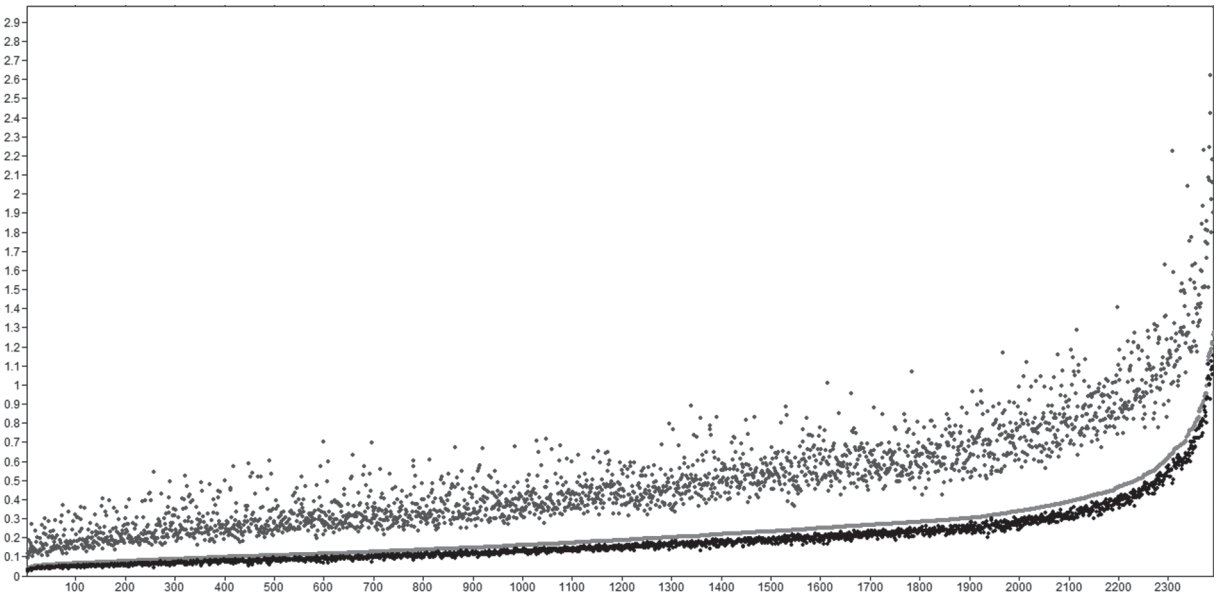


Fig. 3. The distribution of the sorted RMSEs (seen as a line in the middle) of forecasts along with the re-arranged MAEs (points below) and MaxAEs (points above) for the 2390 time series

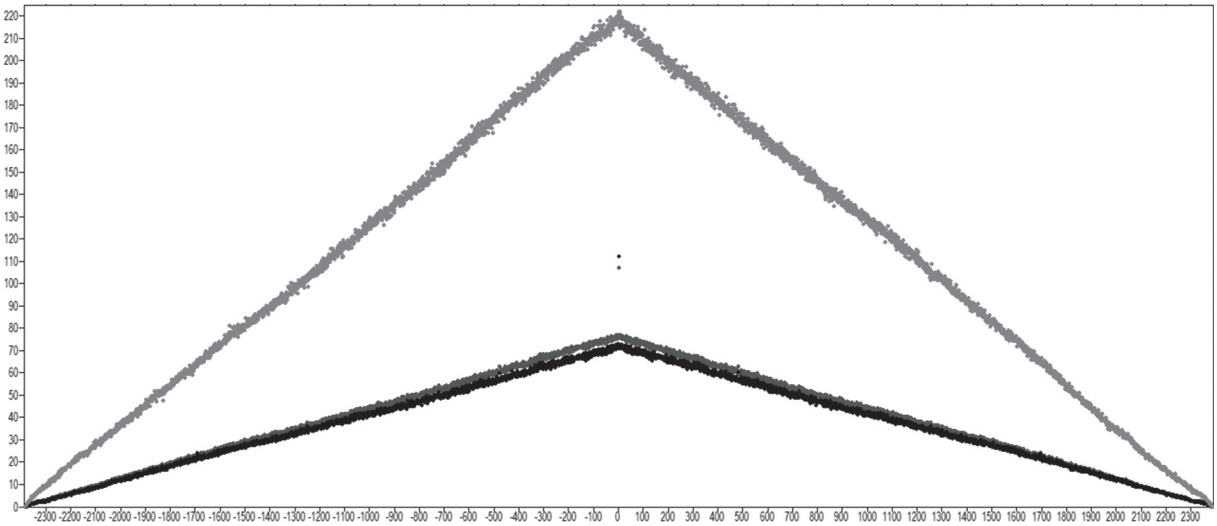


Fig. 4. The three intercorrelation functions of 2390 RMSEs and MAEs, MaxAEs, and a pseudorandom sequence of values (noise) from interval $[0;1]$

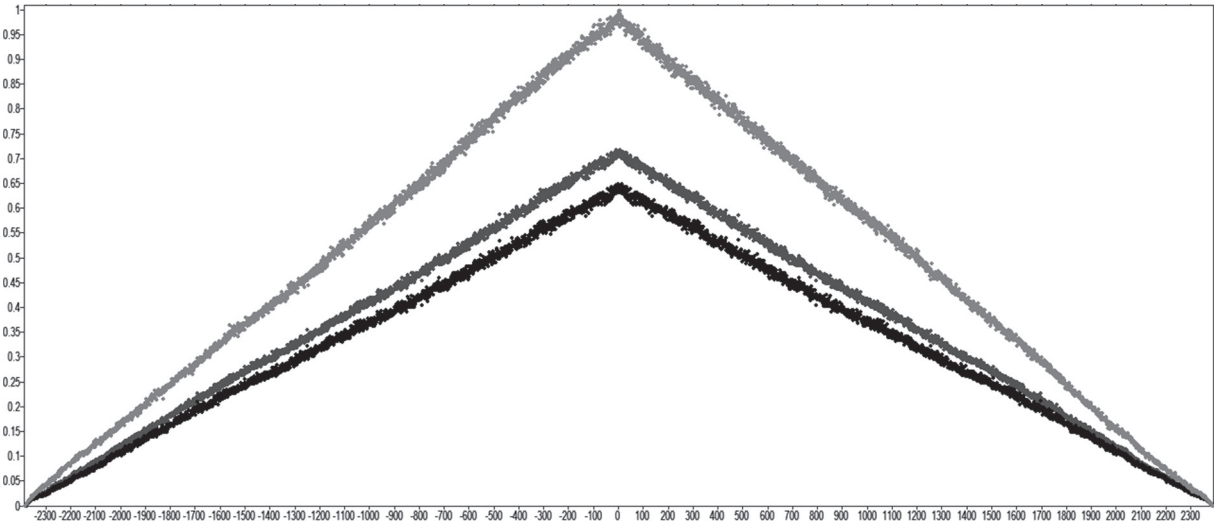


Fig. 5. The normalized intercorrelation functions from Fig. 4

In fact, Figures 3 and 5 are an experimental proof of that using the RMSE and MaxAE criteria is better than using the RMSE and MAE criteria to estimate the time series forecasting quality. When RMSEs are sorted in either ascending or descending order, MaxAEs are more scattered than MAEs (see Figure 3). RMSEs and MaxAEs are "more different" than RMSEs and MAEs (see Figure 5, although Figure 4 may serve herein also), because the correlation between the RMSE and MAE is stronger. This implies that using the MAE criterion along with the RMSE criterion is redundant, whereas the MaxAE criterion can import additional information about forecasts.

Conclusion

Based on the analysis of forecasts for 2400 time series, it is ascertained that the MAE criterion nearly replicates information about the forecasting quality, which is directly drawn from the RMSE criterion. Therefore, the MAE criterion is useless, whichever a group of criteria is, unless the MAE criterion just substitutes the RMSE criterion. Inasmuch as the MaxAE criterion can import additional information about the forecasting quality, the best criteria are RMSE and MaxAE.

Conflict of Interest

The author declares no conflict of interest.

Acknowledgment

This work was technically supported by the Faculty of Mechanical and Electrical Engineering, Polish Naval Academy, Poland

References

[1] *Schelter B., Winterhalder M., Timmer J.* Handbook of Time Series Analysis: Recent Theoretical Developments and Applications. — Wiley, 2006.

[2] *Kotu V., Deshpande B.* Data Science (Second Edition). — Morgan Kaufmann, 2019.

[3] *Hyndman R., Koehler A.* Another look at measures of forecast accuracy // International Journal of Forecasting. — 2006. — Vol. 22, Iss. 4. — P. 679 — 688.

[4] *De Gooijer J. G., Hyndman R. J.* 25 years of time series forecasting // International Journal of Forecasting. — 2006. — Vol. 22, Iss. 3. — P. 443 — 473.

[5] *Edwards R. E.* Functional Analysis. Theory and Applications. — Hold, Rinehart and Winston, 1965.

[6] *Pankratz A.* Forecasting with Univariate Box — Jenkins Models: Concepts and Cases. — John Wiley & Sons, 1983.

[7] *Box G., Jenkins G., Reinsel G.* Time Series Analysis: Forecasting and Control. — Prentice Hall, Englewood Cliffs, NJ, 1994.

[8] *Romanuke V. V.* Decision making criteria hybridization for finding optimal decisions' subset regarding changes of the decision function // Journal of Uncertain Systems. — 2018. — Vol. 12, No. 4. — P. 279 — 291.

[9] *Brockwell P. J., Davis R. A.* Introduction to Time Series and Forecasting. — Springer, Cham, 2016.

[10] *Romanuke V. V.* On optimizing WLF equation over experimental viscosity measurements by finite set of fixed temperatures with running metric // Optoelectronic Information-Power Technologies. — 2011. — No. 1 (21). — P. 24 — 43.

[11] *Romanuke V. V.* A minimax approach to mapping partial interval uncertainties into point estimates // Journal of Mathematics and Applications. — 2019. — Vol. 42. — P. 147 — 185.

[12] *Romanuke V. V.* Theoretic-game methods of identification of models for multistage technical control and run-in under multivariate uncertainties. — Mathematical Modeling and Computational Methods. — Vinnytsia National Technical University, Vinnytsia, Ukraine, 2014.

[13] *Kneusel R.* Random Numbers and Computers. — Springer International Publishing, 2018.

[14] *Romanuke V. V.* Multiple state problem reduction and decision making criteria hybridization // Research Bulletin of NTUU "Kyiv Polytechnic Institute". — 2016. — No. 2. — P. 51 — 59.

[15] *Romanuke V. V.* Time series smoothing and downsampling for improving forecasting accuracy // Applied Computer Systems. — 2021. — Vol. 26, No. 1. — P. 60 — 70.

[16] *Gubner J.* Probability and Random Processes for Electrical and Computer Engineers. — Cambridge University Press, 2006.

[17] *Romanuke V. V.* Computational method of building orthogonal binary functions bases for multichannel communication systems with code channels division. — Mathematical Modeling and Computational Methods. — Ternopil State Technical University, Ternopil, Ukraine, 2006.

The article was delivered to editorial stuff on the 19.02.2021

УДК 004.75 DOi 10.30837/ bi.2021.1(96).02

Kyrychenko I.V.[1], Kolesnyk V.V.[2], Shmelov O.B.[3]

[1]PhD, Senior Lecturer at the Department of Software Engineering, Kharkov National University of Radio Electronics, iryna.kyrychenko@nure.ua, ORCID iD: 0000-0002-7686-6439

[2]Graduate students of the Department of Software Engineering, Kharkov National University of Radio Electronics, valeriia.kolesnyk@nure.ua, ORCID iD: 0000-0001-7811-716X

[3]Graduate students of the Department of Software Engineering, Kharkov National University of Radio Electronics, oleh.shmelov@nure.ua, ORCID iD: 0000-0002-7489-6386

THE USAGE AND IMPLEMENTATION OF PARALLELISM IN GO PROGRAMMING LANGUAGE BASED ON THE MPI INTERFACE AS A MESSAGE EXCHANGE METHOD

The development of the methods for optimizing computer processes by the means of Go programming language. The resources for MPI computations were analyzed from the side of Go programming language. Proposed attempts to fabricate the ties the Go form devices hit their restriction of adaptability quick. Among the advantages of using Go programming language for implementation MPI algorithms, could be saud that it eliminates the need for the developer to manage memory and resources used by software manually, own binaries, fast and efficient compilation. Athough Golang uses several resources to create parallel computations, MPI algorithms implemented by Golang methods and techniques do not fully integrate exchange and computation. Were compared two Jacobi methods for solving partial differential equation. The results showed that Go cannot coordinate the execution of C, although Go scales a part more pleasant when using non-blocking communication when comparing the blocking C usage with the blocking Go execution and and comparing the non-blocking implementations with each other. Go programming language is used for developing massive systems that can speed up software code several times by properly converting sequential algorithms to competing ones, nevertheless MPI developers are not recommended to use it due to its complexity for implementation. As a result, there is currently almost no MPI implemented by Golang methods and techniques that would fully integrate exchange and computation.

MESSAGE PASSING INTERFACE, PARALLEL PROGRAMMING, GO PROGRAMMING LANGUAGE, JACOBI METHOD

І.В. Кириченко, В.В. Колесник, О.Б. Шмельов. Використання і реалізація паралелізму в мові програмування Go на основі інтерфейсу MPI як методу обміну повідомленнями. Розглянуто розробку методів оптимізації ком'ютерних процесів за допомогою засобів мови програмування Go. Проаналізовано ресурси обчислення Інтерфейсу передачі повідомлень, що містить мова програмування Go. Пропонується метод фабрикування зв'язків для аналізу швидкості враження обмеження пристосованості. Серед переваг використання мови програмування Go для реалізації алгоритмів MPI є факт усування необхідності розробнику керувати пам'яттю та ресурсами, що використовуються програмним забезпеченням вручну, власними двійковими файлами, швидкою та ефективною компіляцією. Хоча Golang використовує кілька ресурсів для створення паралельних обчислень, алгоритми MPI, реалізовані методами та техніками Golang, не повністю інтегрують обмін та обчислення. Було порівняно два методи Jacobi для розв'язку рівняння з частковими похідними. Результати показали, що Go не може координувати виконання C, а також, що Go масштабує частину при використанні неблокуючого зв'язку та порівнянні неблокуючих реалізацій між собою. Мова програмування Golang є конкуруючим інструментом розробки послідовні алгоритми на конкуруючі, проте розробникам MPI не рекомендується використовувати його через його складність для реалізації. Як результат, в даний час майже не існує MPI, реалізованого методами та техніками Golang, які б повністю інтегрували обмін та обчислення.

ІНТЕРФЕЙС ПЕРЕДАЧІ ПОВІДОМЛЕНЬ, ПАРАЛЕЛЬНЕ ПРОГРАМУВАННЯ, МОВА ПРОГРАМУВАННЯ GO, МЕТОД JAKOBI

И.В. Кириченко, В.В. Колесник, О.Б. Шмелев. Использование и реализация параллелизма в языке программирования Go на основе интерфейса MPI как метода обмена сообщениями. Рассмотрена разработка методов оптимизации комъютерних процессов с помощью средств языка программирования Go. Проанализированы ресурсы вычисления интерфейса передачи сообщений, которые содержит язык программирования Go. Предлагается метод фабрикации связей для анализа скорости поражения ограничения приспособленности. Среди преимуществ использования языка программирования Go для реализации алгоритмов MPI может быть выделен факт устранения необходимости разработчику управлять памятью и ресурсами, которые используются программным обеспечением вручную, собственными двоичными файлами, быстрой и эффективной компиляцией. Хотя Golang использует несколько ресурсов для создания параллельных вычислений, алгоритмы MPI, реализованные методы и техники Golang, полностью не интегрируют обмен и вычисления. Сравнены два метода Jacobi для решения уравнения с частными производными. Результаты показали, что Go не может координировать выполнение C, а также, что Go масштабирует часть при использовании неблокирующий связи при и сравнения неблокирующий реализаций между собой. Язык программирования Golang является конкурирующим инструментом разработки последовательные алгоритмы на конкурирующие, однако разработчикам MPI не рекомендуется использовать его за его сложности для реализации. Как результат, в настоящее время почти не существует MPI, реализованного методами и техниками Golang, которые полностью интегрировали обмен и вычисления.

ИНТЕРФЕЙС ПЕРЕДАЧИ СООБЩЕНИЙ, ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ, ЯЗЫК ПРОГРАММИРОВАНИЯ GO, МЕТОД JAKOBI

Introduction

With the development of personal computer processor development technologies and the increasing demand for fast calculations, the need to optimize computer computing processes is growing. To reduce the cost of software support and unit processing, software customers rightly require developers to process as much data as possible in the shortest amount of time. Given the vast amount of stakeholder data available to stakeholders, which is subject to, for example, analysis, effective sequential algorithms alone are not enough to speed up processing.

Many algorithms are based on the sequential execution of certain actions on a list of entities.

Because the actions performed on each element of the processed sequence are not likely to change regardless of the iteration number, such algorithms can be accelerated several times using multi-core processor resources, parallel programming methods and tools, and the Go (or Golang) programming language.

Go programming language, which is quite low-level and has many advantages, such as: static typing, garbage collector - a tool that eliminates the need for the developer to manage memory and resources used by software manually, own binaries, fast and efficient compilation, easy to work with multithreading, the developer can manage and interact, for example, with threads and use all the benefits of MPI.

1. Resources for parallel computations

The definition of the parallelism assumes that an application splits its its tasks into smaller sub-tasks, processing, for example, on multiple processors at the same time. It is well-known fact, that a language that supports multiple processes at the same time is ideal for creating global scalable programs.

Go uses several resources to create parallel computations. One of such specifications is Message Passing Interface (MPI). When working with Go, you need to clearly define the concept of competitiveness. Competitiveness is the ability of various elements of the program, algorithms or tasks to run erratically, or in part without affecting the initial result. This significantly affects the efficiency of the software and the speed of performing computational calculations.

The main tools for working with competition in the Golang programming language are Goroutnies - easily executable methods, or functions that are performed regardless of the calling method or function. The efficiency of Goroutines is largely due to their behavior, which is to plan the processing of a given number of system threads, which, in theory, allows you to process any number of Goroutines. It is worth noting that one of the advantages of using Gorutin in the Go is the minimization of the developer's effort to write code.

Also, one of the basic concepts for working with competitiveness in Go are channels - a way of communication between individual gorits. Channels are a conditional «backbone» for communication and synchronization of performed bursts. With the help of channels, there is no need to create intermediate conditional variables that would control and check each of the executed Goroutines for the completion of a certain stage of the method or its execution. Channels, like any container, must be precreated before use as follows: make (chan int). The given code creates a channel for transmitting integers. To process «events» created by transmitting data to channels from bursts, you can use the select operator, which blocks the execution of bursts until the operation of at least one of the described situations.

2. Golang as MPI computation and its results

Golang is a static typed language that uses a compiler. The program needs to explicitly declare types of variables, so even trivial errors are easily detected. Although an interpreter is available for Go, there is not much need for one, as the compilation speed is high enough to ensure interaction while development.

The Go programming language was developed itself when the multi-core processors existed. It is the main reason for builting-in the overlapping into the Golang. Go has goroutine instead of having the threads. The goroutines consume approximately 2 Kb of memory. That is why goroutines could be activated anytime. Among the advantages of goroutines could be highlated:

– goroutins have segmented extensible stacks that consumes using more memory when necessary.

– goroutines run faster than streams, it happens because of going with built-in primitives to safely share data.

– in case of simultaneous using of data structures, it is not necessary to use mutex locking.

– 1 goroutine can operate freely on multiple streams. Goroutins multiplexin a small number of the streams of operating systems.

The MPI computations in Golang should be written as a special algorithm. Obviously, it needs creation of the function that definitely should be straightforward and wrapper. The algorithm of writing such a function itself includes following:

1. Identification of the inputs and ouputs for Golang supports the return values with multiplication. The organization of the function signatures makes the usage of the multiplication more comprehensible. This feature requires separation the inputs from the ouputs. There are some exceptions as memory regions pointers that are used as buffers for arbitrary data.

2. Dispense with superfluous input/yield parameters: A few MPI capacities anticipate clusters of a particular sort (ordinarily numbers or MPI_-Datatype clusters). In C those capacities anticipate an information pointer as

well as the array size. In Golang we can make utilize cuts instep. Since the length/measure of a cut is a portion of its sort, an additional parameter for the estimate is unnecessary. In this manner, it is in some cases conceivable to decrease the parameter list in a sensible way to maintain a strategic distance from extra duplicate overhead (call by esteem) and keep the marks clear by diminishing their complexity.

3. Declare local variables for the output parameters:

MPI functions that are realized in C just return an error code and expect further output so-called output parameters. Obviously, we make use of the multiple return value feature of Golang. This requires that the wrapper holds a local variable for each output parameter that can be passed in the function.

4. Call the C work inside line sort transformation of the input parameters: before an input parameter can be passed into the C work it needs to be converted to the comparing C sort. Rather than investing extra nearby factors which would require memory and runtime, we straightforwardly change over the input parameters inside the contention list of the C function.

5. Convert the sorts of the neighborhood factors and return their worth to return the qualities, the covering got through the yield boundaries of the C capacity, the nearby factors which hold the information must be changed over to Golang types [1].

Go additionally has a particular form framework. When attempting to fabricate the ties the Go form devices hit their restriction of adaptability quick. There were a few issues when attempting to construct the ties for various executions on a solitary framework. The arrangement was to send a custom form framework written in Python. Go has to know the area of the common article documents of MPI to effectively construct the ties.

The Go form framework can use pkg-config. In any case, on many register groups the common article documents are situated in some sort of custom envelope and can't be found by pkg-config. It is feasible to pass the way of the common item documents to Go by sending out it as a climate variable. The Python assemble script first attempts to decide the way of the common item documents by utilizing pkg-config and afterward adding custom ways that are given to it by the order line. Another issue is that not all MPI executions are totally viable with the standard like they present diverse blunder codes. Since Go doesn't uphold macros the source code should be exceptionally produced.

When introducing the ties, it is important to pass the execution to the form script, so it realizes how to create the source code. Since a process group can have more than one MPI execution introduced, as OpenMPI and MPICH2. It should be feasible to introduce the ties more than once. The form content can produce various libraries of the ties in the event that there is more than one execution present. Taking a gander at the exhibition of the MPI covering and the Go language utilizing a custom benchmark and the different compilers used to make it. The benchmark utilizes the Jacobi strategy to settle incomplete differential conditions.

It is a line-by-line port of the C program bite the dust Go program is contrasted and the parallelism is just documented through the MPI covering. Each MPI cycle just comprises of a solitary goroutine. The benchmark is worked with various compilers and linkers. For example, the default GNU BFD linker, the gold linker created by Google for quicker connecting, and others. After the test, the outcomes showed that Go can't coordinate with the presentation of C however that should have been normal. One fascinating perception is that Go scales significantly more pleasant when utilizing non-obstructing correspondence when contrasting the hindering C execution and the impeding Go execution and contrasting the non-obstructing executions and one another.

Traditional bunch-based frameworks (like supercomputers) utilize equal execution between processors utilizing MPI. MPI is a correspondence interface between measures that execute in working framework examples on various processors; it doesn't uphold other cycle tasks like planning [2].

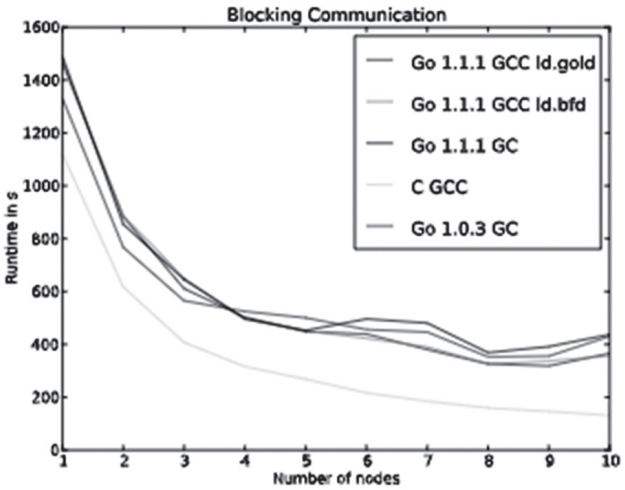


Fig.1. Using the Jacobi method for solving partial differential equation

The point is that Go 1.0.3 is quite faster than 1.1.1. The reason for this could be that Go 1.1.1 repeats slower over cuts than 1.0.3. When taking a gander at the non-obstructing correspondence benchmark clearly the GCC compiler portion preferred improvements over the GC compiler of Go.

The results appear that Go can not coordinate with the execution of C but that was to be anticipated. One curiously perception is that Go scales a part more pleasant when using non-blocking communication when comparing the blocking C usage with the blocking Go execution and and comparing the non-blocking implementations with each other [2].

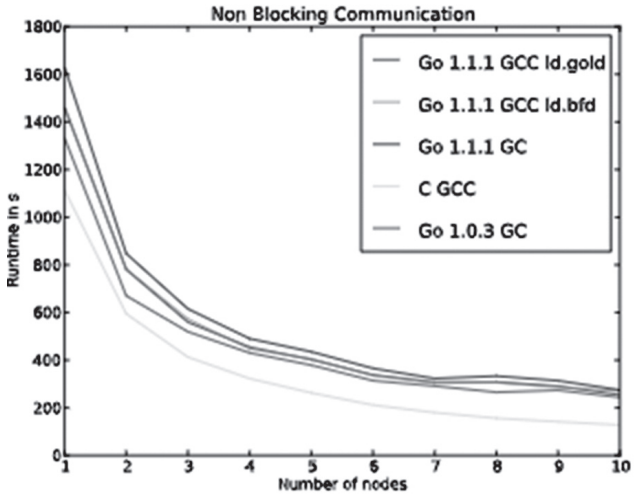


Fig. 2. Using the Jacobi method for solving partial differential equation

Besides all above said nothing was said about communicators, groups, field of communications [3]. A group is a set of branches. One branch may be a member of several groups. MPI_Group type and set of functions working with variables and constants of this type. The constant is actually two of them: MPI_GROUP_EMPTY can be returned if the group with the requested characteristics can be created, but does not yet contain any branch; MPI_GROUP_NULL returns when the requested characteristics are contradictory. According to the MPI concept, once a group is created, it cannot be supplemented or truncated - only a new group can be created under the required set of branches on the basis of the existing one. The field of communication («communication domain») is something abstract: there is no type of data at the programmer's disposal that describes directly the fields of communication, and there are no functions to manage them. Areas of communication are automatically created and destroyed along with communicators. Subscribers of one area of communication are all tasks of either one or two groups. The communicator or, or the communicator area descriptive, is the apex of a three-layer pie (groups, areas of communication, descriptions of communication areas) into which «baked» tasks: it is with communicators that a programmer deals, causing data transmission functions, and most of the support functions.

Why do we need different groups, different areas of communication and different descriptions? In essence, they serve the same purpose as message identifiers - helping the receiver branch and the receiver branch to identify each other more reliably, as well as the content of the message. Branches within a parallel application can be combined into subcollectors to solve intermediate tasks - by creating groups, and areas of communication over groups. Taking advantage of the descriptive of this area of communication, branches are guaranteed not to accept anything from outside subcollectional, and nothing is sent out. In parallel, they may continue to use any other non-subcollectional communicator at their disposal, such as MPI_COMM_WORLD, to exchange data within the entire application. Collective functions create a duplicate of the received communicator argument, and transmit data through a duplicate, without fear that their messages will be inadvertently confused with the messages of the «dot-dot» functions distributed through the original communicator. A programmer for the same purpose in different pieces of code can transmit data between branches through different communicators, one of which was created by copying the other.

Communicators are distributed automatically (by the functions of the «Create a New Comunicator» family), and for them there are no jokers («take it through any communicator») - two more of their essential advantages before message identifiers. Identifiers (integers) are distributed manually by the user and this is the source of two frequent errors due to confusion on the receiving side: Messages with different meanings are mistakenly assigned the same identifier manually. The reception function with the joker collects everything, including those messages that must be received and processed elsewhere in the branch. Intercommunicators and intracommunicators are also important parts in parallel programming in Golang. The descriptions of the communication regions respectively over two groups or over one. MPI_COMM_WORLD is an interagency communicator.

Intercommunicators are not a primary need for beginners, so there is no reference to them outside this paragraph. All the communicator functions mentioned in the document either do not distinguish between «intern» and «intern» at all or explicitly require an «intern». The latter are as follows:

– MPI_Bcast members;

– Topology management functions;

– MPI_Comm_xxx information functions, which are implemented through MPI_Group_xxx, such as MPI_Comm_size;

– MPI_Comm_remote_xxx is used instead [4].

Custom topologies. Inside the problem group are numbered linearly from 0 to (the size of group-1). However, another numbering system can be additionally imposed on them through the communicator. Such additional systems in MPI are two: the Cartesian n-dimensional lattice (cyclically and without), and the bioorientated graph. Functions are provided to create numbering (MPI_Topo_test, MPI_Cart_xxx, MPI_Graph_xxx) and to convert numbers from one system to another. This mechanism should not be perceived as providing an opportunity to adjust branch-to-branch connections to a hardware topology to increase speed; it merely automate the address recalculation that branches are supposed to perform, say, in matrix computation: Through the communicator is given a Cartesian coordinate system, where the coordinates of the branch coincide with the coordinates of the

submatrix it calculates. Multithreading. MPI itself implicitly uses multithreading very widely, and does not prevent the programmer from doing the same. However, different problems have MPI NECESSARILY different numbers, and different threads (threads) within the same problem do not differ [5]. The programmer himself must establish a discipline for threads so that one thread does not, say, cause MPI_Recv to be intercepted by the joker, which must receive and process another thread of the same task. Another source of errors can be the use of different threads of collective functions over the same communicator: use MPI_Comm_dup!File handling. In MPI-2, file redirection tools have been introduced, but not in MPI-1. All function calls are passed directly to the operating system (Unix/Parix/NFS/...) on that machine.

Conclusion:

As the conclusion could be said that nevertheless the Golang programming language is a very promising tool for developing massive systems that can speed up software code several times by properly converting sequential algorithms to competing ones, MPI developers are not recommended to use it due to its complexity for implementation. As a result, there is currently almost no MPI implemented by Golang methods and techniques that would fully integrate exchange and computation.

References:

[1] Gropp W. Using MPI: Portable Parallel Programming with the Message-Passing Interface. / W. Gropp, E. Lusk, S. Anthony – 328 c.

[2] A Golang Wrapper for MPI: https://hps.vi4io.org/_media/teaching/sommersemester_2013/paps-1213-beifuss_weging-golang_bindings_for_mpi-report.pdf

[3] MPI: A Message-Passing Interface Standard Version 3: https://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf.

[4] Alexander Beifuss, Johann Weging, Dr. Julian Kunkel. A Golang Wrapper for MPI: http://docplayer.net/50544314-A-golang-wrapper-for-mpi.html.

[5] Alan A. A. Donovan. The Go Programming Language / Alan A. A. Donovan, Brian W. Kernighan.. – 380 c.

[6] I. Balbaert. The Way to Go: A Thorough Introduction to the Go Programming Language. Iuniverse.Com, 3 2012.

The article was delivered to editorial stuff on the 24.02.2021

УДК 004.912 DOi 10.30837/ bi.2021.1(96).03

О.В. Рябишев[1], А.Л. Єрохін[2], А.Г. Бахмет[3]

[1]ХНУРЕ, м.Харків, Україна. oleksii.riabyshev.cpe@nure.ua,
ORCID iD: 0000-0002-7221-2227

[2]ХНУРЕ, м.Харків, Україна. Andriy.yerokhin@nure.ua,
ORCID iD: 0000-0002-8867-889X

[3]ХНУРЕ, м.Харків, Україна. andrii.bakhmet@nure.ua,
ORCID iD: 0000-0002-3293-9069

АНАЛІЗ ТОНАЛЬНОСТІ ТЕКСТУ УКРАЇНСЬКОЮ МОВОЮ

Стаття присвячена дослідженню методів автоматичного аналізу тональності тексту (сентимент-аналізу) та виявлення найбільш ефективних методів аналізу тональності тексту українською мовою. В результаті дослідження вирішена задача генерування набору даних (датасету) українською мовою на основі відгуків користувачів про мобільні додатки. Отриманий датасет використано для проведення експерименту з виявлення оптимального алгоритму бінарної класифікації для текстів українською мовою, а також побудована модель бінарного класифікатора на основі результатів експерименту. Вирішена задача бінарної класифікації тексту українською мовою за допомогою претренованої багатомовної BERT-моделі з використанням згенерованного датасету.

ТОНАЛЬНІСТЬ ТЕКСТУ, СЕНТИМЕНТ-АНАЛІЗ, УКРАЇНСЬКА МОВА, МАШИННЕ НАВЧАННЯ

Рябышев А.В., Ерохин А.Л., Бахмет А.Г. Анализ тональности текста украинского языка. Статья посвящена исследованию методов автоматического анализа тональности текста (сентимент-анализа) и выявление наиболее эффективных методов анализа тональности текста на украинском языке. В результате исследования решена задача генерирования набора данных (датасета) на украинском языке на основе отзывов пользователей о мобильных приложения. Полученный датасет использован для проведения эксперимента по выявлению оптимального алгоритма бинарной классификации для текстов на украинском языке, а также построена модель бинарного классификатора на основе результатов эксперимента. Решена задача бинарной классификации текста на украинском языке с помощью претренированной многоязычной BERT-модели с использованием сгенерированного датасета.

ТОНАЛЬНОСТЬ ТЕКСТА, СЕНТИМЕНТ-АНАЛИЗ, УКРАИНСКИЙ ЯЗЫК, МАШИННОЕ ОБУЧЕНИЕ

O.Riabyshev, A.Yerokhin, A.Bakhmet Analysis of the sentiment of the text in the Ukrainian language. The article is devoted to the study of the methods of automatic sentiment analysis and the identification of the most effective methods of analysis of the sentiment of the text in the Ukrainian language. In the course of the work, the problem of generating a dataset in Ukrainian was solved based on user reviews of mobile applications. The resulting dataset was used to conduct an experiment to identify the optimal binary classification algorithm for texts in the Ukrainian language, and a binary classifier model was built based on the results of the experiment. The problem of binary classification of Ukrainian text was solved using a pretrained multilingual BERT model using a generated dataset.

TEXT VOLUME, SENTIMENT ANALYSIS, UKRAINIAN LANGUAGE, MACHINE LEARNING

Вступ

Аналіз тональності тексту – це задача обробки природної мови, яка сьогодні широко використовується в таких областях, як соціологія (наприклад, збір даних із соціальних мереж про симпатії та антипатії людей), політологія (наприклад, збір даних про політичні погляди певних соціальних груп), маркетинг (наприклад, створення рейтингів товарів/компаній), медицина та психологія (наприклад, виявлення ознак психічних захворювань або ознак депресії в повідомленнях користувачів, виявлення хуліганів у соціальних мережах) [1]. Проте яким би корисним не був такий інструмент, системи аналізу тональності тексту українською мовою ще немає.

Метою даного дослідження є вивчення найбільш ефективних підходів до аналізу тональності тексту і знаходження прийнятного підходу для впровадження такого аналізатора для української мови на основі підходів, що базуються на алгоритмах машинного навчання.

Останніми роками було запропоновано велику кількість проєктів для аналізу тональності тексту відгуків про готелі, банки, оглядів ресторанів, коментарів до фільмів [2-7], відгуків про товари, повідомлення про політичні події тощо. Велика кількість досліджень присвячена аналізу тональності повідомлень у мікроблогах (наприклад, Twitter). Ці дослідження використовують різні підходи до аналізу тональності тексту: підхід, заснований на словниках та правилах, підхід з використанням машинного навчання. Можна виділити дослідження В. Каспера і М. Вела [8], Д. Кана [9], К. Мойланена та С. Пульмана [10]. Дослідженнями методів машинного навчання займалися Джейкоб Девлін [12], Мінг-Вей Чанг та Крістіна Тоутанова [13].

У цій роботі представлено дослідження в області автоматичного аналізу тональності тексту, що використовує підхід, заснований на алгоритмах машинного навчання.

1. Формування набору даних (датасету) українською мовою

У ході дослідження було сформовано датасет відгуків користувачів про мобільні додатки з платформи Google Play. Для отримання відгуків була використана бібліотека google-play-scraper для мови Python. API цієї бібліотеки приймає ідентифікатор додатка, відгуки для якого треба отримати, двобуквений код мови, в якому потрібно завантажити сторінку додатка та двобуквений код країни, який використовується для отримання додатків (потрібно, коли програма доступна лише в деяких країнах). Метою було отримати збалансований набір даних – з однаковою кількістю позитивних та негативних відгуків та з репрезентативними відгуками для кожного додатка.

Для фільтрування оцінки огляд було використано опцію бібліотеки google-play-scraper. Відгуки були відсортовано за їх корисністю – це ті відгуки, які Google Play вважає найважливішими.

Також була отримана підмножина найновіших відгуків (відсортовано за датою додавання). Із вмісту кожного відгуку було видалено смайли, символи транспорту, прапори та інші символи, що не є текстом. Також з датасету було видалено відгуки, що не містять тексту. Загалом було отримано 10 216 відгуків, які мають такі оцінки (рис. 1):

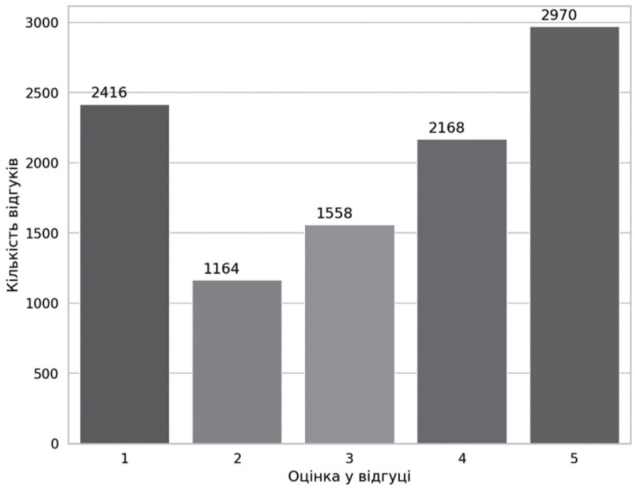


Рис. 1. Зібрані відгуки про мобільні додатки

Для задачі бінарної класифікації тональності тексту відгуки було розділено на дві категорії: ті, що мають негативну тональність, та ті, що мають позитивну тональність. Для позначення відповідних категорій було введено додатковий стовбець «sentimentScore», що має одне з двох можливих значень – 0 (текст з негативним емоційним забарвленням) або 1 (текст з позитивним емоційним забарвленням).

Відгуки з оцінкою 4 та 5 за п'ятибальною шкалою були позначені як позитивні, відгуку з оцінкою 1-3 – як негативні.

З рис. 2 видно, що було отримано збалансований датасет (з рівною кількістю текстів з позитивною та негативною тональністю).

Для отримання найкращих результатів навчання бінарної класифікації навчальні дані мають бути збалансовані (тобто число позитивних і негативних навчальних даних має бути однаковим). Відсутні значення необхідно обробити до навчання.

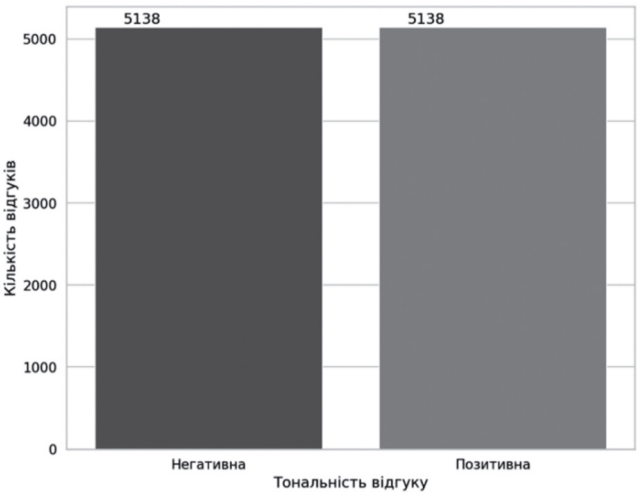


Рис. 2. Кількість відгуків в датасеті для задачі бінарної класифікації

Отриманий датасет був розділений на два набори: набір даних для тренування моделі та набір даних для тестування моделі.

2. Експериментальне дослідження алгоритмів аналізу емоційного забарвлення тексту українською мовою та побудова моделі класифікатора

За допомогою бібліотеки ML.NET проведено експеримент з виявлення оптимального алгоритму для виконання задачі бінарної класифікації тексту українською мовою на прикладі відгуків користувачів про мобільні додатки.

Бінарна класифікація – задача контрольованого машинного навчання, яка прогнозує розподіл елементів даних за двома класами (категоріям). На вхід алгоритму класифікації подається набір прикладів з мітками, кожна з яких представляє собою ціле число 0 або 1. Результатом роботи алгоритму бінарної класифікації є класифікатор, який вміє прогнозувати клас для нових екземплярів без мітки.

Для проведення експерименту використовувалися наступні алгоритми, для яких присутній API бібліотеки ML.NET:

– алгоритм лінійної бінарної класифікації з використанням середнього перцептрона (AveragedPerceptronTrainer);

– алгоритм бінарної логістичної регресії за допомогою стохастичного методу подвійних координат (SdcaLogisticRegressionBinaryTrainer);

– алгоритм на основі використання стохастичного градієнтного спуску (SymbolicSgdLogisticRegression BinaryTrainer);

– алгоритм на основі лінійної моделі логістичної регресії (LbfgsLogisticRegressionBinaryTrainer);

– алгоритми на основі бінарних дерев прийняття рішень (FastTreeBinaryTrainer, FastForestBinaryTrainer, LightGbmBinaryTrainer);

– алгоритм бінарної класифікації з узагальненими адитивними моделями (GamBinaryTrainer);

– алгоритм для прогнозування цільового об'єкту за допомогою моделі лінійної бінарної класифікації, що навчена лінійним SVM (LinearSvmTrainer).

Для проведення експерименту використаний зібраний нами датасет, який відповідає наведеним вище критеріям.

Дані були розділені на 2 категорії: датасет для тренування моделі (80% даних) та датасет для тестування (20% даних).

Результати експерименту (найкращі результати для кожного алгоритму) наведено в табл. 1.

Таблиця 1

Результати виконання зад–ачі бінарної класифікації тексту українською мовою для різних алгоритмів бібліотеки ML.NET

Клас моделі бібліотеки ML.NET	Точність (accuracy)	AUC (площа під кривою)	AUPRC	F1-метрика
SdcaLogistic RegressionBinary	0.8806	0.9380	0.9366	0.8748
AveragedPercep-tronBinary	0.8795	0.9344	0.9244	0.8713
SymbolicSgd LogisticRegres-sionBinary	0.8702	0.9162	0.9108	0.8617
FastTreeBinary	0.8801	0.9374	0.9422	0.8704
SgdCalibrated-Binary	0.8731	0.9295	0.9370	0.8703
LightGbmBinary	0.8681	0.9340	0.9222	0.8560
LinearSvmBinary	0.8445	0.9125	0.8979	0.8290
LbfgsLogistic RegressionBinary	0.8527	0.9130	0.9236	0.8544
FastForestBinary	0.7715	0.8482	0.8506	0.7306

Для отримання найкращих результатів навчання бінарної класифікації навчальні дані мають бути збалансовані, тобто число позитивних і негативних навчальних даних має бути однаковим. Відсутні значення необхідно обробити до навчання.

Як видно з табл. 1, найкращі результати для задачі бінарної класифікації тексту українською мовою має модель SdcaLogisticRegressionBinary, в основу якої покладено алгоритм логістичної регресії з використанням стохастичного методу здвоєних координат. Цей алгоритм заснований на методі стохастичного подвійного координатного підйому (SDCA), найсучаснішій методиці оптимізації опуклих цільових функцій.

Конвергенція забезпечується шляхом періодичного забезпечення синхронізації між первинними та подвійними змінними в окремому потоці. Також можна обрати декілька варіантів функцій втрат (наприклад, логістична втрата). Залежно від використовуваних втрат, навчена модель може бути, наприклад, машиною з підтримкою вектора або логістичною регресією. Метод SDCA поєднує в собі кілька найкращих властивостей, таких як можливість проведення потокового навчання (без розміщення всього набору даних у пам'яті), досягаючи розумного результату за допомогою декількох сканувань всього набору даних.

Побудовану модель з використанням алгоритму логістичної регресії було дотреновано і отримані наступні значення метрик, представлені в табл.2.

Враховуючи значення метрик, можна зробити висновок, що побудована модель класифікатора є якісною.

3. Використання багатомовної BERT-моделі для виявлення емоційного забарвлення текстів українською мовою

У ході дослідження було використано претреновану багатомовну модель BERT (Bidirectional Encoder Representations from Transformers) для задачі бінарної класифікації тексту українською мовою. Для цього було виконано наступні кроки:

– попередня обробка текстових даних для моделі BERT та побудова набору даних;

– використання навчання для побудови класифікатора настроїв;

– оцінка моделі за тестовими даними.

На відміну від моделей спрямованості, які послідовно зчитують введення тексту (зліва направо або справа наліво), кодер Transformer зчитує всю послідовність слів відразу. Тому він вважається двонаправленим, хоча було б точніше сказати, що він неспрямований. Ця характеристика дозволяє моделі вивчати контекст слова на основі всіх його оточень (ліворуч і праворуч від слова).

Модель BERT навчали, маскуючи 15% токенів з метою відгадати їх. Додатковою метою було передбачити наступне речення.

Зібраний в ході цього дослідження датасет був використаний для тренування моделі BERT для задачі бінарної класифікації тексту.

Бібліотека Transformers надає широкий спектр моделей Transformer (включаючи BERT). Це працює з TensorFlow та PyTorch. Сюди також входять токенізатори.

В якості моделі використано bert-base-multilingual-uncased модель.

Саму модель та токенизатор завантажено за допомогою бібліотеки Transformers.

Таблиця 2

Метрики побудованої моделі класифікатора з використанням алгоритму логістичної регресії SDCA

№ з.п.	Метрика	Опис метрики	Отримане значення	Еталонне значення
1	Точність (accuracy)	Частка правильних прогнозів за допомогою перевірочного набору даних. Це співвідношення числа правильно вгаданих і загального числа прикладів вхідних даних. Ця метрика працює добре, якщо існує аналогічна кількість вибірок, що належать кожному класу.	0.8820	Чим ближче до 1, тим ефективнішим вважається класифікатор. Точне значення 1 говорить про проблеми (зазвичай це витік міток і цілей, перенавчання або тестування за допомогою навчальних даних). Якщо тестові дані не збалансовані (більшість примірників відноситься до одного з класів), набір даних малий або оцінка підходить до значення 0 або 1, то точність не відображає фактичну ефективність класифікатора і вам потрібно перевірити додаткові метрики.
2	AUC (area under curve)	Площа під кривою оцінює площу під кривою, створеної підсумовуванням частот істинно позитивних результатів і помилково позитивних результатів.	0.9371	Чим ближче до 1, тим більша якість моделі. Для того, щоб модель була допустима, її значення повинно бути більше 0.5. Модель зі значенням AUC, що не перевищує 0,5, вважається непридатною.
3	AUPRC (area under precision recall curve)	Площа під кривою "точність - повнота": зручна міра успішного прогнозу, коли класи розрізняються (вкрай нерівномірно розподілені набори даних).	0.9413	Чим ближче до 1, тим більш точні результати повертає класифікатор. Високий рівень оцінки, близький до 1, показує, що класифікатор повертає точні результати (висока точність), а також повертає більшу частину всіх позитивних результатів (високий рівень повноти).
4	Позитивна точність (positive precision)	Частка документів, що дійсно належать до класу документів з позитивною тональністю, щодо всіх документів які система віднесла до цього класу.	0.9000	Чим ближче до 1, тим якісніше працює класифікатор з виявлення позитивних документів. Ця метрика застосовується разом з іншою метрикою щодо вилучення інформації – позитивною повнотою.
5	Позитивна повнота (positive recall)	Частка правильно визначених класифікатором документів, що належать до класу позитивних прогнозів, щодо всіх документів цього класу в тестовій вибірці.	0.8600	Чим ближче до 1, тим більш повно класифікатор визначає позитивні документи.
6	Негативна точність (negative precision)	Частка документів, що дійсно належать до класу документів з негативною тональністю, щодо всіх документів які система віднесла до цього класу.	0.8700	Чим ближче до 1.00, тим якісніше працює класифікатор з виявлення негативних документів. Ця метрика застосовується разом з іншою метрикою щодо вилучення інформації – негативною повнотою.
7	Негативна повнота (negative recall)	Частка правильно визначених класифікатором документів, що належать до класу негативних прогнозів, щодо всіх документів цього класу в тестовій вибірці.	0.9009	Чим ближче до 1, тим більш повно класифікатор визначає негативні документи.
8	F1	Показник F1 також називається збалансованою F-оцінкою або F-мірою. Це середнє гармонійне значення точності і повноти. Показник F1 корисний в тому випадку, якщо необхідно знайти баланс між точністю і повнотою.	0.8807	Чим ближче до 1,00, тим краще. Показник F1 досягає кращого значення в 1,00 і гіршого - в 0,00. Він повідомляє, наскільки точний класифікатор.

Модель bert-base-multilingual-uncased є попередньо натренованою моделлю на 102 мовах з найбільш повними версіями у Вікіпедії.

У ході дослідження виконано наступні етапи:

– речення поділені на токени;
– додано спеціальні токени;
– обрано довжину речення.

BERT працює з послідовностями фіксованої довжини. У дослідженні було використано просту стратегію для вибору максимальної довжини. Збережено довжину символів кожного огляду і оцінена їхня довжина (рис. 3).

Після перехресного тестування оцінено модель, результати оцінки наведені в табл. 3.

Більшість оглядів містять менше 150 токенів, але ми виберемо максимальну довжину 200.

Далі було побудовано модель нейронної мережі, проведено навчання нейронної мережі на тренувальному наборі даних та перевірено результат роботи моделі на тестових даних.

Щоб відтворити навчальну процедуру з документації BERT, було використано оптимізатор AdamW, наданий Hugging Face. Він виправляє зменшення ваги. Для моделі необхідно вказати функцію втрат і оптимізатор для навчання.

Оскільки розв'язувана задача є прикладом бінарної класифікації та модель буде показувати

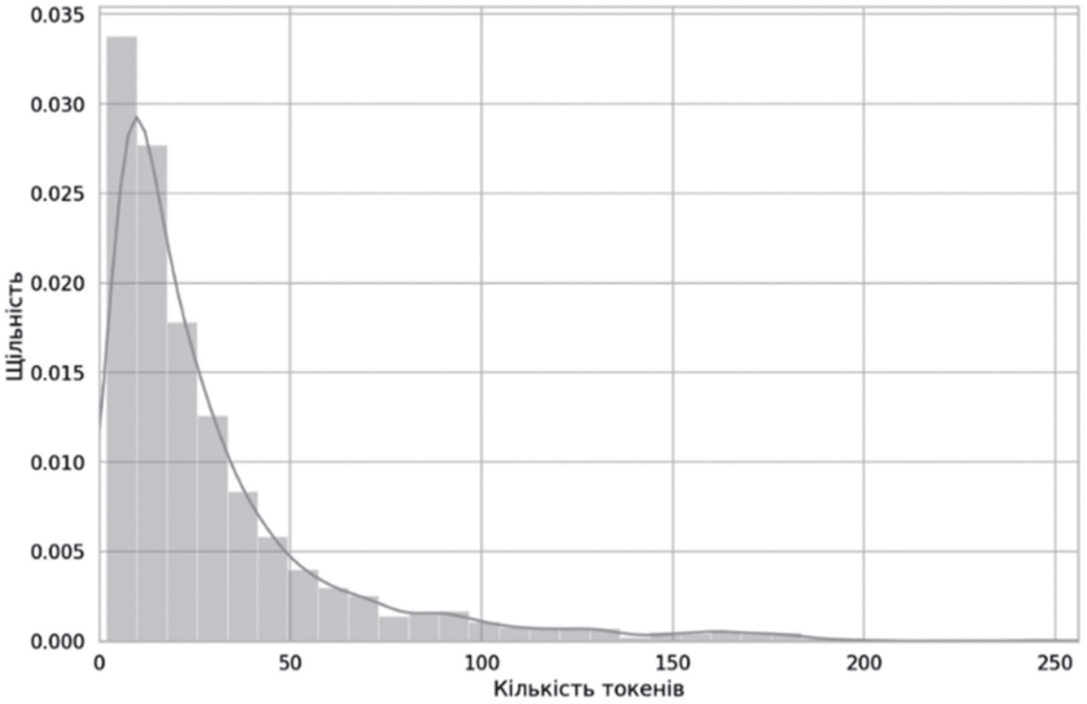


Рис. 3. Розподілення довжини токенів

Таблиця 3

Метрики налаштованої та дотренованої моделі BERT для задачі бінарної класифікації тексту

№ з.п.	Метрика	Опис метрики	Отримане значення	Еталонне значення
1	Точність (accuracy)	Частка правильних прогнозів за допомогою перевірочного набору даних. Це співвідношення числа правильно вгаданих і загального числа прикладів вхідних даних. Ця метрика працює добре, якщо існує аналогічна кількість вибірок, що належать кожному класу.	0.87	Чим ближче до 1, тим ефективнішим вважається класифікатор. Точне значення 1 говорить про проблеми (зазвичай це витік міток і цілей, перенавчання або тестування за допомогою навчальних даних). Якщо тестові дані не збалансовані (більшість примірників відноситься до одного з класів), набір даних малий або оцінка підходить до значення 0 або 1, то точність не відображає фактичну ефективність класифікатора
2	Позитивна точність (positive precision)	Частка документів, що дійсно належать до класу документів з позитивною тональністю, щодо всіх документів які система віднесла до цього класу.	0.87	Чим ближче до 1, тим якісніше працює класифікатор з виявлення позитивних документів. Ця метрика застосовується разом з іншою метрикою щодо вилучення інформації – позитивною повнотою.
3	Позитивна повнота (positive recall)	Частка правильно визначених класифікатором документів, що належать до класу позитивних прогнозів, щодо всіх документів цього класу в тестовій вибірці.	0.88	Чим ближче до 1, тим більш повно класифікатор визначає позитивні документи.
4	Негативна точність (negative precision)	Частка документів, що дійсно належать до класу документів з негативною тональністю, щодо всіх документів які система віднесла до цього класу.	0.87	Чим ближче до 1, тим якісніше працює класифікатор з виявлення негативних документів. Ця метрика застосовується разом з іншою метрикою щодо вилучення інформації – негативною повнотою.
5	Негативна повнота (negative recall)	Частка правильно визначених класифікатором документів, що належать до класу негативних прогнозів, щодо всіх документів цього класу в тестовій вибірці.	0.85	Чим ближче до 1, тим більш повно класифікатор визначає негативні документи.
6	F1	Показник F1 також називається збалансованою F-оцінкою або F-мірою. Це середнє гармонійне значення точності і повноти. Показник F1 корисний в тому випадку, якщо необхідно знайти баланс між точністю і повнотою.	0.87	Чим ближче до 1,00, тим краще . Показник F1 досягає кращого значення в 1,00 і гіршого - в 0,00. Він повідомляє, наскільки точний класифікатор.

ймовірність (шар з єдиного блоку з сигмоїдою у якості функції активації), то була використана функція втрат CrossEntropyLos.

Як відомо, автори BERT дають кілька рекомендацій щодо налаштування моделі: розмір партії (батчу): 16, 32; швидкість навчання (Адам): 5e-5, 3e-5, 2e-5; кількість епох (ітерацій): 2, 3, 4,...,10. Збільшення розміру партії значно скорочує час навчання, але дає меншу точність. Після навчання вимірюються втрати і точність нашої моделі шляхом перевірки на 20% зразків з перевірочного набору даних.

Можна зробити висновок, що дотренована модель BERT є якісною для задачі бінарної класифікації тексту українською мовою.

Для перевірки моделі використовується функція, що отримує на вхід два масиви – тестові рецензії та відповідні маркування позитивної/негативної рецензії. Результатом виконання цієї функції є пара чисел: відсоток втрат (loss) та точність (accuracy), як показано на рис. 4.

Порівняння метрик для моделей логістичної регресії та BERT наведено в табл. 4.

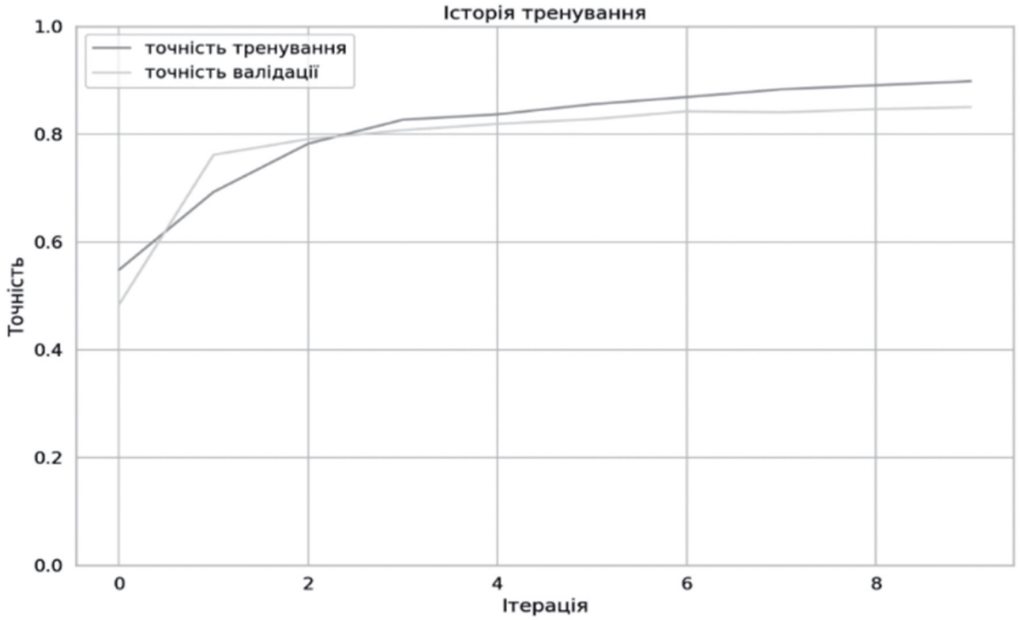


Рис. 4. Історія тренування моделі BERT на датасеті українською мовою

Таблиця 4

Порівняння значень метрик для моделі логістичної регресії та BERT-моделі

№ з.п.	Метрика	Отримане значення		Порівняння
		Модель логістичної регресії	BERT-модель	
1	Точність (accuracy)	0.88	0.87	Отримані значення свідчать про те, що класифікатори обох моделей є ефективними, але класифікатор моделі логістичної регресії є трохи ефективнішим за класифікатор BERT-моделі.
2	Позитивна точність (positive precision)	0.94	0.87	Класифікатор моделі логістичної регресії якісніше працює з виявлення тексту з позитивною тональністю, ніж класифікатор моделі BERT.
3	Позитивна повнота (positive recall)	0.94	0.88	Класифікатор моделі логістичної регресії більш повно визначає текст з позитивною тональністю, ніж класифікатор моделі BERT.
4	Негативна точність (negative precision)	0.90	0.87	Класифікатор моделі логістичної регресії якісніше працює з виявлення тексту з негативною тональністю, ніж класифікатор моделі BERT.
5	Негативна повнота (negative recall)	0.86	0.85	Класифікатор моделі логістичної регресії більш повно визначає текст з негативною тональністю, ніж класифікатор моделі BERT.
6	F1	0.88	0.87	Метрика характеризує точність класифікатора. Обидві моделі мають високу точність, але модель логістичної регресії в даному випадку більш точна за BERT-модель.

Після тренування та тестування моделей з використанням зібраного датасету отримані значення метрик для моделі логістичної регресії та для BERT-моделі свідчать про високу якість обох моделей для виявлення тональності тексту українською мовою.

Проте класифікатор моделі логістичної регресії характеризується більшою точністю, ефективністю з виявлення текстів з позитивною та негативною тональністю і може вважатися більш якісним для виконання задачі бінарної класифікації текстів українською мовою.

Висновки

У роботі досліджено підходи до аналізу тональності тексту з використанням алгоритмів машинного навчання. У ході дослідження був згенерований набір даних (датасет) текстів українською мовою на базі відгуків про мобільні додатки. Було проведено експериментальне дослідження з виявлення найефективнішого алгоритму для задачі бінарної класифікації тексту українською мовою та побудовано модель логістичної регресії, налаштовано і дотреновано багатомовну модель BERT для задачі бінарної класифікації тексту з використанням згенерованого датасету.

Список літератури:

[1] *Lerman K, Gilder A, Dredze M, Pereira F.* Reading the markets: forecasting public opinion of political candidates by news analysis. In: Proceedings of the 22nd international conference on computational. – Linguistics 1, 2008. – P. 473–480.

[2] *Khan A, Baharudin B, Lee LH, Khan K.* A review of machine learning algorithms for text-documents classification. – J Adv Inf Technol 1, 2010. – P. 4–20.

[3] Text classification and prediction using the Bag Of Words approach // Medium – a place to read and write big ideas and important stories. URL:https://medium.freecodecamp.org/text-classification-and-prediction-using-bag-ofwords-8aeb1396cded

[4] A General Approach to Preprocessing Text Data // Machine Learning, Data Science, Big Data, Analytics, AI. URL: https://www.kdnuggets.com/2017/12/generalapproach-preprocessing-text-data.html

[5] Bullinaria, John A., and Joseph P. Levy. Extracting semantic representations from word cooccurrence statistics: stop-lists, stemming, and SVD. – Behavior research methods 44.3, 2012. – P.890-907.

[6] The Stanford Natural Language Processing Group. URL: https://nlp.stanford.edu/IRbook/html/htmledition/support-vector-machines-and-machine-learning-on-documents-1.html (дата звернення: 05.03.2021).

[7] Тональний словник української мови // GitHub. URL: https://github.com/lang-uk/tone-dict-uk

[8] *Kasper W.* Sentiment Analysis for Hotel Reviews / Walter Kasper, Mihaela Vela. – Proceedings of the Computational Linguistics-Applications Conference. – Jachranka, Poland: Polskie Towarzystwo Informatyczne, Katowice, 10/2011. – P. 45–52.

[9] *Kan D.* Rule-based approach to sentiment analysis at ROMIP 2011 / Dmitry Kan. URL: http://www.slideshare.net/dmitrykan/rule-based-approach-to-sentiment-analysis-atromip-2011

[10] *Moilanen K.* Multi-entity Sentiment Scoring / Karo Moilanen, Stephen Pulman. – Proceedings of Recent Advances in Natural Language Processing (RANLP 2009). – Borovets, Bulgaria, September 14–16 2009. – P. 258–263.

[11] *Jonathan Herzig.* Unlocking Compositional Generalization in Pre-trained Models Using Intermediate Representations / Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, Yuan Zhang. // Cornell University. URL: https://arxiv.org/abs/2104.07478

[12] *Jacob Devlin.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova // Cornell University. URL: https://arxiv.org/abs/1810.04805

[13] *Peter Shaw, Ming-Wei Chang, Panupong Pasupat, Kristina Toutanova.* Compositional Generalization and Natural Language Variation: Can a Semantic Parsing Approach Handle Both? // Cornell University. URL: https://arxiv.org/abs/2010.12725

Надійшла до редколегії 17.03.2021

УДК 004.912 DOi 10.30837/ bi.2021.1(96).04

[1]В.О.Нечіпор, [2]А.Л.Єрохін
[1]ХНУРЕ, Україна, volodymyr.nechipor.cpe@nure.ua
[2]ХНУРЕ, Україна, andriy.yerokhin@nure.ua, ORCID iD: 0000-0002-8867-889X

МОДИФІКАЦІЯ МЕТОДУ КЛАСИФІКАЦІЇ БАЙЄСА В ЗАДАЧАХ ВИЯВЛЕННЯ СПАМУ УКРАЇНСЬКОЮ МОВОЮ

Стаття присвячена аналізу існуючих технологій для виконання задачі класифікації української мови з метою фільтрації спаму. В рамках дослідження було проаналізовано недоліки методу класифікації Байєса в рамках сучасної реалізації цього методу на мові програмування Python для роботи з українською мовою. Основним недоліком програмної реалізації методу Байєса було виявлено некоректний для української мови поділ на слова за умови, що слова містять апостроф. Для виправлення цієї проблеми було розроблемо модифікований метод класифікації за Байєсом, який коректно працює зі словами української мови, що містять апостроф. В результаті вдалось підняти ефективність спрогнозованого класифікування спаму з 86% до 91%.

МАШИННЕ НАВЧАННЯ, МЕТОД КЛАСИФІКАЦІЇ БАЙЄСА, УКРАЇНСЬКА МОВА, АПОСТРОФ, СПАМ, ПРОГНОЗУВАННЯ

Нечипор В.А., Ерохин А.Л. Модификация метода классификации Байеса в задачах обнаружения спама на украинском языке. Статья посвящена анализу технологий решения задачи классификации украинского языка с целью фильтрации спама. В рамках исследования были проанализированы недостатки метода классификации Байеса в рамках современной реализации этого метода на языке программирования Python для работы с украинским языком. Главным недостатком программной реализации метода Байеса был выбран некорректный для украинского языка разбор на предложения на слова при условии, что слова содержат апостроф. Для исправления этой проблемы был разработан модифицированный метод классификации по Байесу, который корректно разбивает предложения на слова, даже при наличии апострофов. В результате удалось поднять эффективность прогнозированной классификации спама с 86% до 91%.

МАШИННОЕ ОБУЧЕНИЕ, МЕТОД КЛАССИФИКАЦИИ БАЙЕСА, УКРАИНСКИЙ ЯЗЫК, АПОСТРОФ, СПАМ, ПРОГНОЗИРОВАНИЕ

Nechipor V.O., Yerokhin A.L. Modification of the Bayes classification method in the tasks of detecting spam in the Ukrainian language. Existing technologies for the task of Ukrainian language classification with in order to filter spam are analyzed in this article. As part of the study, the shortcomings of the Bayesian classification method towards Ukrainian language were analyzed in the context of the modern implementation of this method in Python programming language. The main disadvantage of the software implementation of the Bayesian method was incorrect for the Ukrainian language parsing of sentences into words, provided that the words contain apostrophes. To correct this problem, a modified Bayesian classification method was developed, which correctly splits sentences into words, even if they contain apostrophes. As a result, the efficiency of the predicted spam classif ication was raised from 86% to 91%.

MACHINE LEARNING, BAYES CLASSIFICATION METHOD, UKRAINIAN LANGUAGE, APOSTROPHE, SPAM, FORECASTING

Вступ

Життя в сучасному світі тісно пов'язане з онлайн присутністю – Інтернет є джерелом новин, соціальною мережею, архівом інформації про будь-що з історії, робочою платформою, інструментом для освіти, тощо. У 2020-му році більш ніж 4.5 мільярди людей постійно користувались Інтернетом, а це більша половина населення планети [1].

Невід'ємною частиною онлайн присутності є електронна пошта, яка використовується для ідентифікації користувача в мережі і листування, отримання звітів і чеків про купівлю в мережі. Згідно останньою статистики щоденно відправляється 319 мільярдів листів [2], з яких, очевидно, далеко не всі є бажаними. Електронна пошта є не тільки зручним інструментом, але й небезпекою – величезна кількість небажаних листів щоденно розсилається багатьом користувачам, чия електронна адреса стала відома тим, хто розсилає їх. Саме феномен відправки небажаних листів називається спамом, а зміст листів може бути як просто набридливою рекламою, так і погрозами, спробами обманути і заманити на не доброчесний сайт, перейти за фішинговим посиланням, розсилкою з інформацією про незаконні ресурси: порно, жорстокі веб-сайти, тощо. Феномен спаму набув нечуваних масштабів у світі, що робить розробку сервісів захисту від спаму надзвичайно актуальним завданням.

В основі будь-якої програми захисту від спаму (спам-фільтрів) лежить мовний аналіз, а саме алгоритм, який зможе класифікувати повідомлення за бінарним критерієм «спам»/«не спам». Необхідність мовного аналізу ускладнює створення спам-фільтрів тим, що різні мови належать до різних груп мов, відрізняються граматичними і семантичним правилами побудови речень, абеткою, містять унікальні допоміжні символи (як апостроф в українській мові).

На сьогодні існують готові рішення для спам-фільтрів найбільш популярних мов, в першу чергу для англійської, як для найпоширенішої мови світу. В той же час український сектор лише починає

розвиватись в напрямку створення сервісів, які вміють досконало аналізувати українську мову.

Спам-фільтр є одним зі складних прикладів програмного забезпечення і вимагає застосування машинного навчання. В рамках дослідження розроблено нейронну мережу з вчителем.

У даній роботі проаналізовано методи класифікації текстів з метою фільтрування спаму українською мовою. Для виконання задачі дослідження обрано метод наївної класифікації Байєса як метод, який, на думку дослідників, найкраще підходить для даної мети.

Для програмної реалізації методу Байєса було обрано Python, як мову з найбільшою історією роботи з машинним навчанням. Саме на Python були розроблені одні з перших нейронних мереж, що слугувало поштовхом подальшого розвитку інфраструктури саме цією мовою.

Оскільки перші кроки для створення програмного забезпечення машинного навчання з аналізу української мови потребують якомога більшої підтримки, було вирішено, що Python і його сучасна база буде кращим рішенням. Під час написання програми було використано найбільш розповсюджену бібліотеку з уже реалізованим методом Байєса — nltk. В процесі реалізації методу наївної класифікації Байєса для класифікації текстів українською мовою на мові Python було знайдено перепону, яка заважала правильно класифікувати деякі повідомлення за ознакою «спам»/«не спам», а саме - український апостроф, який на відміну від англійського апострофа, часто є невід'ємною частиною кореня слова, а не способом скорочення речення шляхом поєднання двох різних слів. Внаслідок цього підготовка тексту до класифікації за Байєсом некоректно розділяла речення на слова і призводила до хибних висновків.

1. Постановка задачі

Об'єктом даного дослідження є мовний аналіз текстів з метою їх подальшої класифікації, а предметом дослідження – методи класифікації текстів українською мовою.

Основним завданням роботи є розробка модифікованого методу наївної класифікації Байєса з метою покращення результатів обробки листів українською мовою. Виділимо наступні завдання, які необхідно виконати в рамках дослідження:

1) проаналізувати аналіз методів класифікації текстів;

2) розробити програмне забезпечення спам-фільтру українською мови з використанням методу наївної класифікації Байєса;

3) розробити модифіковане програмне забезпечення спам-фільтру українською мови з використанням методу наївної класифікації Байєса, яке коректно розділятиме українське речення на слова;

4) порівняти дві реалізації.

2. Аналіз задач та методів класифікації тексту

Проаналізуємо основні методи класифікації текстів, розберемо їхні особливості з точки зору придатності для використання в методах фільтрації спаму українською мовою. Методи класифікації текстів лежать на межі двох областей – машинного навчання та інформаційного пошуку [3]. В рамках даного дослідження в першу чергу аналізуватимемо саме автоматичну класифікацію текстів, адже саме вона потрібна для реалізації програмного забезпечення спам-фільтру українською мови. Для успішного функціонування система автоматичної класифікації потребує розробок на основі машинного навчання, а саме – створення нейронної мережі з вчителем. У випадку створення програмного забезпечення спам-фільтру потрібна наявність бази даних листів чи текстів які вже класифіковані за категорією «спам»/«не спам». При цьому враховуємо, що автоматична класифікація складається з двох етапів: навчання моделі і її використання.

На етапі навчання моделі необхідно виконати опис множини заздалегідь визначених категорій і представлення тренувального набору елементів з уже визначеною категорією. Модель алгоритму побудує власні правила класифікації на основі певної тренувальної вибірки даних, які ми їй запропонуємо. Саме крок підбору даних для тренування моделі є найбільш важливим для правильного функціонування моделі, адже неправильно класифіковані дані або їх дуже мала кількість можуть спричинити збої в роботі моделі. Використання моделі полягає у визначенні категорій нових, раніше невідомих даних. Модель вважається успішною, якщо кількість правильно здійснених класифікацій перевищує 50%.

Задачу класифікації можна представити як необхідність визначити множину, до якої належить певний текст українською мовою, або необхідність знайти функцію, яка буде максимально наближена до ідеалу. Кожному елементу t ставиться у відповідність набір ознак $t = \{X_i\}$. Далі застосовується алгоритм класифікації для виділення текстів, найбільш відповідних заданому класу [4].

Для класифікації застосовуються різноманітні методи, кожен з яких має свої переваги і особливості використання. Переважна більшість методів класифікації текстів так чи інакше засновані на припущені, що тексти, які відносяться до однієї категорії, мають однакові ознаки (слова чи словосполучення), і наявність чи відсутність таких ознак в тексті визначає його приналежність чи неприналежність до тієї чи іншої теми. Таким чином, для кожної категорії повинна бути множина ознак, яку називають словником, через те, що вона складається з лексем, які включають слова та/або словосполучення, що характеризують категорію. Подібно категоріям, кожен текст також має ознаки, за якими його можна

віднести з деяким ступенем вірогідності до однієї чи декількох категорій. Множина ознак усіх текстів повинна співпадати з множиною ознак категорій.

Задача методів класифікації текстів полягає в тому, щоб якнайкраще обрати такі ознаки і сформулювати правила, опираючись на які й буде прийматися рішення щодо віднесення документа до рубрики. Найбльш відомими з цих методів є [5]: класифікація через дерево рішень; Байєсівська (наївна) класифікація; класифікація за методом опорних векторів; класифікація за методом найближчого сусіда; класифікація штучними нейронними мережами. Для подальшого дослідження і вдосконалення оберемо метод наївної класифікації Байєса.

Наївний Байєсівський класифікатор традиційно використовується в задачах класифікації текстів, таких як фільтрація спаму, автоматична рубрикація або визначення тональності документа. Набули поширеного розвитку два його різновиди: багатомірний (multivariate) та мультиноміальний (multinomial). У загальному вигляді визначають найбільш вірогідний клас алгоритмом наївної байєсівської класифікації; класифікація зводиться до обчислення максимального значення аргументу, при відомому наборі незалежних ознак $x_1, x_2, \supset x_n$.

Класифікація тексту при цьому виглядає так:

$$C(T) = \max \sum (t_1, t_2, \ldots t_n | C),$$

де T – текст, що класифікується, а $t_1, t_2, \supset t_n$ – набір речень тексту. Так, приналежність тексту до тієї чи іншої категорії зводиться до обчислення максимального значення суми коефіцієнтів приналежності реченням [6].

Зазначений метод використовує ймовірнісну модель, в якій класифікація та включення у відповідну категорію документів проводиться шляхом оцінювання ймовірності появи слів у документі. Ймовірності можуть бути використані для оцінки найбільш близьких категорій тестового документа [7].

Метод класифікації Байєса має декілька різних варіацій, кожна з яких краще підходить для вирішення тієї чи іншої задачі. Наївний класифікатор найкраще себе показує тоді, коли певну множину об'єктів потрібно класифікувати – розбити на класи. Однак наївний класифікатор менш ефективний, коли логіка розподілення має бути трішки складнішою. Мережевий класифікатор Байєса, наприклад, дозволить спочатку згрупувати множину об'єктів за проміжними спільними ознаками і знайти залежність класу від однієї чи групи ознак. Основні переваги наївного байєсівського класифікатора – це простота реалізації і низькі обчислювальні витрати при навчанні та класифікації. У порівнянні, наприклад, з методом k-найближчого сусіда, який пропонує порівнювати аналізований документ з усіма документами з навчальної вибірки і тому вимагає тривалого часу роботи, алгоритм Байєса швидший в десятки разів [8]. Основним недоліком методу є відносно невисока якість класифікації в більшості реальних завдань [8]. Зазначений метод часто використовується як базовий метод при порівнянні різних методів машинного навчання.

3. Датасет спам фільтру

Задача фільтрації спаму не нова, найбільшого розвитку вирішення цієї проблеми досягло саме в рамках англійської мови, як однієї з найбільш популярних мов в Інтернет кореспонденції. На веб-сайті kaggle.com, наприклад, легко знайти чіткі інструкції з налаштування спам-фільтру [9], а також готові дата сети текстів, які вже класифіковані за принципом «спам»/«не спам».

Збір, аналіз і класифікація такого датасету є окремою масштабною задачею, тому його наявність конкретною мовою є важливим елементом для підготовки в роботі зі спам-фільтром. В рамках роботи над поставленою задачею не було виявлено доступних датасетів спам текстів українською мовою. Це підтверджує актуальність даного дослідження і створює певні складнощі для досягнення мети роботи.

Для продовження роботи над аналізом методів класифікації текстів було зібрано власний датасет відносно невеликого обсягу розміром в декілька сотень текстів. Частина текстів була зібрана з поштових ящиків і смс-повідомлень. Оскільки мета спам-повідомлень різною мовою приблизно однакова – повідомити небажані новини або спробувати обманути і виманити дані – знехтуємо тим, що смс-повідомлення і електронні листи часто дещо відрізняються об'ємами тексту, що передається.

Метадані повідомлення в рамках цього дослідження також не бралися до уваги (інформація щодо адреси поштової скриньки відправника, сайту відправника, рейтингу доброчесності відправника, заголовка листа, тощо). Предметом дослідження стало власне тіло листа українською мовою. Зібраний датасет обсягом 356 текстових повідомлень, що включають 30% спам повідомлень і 70% звичайних повідомлень було оформлено в такому вигляді (табл.1):

Таблиця 1

Датасет повідомлень українською

Label	Body
spam	МЕГА РОЗПРОДАЖ! Знижки -70%+ додатково -20%: http://bit.ly/**
ham	Добрий день, гадаю, в такому вигляді можна прийняти звіт

Спам повідомлення отримали маркування «spam», а звичайні повідомлення – «ham». У сфері спам фільтрів досить давно ввели поняття «ham» як коротке і співзвучне зі спамом для маркування повідомлень, що не являються спамом. Адже кожного разу використовувати «не спам» досить не зручно. Дотримуватимемось загальноприйнятої тенденції в цьому дослідженні.

4. Практична реалізація метода Байєса

Для практичної реалізації методу Байєса використаємо сервіс Datalore від компанії Jetbrains, який надає безкоштовний доступ до командої оболонки для інтерактивних розрахунків Jupiter Notebook. Спочатку спробуємо використати уже розроблений мультиноміальний метод Байєса з бібліотеки nltk. На першому кроці випадковим чином розіб'ємо датасет на частину для навчання і частину для перевірки, де 80% усіх повідомлень будуть використані для навчання. Після цього навчаємо модель на основі тренувального датасету і проводимо тестування. Метод включає в себе наступні кроки:

1) прибираємо дублікати повідомлень;

2) прибираємо пунктуацію;

3) трансформуємо всі літери в нижній регістр;

4) розбиваємо речення на слова;

5) прибираємо стоп-слова (слова, які не мають важливого значення, коли відірвані від контексту).

6) вираховуємо вагу кожного слова відносно категорії (залежить від частоти використання в певній категорії в навчальному датасеті);

7) перевіряємо результат на тестовому датасеті.

В даному дослідженні ефективність роботи класифікаторів за Байєсом вимірювалась за шкалою від нуля до одиниці, де 1 це 100% правильно класифікованих повідомлень, а 0 – жодного правильно класифікованого повідомлення. У результаті фільтрування спаму отримуємо результат 0.86 (рис. 1). Це показник ефективності, який частково залежить від невеликого обсягу датасету і від того, що весь датасет отримано з поштової скриньки однієї людини.

```
              precision    recall  f1-score   support

           0       0.76      1.00      0.87        13
           1       1.00      0.75      0.86        16

    accuracy                           0.86        29
   macro avg       0.88      0.88      0.86        29
weighted avg       0.89      0.86      0.86        29

Confusion Matrix:
 [[13  0]
 [ 4 12]]

Accuracy:  0.8620689655172413
```

Рис. 1. Ефективність мультиноміального методу Байєса

Звернімо увагу на те, що всі 100% листів не були класифіковані правильно. Проаналізуймо помилкові приклади і спробуємо знайти закономірність, притаманну українській мові. Серед хибно класифікованих повідомлень знайдемо декілька прикладів з словами, що містять апострофи і висуваємо теорію, що саме це може бути причиною хибного висновку.

Таблиця 2

Хибно класифіковані повідомлення

Label	Body	Prediction
spam	Спекотна п'ятниця! 50% на все. Тільки 3 дні.	ham
spam	М'язиста пропозиція! Дев'ять днів тренування за кошт п'яти. ФК Плутон на Широнінців.	ham
ham	Припиню зв'язок з сім'єю на декілька днів доки не з'ясую свою ситуацію із здоров'ям.»	spam

Пояснимо причину такого висновку на першому прикладі, відслідкувавши трансформацію повідомлення протягом семи описаних кроків трансформації тексту.

Прибираємо пунктуацію: *«Спекотна п ятниця 50 на все Тільки 3 дні»*.

Трансформуємо всі літери в нижній регістр: *«спекотна п ятниця 50 на все тільки 3 дні»*.

Розбиваємо речення на слова: *[‘спекотна’, ‘п’, ‘ятниця’, ‘50’, ‘на’, ‘все’, ‘тільки’, ‘3’, ‘дні’]*.

Прибираємо стоп-слова (слова, які не мають важливого значення, коли відірвані від контексту): *[‘спекотна’, ‘все’, ‘тільки’, ‘дні’]*.

Вираховуємо вагу кожного слова відносно категорії (залежить від частоти використання в певній категорії в навчальному дата сеті). Кількість слів у реченні, яке ми розглядаємо, невелика: після видалення слова «п'ятниця» залишилось лише чотири слова (табл. 3).

Таблиця 3

Розбір прикладу речення на оцінені слова

Слово	Кількість використань в спамі	Кількість використань не в спамі
‘спекотна’	0	0
‘все’	21	23
‘тільки’	6	4
‘дні’	8	10

Наступним кроком за допомогою алгоритма оцінена кількість вживань даних слів у повідомленнях, промаркованих як «спам» і «не спам» у тренувальному датасеті. Кількість вживань у повідомленнях з позитивною конотацією переважає у двох з чотирьох слів, а одне із слів не вживалось в жодному з повідомлень обраного тренувального дата сету. Це лише початкова обробка тексту після якої необхідна обробка даних алгоритмом Байєса, однак цих даних вже достатньо, щоб зрозуміти, що алгоритму надається некоректна інформація, що веде до хибного висновку.

Як уже було зазначено, слово «п'ятниця» було поділене на «п» і «ятниця», тому що стандартний механізм поділу, розроблений з огляду на англійську мову, замінює апостроф на пробіл, і далі ділить по ньому.

При фінальній класифікації текстів вага кожного слова в категоріях «spam» і «ham» дуже важлива і може схилити класифікатор на одну чи іншу сторону. Якби слово «п'ятниця» було розпізнане вірно, воно мало б високий негативний коефіцієнт ймовірності бути в спамі, тому що лише серед підібраного невеликого дата сету було декілька повідомлень про «Чорну п'ятницю» і супутні знижки. Оскільки слово розпізнане не було, то його коефіцієнтом знехтували, а слова, що залишились, були недостатньо переконливими для вірної класифікації.

Схожа ситуація і з іншими прикладами – багато з них були класифіковані некоректно через те, що аналізатор зміг обробити лише частину слів через неправильний поділ речення на слова.

Отже, після аналізу речення *«Спекотна п'ятниця! 50% на все. Тільки 3 дні.»* зробимо висновок, що наявність апострофів в словах української мови є перепоною для роботи алгоритму класифікації Байєса. Серед хибно класифікованих речень більшість містять апострофи, які, якщо і не є головною причиною хибного висновку, є причиною неможливості коректно класифікувати конкретні слова. Це, в свою чергу, призводить до зменшення кількості слів, які могли б схилити алгоритм на одну із сторін класифікації – «спам»/«не спам».

5. Реалізація методу наївної класифікації Байєса

Метод мультиноміальної класифікації Байєса показав результат в 86% ефективності класифікації повідомлень на «spam» і «ham». Оскільки саме варіації методу Байєса в вирішенні задачі класифікації повідомлень на дві основні категорії вважаються найбільш ефективними, порівняємо метод мультиноміальної класифікації з методом наївної класифікації Байєса. Для подальшого аналізу було реалізовано метод наївної класифікації Байєса для того щоб була можливість модифікувати його частини з метою усунення проблеми виявлення слів з апострофами.

Спочатку завантажуємо тренувальну базу до алгоритму і виконуємо приведення до нижнього регістру і прибирання пунктуації. В першій реалізації методу наївної класифікації Байєса жодної оптимізації по роботі з апострофами не очікується. Спочатку порівняймо який із варіантів методу краще впорається з задачею класифікації тексту на «spam» і «ham».

Одразу після виконання частини програмного коду сервіс Datalore дозволяє вивести частину результату для візуалізації. Було виведено три останні екземпляри листів з тренувального датасету. Лише серед початків речень трьох екземплярів видно проблемне місце зі словом «кур'єр», яке було трансформовано в «кур» і «єр» і, відповідно, не буде вірно проаналізоване з точки зору використання в листах спаму і бажаних листах.

Наступним кроком реалізації алгоритму є розбиття речення на слова і підрахунок кількості використання слів в конкретних реченнях включаючи асоціацію з відповідним типом повідомлення: «spam» і «ham». Результат виконання цієї частини наведено у табл. 4, яка представлена у вигляді матриці; в ній наведено приклад на основі трьох колонок, а отже трьох унікальних слів.

Таблиця 4

Розбір прикладу речення на оцінені слова

label	body	підозра	єр	витрачайте
ham	[«є», «підозра», «що», «ловити» ...]	1	0	0
spam	[«володимире», «не», «витрачайте», «кошти» ...]	0	0	1
ham	[«вітаємо», «кур», «єр», «в», «дорозі» ...]	0	1	0

Повна кількість колонок дорівнює кількості унікальних слів в повідомленнях. Значення нуля чи одиниці означає відповідно відсутність і використання слова в повідомлені, що вказане в даному рядку.

На даному етапі можна вважати, що навчання моделі виконано, за умови що вищевказану класифікацію окремих слів було пройдено на датасеті великого розміру.

Оскільки готового класифікованого датасету спам повідомлень українською мовою о не було, обмежимось кількістю повідомлень в 364 одиниці, які були зібрані власноруч.

Для навчання моделі використаємо приблизно 80% датасету, що складає 293 повідомлення. Повідомлення, що залишились, розміром в 71 одиницю, будуть використані для перевірки тренованої моделі. Результат роботи наївного класифікатора Байєса на тому ж самому датасеті, який було використано в реалізації мультиноміального методу, наведено на рис. 2.

```
Correct: 60
Incorrect: 11
Accuracy: 0.8450704225352113
```

Рис. 2. Ефективність наївного класифікатору за Байєсом

Отже ефективність наївного класифікатору за Байєсом складає 84%, тобто 84% повідомлень були класифіковані вірно. Це на 2% менше ніж за мультиноміальним методом класифікації Байєса. Невеликий відсоток різниці обумовлений, перш за все, відносно невеликим розміром тестового дата сету. Наявність різниці дозволяє нам припустити, що мультиноміальний метод класифікації Байєса є більш ефективним для виконання задачі класифікації листів на «spam» і «ham» за умови наявності більшого тренувального дата сету. Проте і мультиноміальний і наївний методи підходять для успішного виконання задачі класифікації спаму.

Проведемо статистичне порівняння ефективності методів класифікації. Оскільки кожен з методів класифікації було використано лише один раз, ми не можемо бути достеменно впевнені в коректності наведених результатів. Існує декілька основних підходів до поділу датасету на тестовий і тренувальний під час навчання моделі:

Свідомо використовуємо один і той самий набір тренувальних текстів та не міняємо їх між використаннями алгоритму. Перевага: дозволяє уникнути непередбачуваності тренувальних даних після першої обробки. Недолік: за умов неправильно підібраного тренувального датасету, модель приречена на провал, адже у неї немає шансу перевчитись.

Випадково розбиваємо набір даних на тренувальні і тестові під час кожного навчання моделі у розробці. Перевага: усуває проблему першого підходу – модель може перенавчитись, якщо в одному з попередніх використань було підібрано невірний датасет. Недолік: випадковість результату ефективності класифікації викликає недовіру до одного отриманого результату, який може бути «випадково» успішним.

Під час попередніх вимірів ефективності мультиноміального і наївного класифікаторів за Байєсом був використаний другий підхід з випадковістю розподілення даних на тестові і тренувальні. Для того, щоб впевнитись в отриманих результатах і усунути або хоча б мінімізувати вплив випадковості результатів, збережемо результат виконання кожного з алгоритмів щонайменше десять разів і знайдемо середню ефективність. Результат навчання моделей наведено у табл. 5.

Таблиця 5

Середнє арифметичне ефективності мультиноміального і наївного методів класифікації

Спроба	Мультиноміальний	Наївний
1	0,862	0,845
2	0,794	0,816
3	0,884	0,810
4	0,826	0,873
5	0,869	0,788
6	0,783	0,845
7	0,873	0,802
8	0,836	0,760
Спроба	**Мультиноміальний**	**Наївний**
9	0,921	0,901
10	0,894	0,830
Середнє	**0,854**	**0,827**

Середнє арифметичне результатів ефективності використання мультиноміального методу класифікації Байєса складає 0,854, а середнє арифметичне результатів методу наївної класифікації Байєса 0,827. Різниця ефективності збереглась, і навіть збільшилась на користь мультиноміального методу Байєса з 0,017 до 0,027.

На основі проведеного дослідження можемо зробити остаточний висновок, що мультиноміальний метод класифікації Байєса є більш ефективним, ніж метод наївної класифікації для задачі класифікації спаму листів українською мовою.

Під час дослідження обох методів класифікації було помічено проблему розподілення слів речення на слова у зв'язку з українськими апострофами, які, на відміну від англійських апострофів, є невід'ємною частиною слова.

З метою подальшого вдосконалення способів вирішення задачі класифікації текстів з метою фільтрування спаму українською мовою внесемо модифікацію, пов'язану з вирішенням проблеми апострофів, в метод наївної класифікації, який вже було реалізовано.

6. Реалізація модифікованого наївного методу класифікації Байєса

Модифікуємо частину коду, яка відповідає за розбиття речення на слова. Для цього потрібно змінити налаштування пунктуаційних символів, щоб апостроф перестав вважатись символом пунктуації і став вважатись частиною слова. Також необхідно впевнитись, що внесені зміни не зменшують ефективність інших частин алгоритму. Переглянемо символьну таблицю і визначимо символи пунктуації, які необхідно прибрати з речення українською мовою. Далі використаємо модифіковану частину програмного коду для роботи з пунктуаційними символами в методі наївної класифікації Байєса. При розбитті речень на слова після модифікації отримуємо:

1. ['на', 'ваше', 'ім'я', 'надійшли', 'кошти', 'для', 'отримання', 'дзвоніть', '080****305', 'код', '22'];
2. ['спекотна', 'п'ятниця', '50', 'на', 'все', 'тільки', '3', 'дні'].

Слова з апострофами тепер сприймаються правильно і будуть мати коректний коефіцієнт ймовірності спаму. Це особливо суттєво в невеликих повідомленнях, де може бути лише декілька навантажених важливим значенням слів. Якщо якусь частину з них просто прибрати так само як «стоп-слова» то класифікація найбільш ймовірно буде проведена хибно. Виміряємо ефективність модифікованого методу на тому ж дата сеті, що й класичні методи мультиноміальної та наївної класифікації. Результат наведено на рис. 3.

```
Correct: 65
Incorrect: 6
Accuracy: 0.9154929577464789
```

Рис. 3. Ефективність Модифікованого Методу Байєса

Таким чином, модифікація однієї частини методу класифікації Байєса внесла зміни в фінальний прогноз і дозволила вірно класифікувати слова, специфічні для української мови. Відзначимо, що результат модифікованого методу наївної класифікації вищий,

ніж результат мультиноміального класифікатору. Для повноти аналізу проведемо статистичне дослідження ефективності модифікованого методу і визначимо середнє арифметичне ефективності за 10 тренувань алгоритму. Результат наведено у табл. 6.

Таблиця 6

Середнє арифметичне ефективності мультиноміального, наївного і модифікованого наївного методів класифікації

Спроба	Мультиноміальний	Наївний	Модифікований Наївний
1	0,862	0,845	0,915
2	0,794	0,816	0,901
3	0,884	0,810	0,830
4	0,826	0,873	0,915
5	0,869	0,788	0,845
6	0,783	0,845	0,887
7	0,873	0,802	0,920
8	0,836	0,760	0,887
9	0,921	0,901	0,901
10	0,894	0,830	0,830
Середнє	**0,854**	**0,827**	**0,883**

Середнє арифметичне ефективності класифікованого спаму листів українською мови модифікованого методу наївної класифікації Байєса складає 0,883. Це на 0,561 або 5,6% більше, ніж метод наївної класифікації до внесення змін. Отже, вдалось підвищити ефективність наївного класифікатору Байєса для задачі класифікації листів українською мовою на 5,6%, окрім цього, різниця з середнім арифметичним ефективності мультиноміального методу класифікації складає 0,289 або 2.9%.

Для того, щоб остаточно впевнитись в ефективності внесених змін, протестуємо оригінальний і модифікований методи наївного класифікатора на цільовому тестовому наборі повідомлень з великою кількістю апострофів. Було відібрано 24 повідомлення з словами де є один і більше апострофів і протестовано виключно на цій вибірці. Результат наведено на рис. 4.

```
Correct: 15
Incorrect: 9
Accuracy: 0.625
```

Рис. 4. Ефективність Методу Байєса на прикладах з апострофами

Як бачимо, початкова ефективність оригінального методу впала з 0.86 до 0.62. Наведемо результат тестування модифікованого методу на рис. 5.

```
Correct: 19
Incorrect: 5
Accuracy: 0.7916666666666666
```

Рис. 5. Ефективність Модифікованого Методу Байєса на прикладах з апострофами

Отже, внесену зміну в метод вважатимемо ефективною, адже вона дозволила підняти ефективність методу класифікації Байєса з 0.86 до 0.91 для української мови, що є суттєвою зміною.

Висновки

В рамках даного дослідження було проаналізовано ринок засобів машинного навчання для аналізу української мови, перш за все для виконання задачі класифікації. Були використані інструменти для роботи з мовою: ВЕСУМ (Великий Електронний Словник Української мови), інструмент лематизації (знаходження кореню слова) тощо. За допомогою комбінацій інструментів розроблено модифікований метод класифікації Байєса. Оскільки готового датасету класифікованих на «спам» і «не спам» листів українською мовою знайдено не було, тому датасет був підібраний з окремих прикладів. В процесі класифікації датасету було виявлено похибку стандартного nltk Python методу класифікації, розробленого з огляду на англійську мову, який некоректно знаходив границі слів з апострофами. Для покращення аналізу тексту українською мовою розроблено модифікований метод наївної класифікації Байєса з покращеною очисткою пунктуації тексту. В результаті було отримано кращий результат відсотку правильних класифікацій – 0.91 замість 0.86. Для повноцінного аналізу потрібно випробувати цю модифікацію на значно більшому датасеті (бажано в сотні тисяч повідомлень), що стане напрямком подальших досліджень.

Таким чином, досліджена та частково вирішена проблема аналізу текстів української мови за допомогою існуючих інструментів мови програмування Python, яка полягає в словах з апострофами.

Список літератури:

[1] *Simon Kemp*. Digital 2020: 3.8 billion people use social media / Simon Kemp URL: https://wearesocial.com/blog/2020/01/digital-2020-3-8-billion-people-use-social-media (дата звернення: 25.03.2021).

[2] *Joseph Johnson*. Number of sent and received e-mails per day worldwide from 2017 to 2025 / Johnson Joseph URL: https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/

[3] *Барсегян А.А.* Анализ данных и процессов: учеб. пособие / А.А. Барсегян, М.С. Куприянов, И.И. Холод, М.Д. Тесс, С.И. Елизаров. – 3-е изд., перераб. и доп. – Санкт-Петербург : БХВ-Петербург, 2009. – 512 с.

[4] *Yang Y.* A re-examination of text categorization methods / Y. Yang, X. Liu // Proc. of Int.ACM Conference on Research and Development in Information Retrieval (SIGIR-99), 1999.– P. 42-49.

[5] *Вагин В.Н.* Достоверный и правдоподобный вывод в интеллектуальных системах /В. Н. Вагин, Е. Ю. Головина, А. А. Загорянская, М. В. Фомина. – Москва : Физматлит, 2004. – 704 с.

[6] *Bradley P.* Carlin Bayes and Empirical Bayes Methods for Data Analysis, Second Edition 2nd Edition / P. Bradley Carlin, A. Thomas Louis. – 2000. – 440 p.

[7] *Joachims T.* Making large-scale SVM learning practical / T. Joachims // Advances in Kernel Methods Support Vector Learning. – MIT Press, 1999. – 218 p.

[8] *Sebastiani F.* Machine learning in automated text categorization / F. Sebastiani // ACM Comput. Surv. – March 2010. – Vol. 34, No. 1. – P. 1-47.

[9] *Jean Dos Santos*. Ham or Spam? SMS Text Classification Walkthrough / Santos Dos Jean URL: https://www.kaggle.com/jeandsantos/ham-or-spam-sms-text-classification-walkthrough

[10] *Richard S.* Sutton Reinforcement Learning, second edition: An Introduction (Adaptive Computation and Machine Learning series) / Sutton S. Richard Barto G. Andrew. – 2018. – 552 p.

Надійшла до редколегії 09.04.2021

УДК 004.8

DOi 10.30837/ bi.2021.1(96).05

Nataliia Golian[1], Iryna Afanasieva[2], Vira Golian [3], Dmytro Panchenko[4]

[1]associate professor of the department of Software Engineering,
NURE, Ukraine, nataliia.golian@nure.ua
[2]associate professor of the department of Software Engineering,
NURE, Ukraine, iryna.afanasieva@nure.ua
[3]associate professor of the department of Software Engineering,
NURE, Ukraine, vira.golan@nure.ua
[4]master's student of the department of Software Engineering,
Ukraine, dmytro.panchenko@nure.ua

APPLYING GRADIENT BOOSTING AS A STACKING ALGORITHM OVER BOTTLENECK FEATURES TO ACHIEVE HIGH IMAGE CLASSIFICATION ACCURACY

With the development of the Internet, making many images available online for analysis, object recognition software is gaining more and more attention from researchers. Factors are driving the development of computer vision today: mobile devices with built-in cameras, the availability of computing power, the availability of computer vision and analysis equipment, and new algorithms such as convolutional neural networks that take advantage of the power of hardware and software. The work is generally devoted to the consideration of the problem of image classification using convolutional neural networks. And in particular, one of the most popular and applied in practice machine learning algorithms – gradient boosting applied to the bottlenecks of deep convolutional neural networks. It also discusses three scenarios for applying gradient boosting to bottlenecks extracted from the last convolutional layer of the neural network. The essence of boosting, as well as of other ensembles of algorithms, is to collect one strong from several weak models. The general idea of boosting algorithms is to consistently apply predictors so that each subsequent model minimizes the error of the previous one. Gradient boosting works by sequentially adding new models to past models so that errors made by previous predictors are corrected.

ARTIFICIAL INTELLIGENCE, COMPUTER VISION, GRADIENT BOOSTING, IMAGE, MACHINE LEARNING, NEURAL NETWORK, PATTERN RECOGNITION

Голян Н., Афанасьева И., Голян В., Панченко Д. Применение градиентного бустинга в качестве алгоритма стекинга по узким местам для достижения высокой точности классификации изображений. С развитием Интернета, сделавшим многие изображения доступными онлайн для анализа, программное обеспечение для распознавания объектов привлекает все больше внимания исследователей. Факторы стимулируют развитие компьютерного зрения сегодня: мобильные устройства со встроенными камерами, доступность вычислительной мощности, доступность оборудования для компьютерного зрения и анализа, а также новые алгоритмы, такие как сверточные нейронные сети, которые используют аппаратные и программные возможности. Работа в целом посвящена рассмотрению проблемы классификации изображений с помощью сверточных нейронных сетей. И, в частности, одному из самых популярных и применяемых на практике алгоритмов машинного обучения – градиентного бустинга, применяемого к узким местам глубоких сверточных нейронных сетей. Также рассматриваются три сценария применения градиентного бустинга к узким местам, извлеченным из последнего сверточного слоя нейронной сети. Суть бустинга, равно как и других ансамблей алгоритмов, состоит в том, чтобы из нескольких слабых моделей собрать одну сильную. Общая идея алгоритмов бустинга – последовательно применять предикторы так, чтобы каждая последующая модель минимизировала ошибку предыдущей. Градиентный бустинг работает последовательно добавляя к прошлым моделям новые так, чтобы исправлялись ошибки, допущенные предыдущими предикторами.

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, ГРАДИЕНТНЫЙ БУСТИНГ, ИЗОБРАЖЕНИЕ, МАШИННОЕ ОБУЧЕНИЕ, НЕЙРОННАЯ СЕТЬ, РАСПОЗНАВАНИЕ ОБРАЗОВ

Голян Н., Афанасьєва І., Голян В., Панченко Д. Застосування градієнтного бустінгу в якості алгоритму стекінгу по вузьких місцях для досягнення високої точності класифікації зображень. З розвитком Інтернету, що зробив багато зображень доступними онлайн для аналізу, програмне забезпечення для розпізнавання об`єктів привертає все більше уваги дослідників. Фактори стимулюючі розвиток комп'ютерного зору сьогодні: мобільні пристрої з вбудованими камерами, доступність обчислювальної потужності, доступність обладнання для комп'ютерного зору і аналізу, а також нові алгоритми, такі як згорткові нейронні мережі, які використовують можливості обладнання і програмне забезпечення. Робота в цілому присвячена розгляду проблеми класифікації зображень за допомогою згорткових нейронних мереж. І, зокрема, одному з найпопулярніших і застосовуваних на практиці алгоритмів машинного навчання – градієнтному бустінгу, що застосовується до вузьких місць глибоких згорткових нейронних мереж. Також розглядаються три сценарії застосування градієнтного бустінгу до вузьких місць, добутих з останнього згорткового шару нейронної мережі. Суть бустінга, так само як і інших ансамблів алгоритмів, полягає в тому, щоб з кількох слабких моделей зібрати одну сильну. Загальна ідея алгоритмів бустінга – послідовно застосовувати предиктори так, щоб кожна наступна модель мінімізувала помилку попередньої. Градієнтний бустінг працює послідовно додаючи до минулих моделей нові так, щоб виправлялися помилки, допущені попередніми предикторами.

ШТУЧНИЙ ІНТЕЛЕКТ, КОМП'ЮТЕРНИЙ ЗІР, ГРАДІЄНТНИЙ БУСТІНГ, ЗОБРАЖЕННЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, РОЗПІЗНАВАННЯ ОБРАЗІВ

1. Introduction and preliminaries

During the last few years, computer vision (and image classification in particular) became one of the fastest developing areas in computer science. Different methods and algorithms to solve this problem were developed, and the most prominent of them is the usage of convolutional neural networks (CNN).

A neural network is a system of many neurons (processors). Separately, these processes are quite simple, but connected into a system, neurons perform very complex tasks of collecting information.

Among the main areas of application of neural networks are forecasting, decision making, pattern recognition, optimization, data analysis. Neural networks are at the heart of most modern speech recognition and synthesis systems, as well as image recognition and processing.

CNN is a class of deep neural networks which was originally created specifically for image processing and analysis. CNN is based on multilayer perceptron, but instead of using only fully-connected layers (which are prone to overfitting) it includes specific types of layers designed for extracting patterns from complex input image. Moreover, those layers allow CNNs to have less connections between neurons therefore drastically decreasing time required for network running.

The most basic CNNs are created using following building blocks:

– convolutional layers;
– pooling layers;
– fully-connected layers.

Convolutional layers consist of neurons, each of them covers specific part of the image (i.e. receptive field). Kernel (filter) of the layer defines how data from receptive field will be transformed into the output by its convolution with an input. It means that network will learn filters that activate when some specific type of feature is present at that part of the image.

Apart from receptive field width and height convolutional layers are also parametrized by their depth. Depth means number of convolutions (channels) in the layer which point to the same location of the input. These convolutions represent different features of the input area.

Pooling layers are used to reduce size of the input by partitioning it into a set of non-overlapping squares. For each such region single value is output, the most common is to use maximum of all values in this region (max pooling), but alternative approaches exist as well (e.g. average pooling). As the result, minor and unimportant for the problem at hand features are discarded, and important ones are kept.

Common architectures of CNNs use alternating sequence of convolutional and pooling layers (latter ones are inserted between successive convolutional layers). Such approach allows to separate more and more general features step by step.

After several convolutional and pooling layers usually go one or more fully-connected layers. They are identical to layers in regular (non-convolutional) neural networks and used for actual classification of input based on features extracted by convolutional layers.

2. Subject area

Neural networks derive their strength, firstly, from the parallelization of information processing and, secondly, from the ability to self-learn, that is, to create generalizations. The term generalization refers to the ability to obtain a reasonable result based on data that was not encountered in the learning process. These properties allow neural networks to solve complex (large-scale) problems that are considered intractable today. However, in practice, when working autonomously, neural networks cannot provide ready-made solutions. They need to be integrated into complex systems. In particular, a complex problem can be divided into a sequence of relatively simple ones, some of which can be solved by neural networks.

First attempts at applying deep neural networks to computer vision problems were made in 90s by Yann LeCun et al [1] – researchers achieved substantial results in recognizing handwritten numbers in images.

However, deep learning did not become a commonly used approach in computer vision until 2012, when it was used by Alex Krizhevskiy et al [2] to achieve state-of-the-art accuracy in ImageNet LSVRC-2010 competition. Krizhevskiy, Hinton and LeCun used convolutional neural networks to develop automatic feature extraction that is trained in an end-to-end manner with the classifier.

Main constraints that slowed down development of deep learning at that point were:

– absence of the dataset big enough to fit models with millions of trainable parameters;
– insufficient computational powers.

The identified problems were solved in the following three ways.

First of all, several huge datasets were created and labeled. The most well-known one is ImageNet [3] that consists of several millions of images labeled to a thousand classes (multiclass classification task).

Secondly, modern GPU architectures made researchers able to train deep models quite fast using hardware acceleration. Even specialized hardware optimized for tensor computations was developed [4].

It allowed researchers to train huge models and even apply hyperoptimization algorithms that build large architectures utilizing hundreds of GPUs at a time as shown by Zoph et al [5]. Network architecture developed by Zoph et al called NASNet contained 88.9 million parameters and achieved unprecedented at that time result of 0.960 top-5 accuracy on ImageNet. This example shows that nowadays computational complexity is not a problem. However, model speed still matters sometimes, and there

are different tradeoffs, such as tradeoff of accuracy versus inference speed (it is important for mobile applications and IoT devices using deep learning models for instance) and tradeoff of accuracy versus training speed (it can be important for baselining and exploratory analysis, when researcher wants to quickly assess how much accuracy can be expected from the given data at all). At last, there are cases when model speed does not matter at all, and accuracy is the only crucial metric.

Lastly, researchers developed a way to reuse weights of model trained on one dataset to another machine learning problem. This approach is called transfer learning.

Today, in most applications of deep learning, especially in the field of computer vision, training a deep neural network from scratch is impractical. The use of transfer training is currently the key to the top results in many problems. Transfer learning is an approach that takes a network that has already been trained on a larger dataset and then uses it as an initialization of the weight for further training. Typically, ImageNet acts as such a large dataset for preliminary training.

Transfer learning is based on the fact that convolutional neural network consists of two parts – convolutional part and fully-connected part.

Convolutional part acts essentially as a feature extractor that generates structured and highly separable features (sometimes called *bottleneck features*) from the unstructured input data – images in case of computer vision – by applying several stacked spatial transformations – convolutional and pooling layers.

Fully-connected layers in turn act as a classifier that models decision boundary for specific categories of objects.

So the idea of network-level transfer learning [6] is to reuse convolutional part of the network to apply weights pre-trained on a large dataset to the target dataset which is smaller (and in many cases even too small to train network from scratch at all).

Another important aspect of using convolutional part of a convolutional neural network as a feature extractor is studied in a paper by Zeiler & Fergus [7]. The fact is demonstrated that convolutional layers differ in terms of complexity and level of abstraction of structures recognized by these layers. It is shown that the deep layers of the network, close to the input, are studied low-level structures and simple geometric shapes, such as colored spots, lines, gradients. The closer the layer is to the bottleneck the more high-level features are detected by this layer. It means that latest layers of the deep convolutional neural network represent dataset-specific structured features (for example, for ImageNet last convolutional layer represents such features as cat ears, car wheel, etc.) that act as effective descriptors for solving not only a problem at hand for this particular dataset but a set of computer vision tasks for all datasets that are visually similar to training data.

Meanwhile, in the field of tabular data classification, where features are structured by nature, gradient boosting has established itself as one of the state-of-the-art classifiers outperforming other algorithms in various problems with complex feature spaces and non-linear decision boundaries [8]. The most popular and powerful implementations of gradient boosting over decision trees are LightGBM [9], xgboost [10] and CatBoost [11].

LightGBM is a fast, distributed, high-performance gradient boosting structure based on a decision tree algorithm used for ranking, classification, and many other machine learning tasks. Because it is based on decision tree algorithms, it splits the best-matched leaf of the tree, whereas other boosting algorithms divide the tree by depth or level rather than leaf. Thus, when grown on the same leaf in Light GBM, the leaf algorithm can reduce more losses than the layer-by-layer algorithm, and therefore results in much better accuracy, which can rarely be achieved by any of the existing boosting algorithms.

LightGBM uses the gradient one-sided sampling technique (GOSS) technique to filter data instances and find the split value. At the same time, XGBoost uses a pre-sorted algorithm and a histogram based algorithm to calculate the best split. Observations / Samples are examples here.

More specifically, a histogram-based algorithm breaks all data points for an object into discrete elements and uses them to find the split value of the histogram. It is more efficient than the pre-sorted algorithm in learning speed, which lists all possible split points on the pre-sorted feature values, but it still lags behind GOSS in terms of speed.

GOSS (Gradient One-Way Sampling) is a new sampling technique that downsamples based on gradients. Cases with small gradients are well trained (small learning error), and cases with large gradients are underscored. A naive approach is to discard instances with small gradients, focusing exclusively on instances with large gradients, but that would change the distribution of the data. GOSS keeps instances with large gradients by randomly sampling instances with small gradients.

Benefits of LightGBM:

– high efficiency and fast learning speed. LightGBM uses a histogram-based algorithm, that is, it combines continuous feature values into discrete cells, which speed up the learning process;

– lower memory usage. Replaces continuous values with discrete cells, which in turn results in lower memory usage;

– better accuracy than any other gain algorithm. It creates much more complex trees using a sheet rather than level approach, which is a major factor in achieving higher accuracy;

– compatible with large datasets. It is capable of performing equally well on large datasets with a significant reduction in training time.

Finally, ensembles are used either in computer vision or tabular data tasks. Ensembling is a technique of combining outputs of several base learners to reach prediction quality that is better than any of the standalone models is able to reach by itself. Many researchers have studied ensembling and different ways of combining classifiers starting with the bootstrap aggregation [12] and ending with stacking [13].

Stacking is an ensembling technique that allows to combine several models (in our case – classifiers) by using their predictions as a set of input features to the second level model (so called meta-classifier) [14].

The main idea of stacking is to use basic classifiers to get predictions and use them as features for some general algorithm. That is, the essence of stacking is the transformation of the original space of features of the problem into some new space, the points of the latter are the predictions of the basic algorithms.

First, a set of pairs of arbitrary subsets is selected from the training sample, and then, for each pair, it is necessary to train the basic algorithms on the first subset, and also predict the target variable for the second subset with them. In this case, the predicted values become objects of the new space. Stacking has been the primary way to ensemble the underlying algorithms of many machine learning competitions.

It is proposed to use gradient boosting as a classification algorithm for image classification task by applying it over bottleneck features of deep convolutional neural networks (in fact – stacking of bottleneck descriptors by gradient boosting). The main contribution and novelty is studying different scenarios of such stacking.

3. Methods

This article explores three scenarios of applying gradient boosting over bottleneck features extracted from the last convolutional layer of a CNN:

– a convolutional neural network is taken, previously trained on ImageNet. The bottleneck features of the frozen network are extracted for all images in the training and test parts part of the target dataset. After that the gradient boosting is fitted on the training set and evaluate it on the test set;

– a convolutional neural network is taken, pre-trained on ImageNet. The network is fine-tuned to the target dataset by fitting it on the whole training subset. After that, the bottleneck features of the fine-tuned network are extracted for all images in the training and test parts part of the target dataset, the gradient boosting isfited on the training set and evaluate it on the test set;

– several CNNs of different architectures are taken with pre-trained ImageNet weights. The bottleneck features are extracted from those networks, concatenated together and trained gradient boosting on the joint feature vector on the training set. After that, it is evaluated again on the test set.

The experiments are restricted by several assumptions. First of all, the scenario when several networks are fine-tuned to the target set is not tested, because in such case it is possible to apply classical stacking over models' predictions instead of learning from the bottleneck layers.

Only scenarios in which meta-classifier is trained on intermediate features extracted from a neural network are discussed. Also, the folds-in-folds stacking approach that prevents data leakage in the training set is not tested, because, firstly, out-of-fold model training and prediction are impractical in real world situation and, secondly, because there is no any guarantee that particular bottleneck features learnt by a neural network would have similar meaning at different folds which is a crucial assumption for using gradient boosting classifier after all.

The hypothesis is that structured features extracted by a convolutional neural network from raw input data (such as images) can serve as a suitable feature space.

4. Experiments

The experiments use data from the iMaterialist Challenge (Furniture) [15]. Dataset consists of 194 828 images in the training set and 6400 images in the validation set. Each image in dataset represents one of 128 classes of furniture and household items (e.g. chairs, beds, cookware, etc.), so it is a multiclass classification problem.

Examples of images (cookware) from dataset are presented in the Figure 1.

Fig. 1. Examples of images from iMaterialist Challenge (cookware)

Also taken for research, examples of images (chairs) from the dataset are presented in Figure 2.

Fig. 2. Examples of images from iMaterialist Challenge (chairs)

LightGBM used as a gradient boosting classifier implementation.

For testing purposes, the following CNN architectures must be configured, pre-trained on ImageNet:

– Xception [16];
– NASNet Large [5];
– DenseNet-201 [17];
– ResNet-152 [18].

A conventional convolutional layer handles the simultaneous correlation of adjacent points within one channel, spatial information (spatial information) and inter-channel information, because convolution is applied to all channels as a whole.

The Xception architecture assumes that these two types of information can be processed sequentially and without sacrificing network quality.

Xception decomposes regular convolution into spatial convolution (processes spatial correlation in terms of a single channel) and pointwise convolution (processes inter-channel correlation).

The initial depth-separable convolution is the depth convolution followed by the point convolution. Depth convolution is a channel-by-channel n×n spatial convolution. Point convolution is a 1×1 convolution to change a dimension.

The modified depth-separable convolution is a point convolution followed by a deep convolution. Modified deeply split convolution is used as the seed module in Xception (an extreme version of the seed module).

Differences:

– sequence. The original depth-separable convolutions first perform channel-by-channel spatial convolution, and then perform 1×1 convolution, as, for example, in TensorFlow. The modified depth-separable convolution first performs a 1×1 convolution followed by spatial wise convolution. This is denoted as not so important as, when used in a multi-layered setup, only slight differences appear at the beginning and end of all related starter modules;

– non-linearity. Non-linearity is observed in the entry-level starter module after the first operation. In Xception (modified depth-separable convolution), there is no intermediate non-linearity of ReLU [16].

NASNet-Large is a pretrained model on a subset of the ImageNet database. It belongs to the models of the NASNet architecture family. The NASNet architecture has been learned from data using a repetitive neural network, instead of fully developed by humans like other pretrained models [5].

DenseNet-201 is a convolutional neural network with 201 layers deep. It is possible to download a pretrained version of the network that has been trained on over a million images from the ImageNet database. A pretrained network can classify images into specific categories of objects such as keyboard, mouse, pencil, and others. As a result, the network studied the representations of functions for a fairly large range of images.

ResNet is a deep residual learning framework for image classification problem. Supports multiple architectural configurations to achieve the right balance between speed and quality. The ResNet architecture (with three of its implementations: ResNet-50, ResNet-101 and ResNet-152) has received successful results in ImageNet competitions. The basic idea used in these models, residual couplings, greatly improves gradient flow. This allows you to train much deeper models with tens and hundreds of layers.

For each experiment two metrics are calculated: accuracy and logloss.

Equation (1) displays the metric for calculating the accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Equation (2) shows the calculation for the metric relative to logloss.

$$H(p,q) = \sum_{x \in \S} p(x) \log q(x) \quad (2)$$

For performance reasons, gradient boosting is not trained on the full concatenated feature vector from four networks. Instead, PCA [19] is first applied to reduce the feature space to 2048 vectors, and only then is gradient boosting applied. Also, added flipped images to the train dataset as a mean of simple offline augmentation.

Principal component analysis (multivariate statistical analysis technology) is used to reduce the size of the feature space with minimal loss of useful information. The implication is that each principal component is associated with a certain proportion of the total variance of the original dataset (load). In turn, variance, which is a measure of data variability, can reflect the level of their information content. Principal component analysis is included in most analytical platforms and is widely used to reduce the dimension of input data at the stage of their preprocessing.

The main limitations of the principal component analysis are:

– impossibility of semantic interpretation of the components;

– the method can only work with continuous data.

The method is sometimes considered as part of a more general approach to data dimensionality reduction - factor analysis. In analytical platforms, it is the principal component method that is often practically implemented in factor analysis modules.

Networks of same architectures are used, fine-tuned to the dataset as a baseline.

The data obtained from the results of the experiments are presented in Table 1.

Table 1

Experiment data

Name	Accuracy	Logloss
Xception	0.8677	0.5299
LightGBM over ImageNet pre-trained Xception bottleneck features	0.7332	0.8920
LightGBM over fine-tuned Xception bottleneck features	0.6698	1.1906
NASNet Large	0.8677	0.4956
LightGBM over ImageNet pre-trained NASNet Large bottleneck features	0.7469	0.8879
LightGBM over fine-tuned NASNet Large bottleneck features	0.6565	1.2727

5. Results and discussion

Experiments show that stacking bottleneck features from a bunch of neural networks pre-trained on the ImageNet performs best in terms of logloss (i.e. outputs the most optimal probabilistic predictions of all tested models). In terms of accuracy it is only 0.77% worse than fine-tuned deep convolutional neural networks, however training gradient boosting on bottleneck features requires only a single inference from each network and fitting of boosting itself, while fine-tuning of a single neural network with Xception architecture takes around 30 GPU-hours and fine-tuning of a single NASNet takes around 40 GPU-hours on GTX 1080.

6. Conclusion

The paper considers the use of gradient boosting as a classification algorithm for the image classification problem. A study is presented to investigate scenarios for applying gradient boosting to bottlenecks extracted from the last CNN convolutional layer.

It can be concluded that gradient boosting on bottleneck features of the pre-trained networks performs well as a quick solution that does not require a lot of training. It is important to notice that boosting on ImageNet features works better than boosting on the features of already fine-tuned model despite the fact that ImageNet features are less optimized for this particular task. It can be explained by the fact that training convolutional neural network as a feature extractor and then gradient boosting as a classifier on a single dataset leads to a data leakage which presents itself in a form of overfitting. Though it is anyway impractical to use gradient boosting as a classifier when the network is already fine-tuned, since full convolutional network tuned in an end-to-end manner always performs better.

As a point for further research it is proposed to compare the considered approach stacking of bottleneck features with gradient boosting as a meta-classifier, with classical stacking approaches to determine optimal usage strategies.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] *LeCun Y., Boser B., Denker J., Henderson D., Howard R., Hubbard W., Jackel L.* Handwritten digit recognition with a back-propagation network // Advances in Neural Information Processing Systems 2, 1990. – P. 396-404.

[2] *Krizhevsky A., Sutskever I., Hinton G.* ImageNet classification with deep convolutional neural networks // Neural Information Processing Systems, 2012.

[3] *Deng J., Dong W., Socher R., Li L.-J., Li K., Fei-Fei L.* ImageNet: A Large-Scale Hierarchical Image Database // CVPR09, 2009.

[4] *Jouppi N.P., Young C., Patil N.* In-datacenter performance analysis of a tensor processing unit // CoRR, 2017. – http://arxiv.org/abs/1704.04760.

[5] *Zoph B., Vasudevan V., Shlens J., Le Q.V.* Learning transferable architectures for scalable image recognition // CoRR, 2017. – http://arxiv.org/abs/1707.07012.

[6] *Tan C., Sun F., Kong T., Zhang W., Yang C., Liu C.* A Survey on Deep Transfer Learning // 27th International Conference on Artificial Neural Networks, 2018.

[7] *Zeiler M.D., Fergus R.* Visualizing and Understanding Convolutional Networks // Computer Vision – ECCV 2014.

[8] *Friedman J.* Greedy function approximation: a gradient-boosting machine // Ann. Statist. 29, 2001. – P. 1189-1232.

[9] *Ke G., Meng Q., Finley T., Wang T., Chen W., Ma W., Ye O., Liu T.-Y.* LightGBM: A Highly Efficient Gradient Boosting Decision Tree // Advances in Neural Information Processing Systems 30, 2017. – P. 3146-3154.

[10] *Chen T., Guestrin C.* XGBoost: A Scalable Tree Boosting System // CoRR, 2016. – http://arxiv.org/abs/1603.02754.

[11] *Dorogush V., Ershov V., Gulin A.* CatBoost: gradient boosting with categorical features support // CoRR, 2018. – http://arxiv.org/abs/1810.11363.

[12] *Machova K., František B., Bednár P.* A Bagging Method using Decision Trees in the Role of Base Classifiers // Acta Polytechnica Hungarica 3, 2006.

[13] *Wolpert D. H.* Stacked generalization // Neural networks, 5(2), 1992. – P. 241-259.

[14] Marios Michailidis StackNet, StackNet Meta Modelling Framework, 2017. – https://github.com/kaz-Anova/StackNet.

[15] iMaterialist Competition 2018 https://sites.google.com/view/fgvc5/competitions/imaterialist.

[16] *Chollet F.* Xception: Deep Learning with Depthwise Separable Convolutions // CoRR, 2016. – http://arxiv.org/abs/1610.02357.

[17] *Huang G., Liu Z., Weinberger K.* Densely Connected Convolutional Networks // CoRR, 2016. – http://arxiv.org/abs/1608.06993.

[18] *He K., Zhang X., Ren S., Sun J.* Deep Residual Learning for Image Recognition // CoRR, 2015. – http://arxiv.org/abs/1512.03385.

[19] *Shlens J. A Tutorial on Principal Component Analysis // CoRR, 2014. – http://arxiv.org/abs/1404.1100.*

The article was delivered to editorial stuff on the 13.01.2021

УДК 681.3.06 DOi 10.30837/ bi.2021.1(96).06

Д.А. Золотарев

Кандидат физико-математических наук, г. Харьков, Украина, denis@zolotariov.org.ua,
ORCID: 0000-0003-4907-7810

ОБ ОДНОМ ПОДХОДЕ К АВТОМАТИЗИРОВАННОМУ ВЕРСИОНИРОВАНИЮ ПРОГРАММНОГО ОКРУЖЕНИЯ ВО ВРЕМЯ ЭТАПА РАЗРАБОТКИ

Рассмотрены, исследованы и разработаны теоретические и практические рекомендации относительно формирования механизма версионирования программного окружения времени разработки, который является автоматизированным, гибким и универсальным, и не зависит от конкретных языков программирования, компонентов программного обеспечения и других особенностей программного окружения. Объединяет в себе два подхода: универсальный по-файловый и индивидуальный для программных продуктов, поддерживающих внутренние механизмы создания простых и инкрементальных файловых копий данных. Приведен анализ элементов такого механизма и функциональная нагрузка каждого из них, обоснована его роль в общей структуре.

ВЕРСИОНИРОВАНИЕ СОСТОЯНИЯ, ПРОГРАММНОЕ ОКРУЖЕНИЯ, РАЗРАБОТКА ПО, API, BASH

Золотарьов Д.О. Про один підхід до автоматизованого версіонування програмного оточення під час етапу розробки. Розглянуті, досліджені та розроблені теоретичні та практичні рекомендації щодо формування механізму версіонування програмного оточення часу розробки, що є автоматизованим, гнучким та універсальним, й не залежить від конкретних мов програмування, компонентів програмного забезпечення та інших особливостей програмного оточення. Поєднує у собі два підходи: універсальний пофайловий та індивідуальний для програмних продуктів, що підтримують внутрішні механізми створення простих та інкрементальних файлових копій даних. Наведений аналіз складових елементів такого механізму та функціонального навантаження кожного з них, обгрунтована його роль у загальній структурі.

ВЕРСІОНУВАННЯ СТАНУ, ПРОГРАМНЕ ОТОЧЕННЯ, РОЗРОБКА ПЗ, API, BASH

Zolotariov D. One approach to automated versioning of the software environment during development stage. In article considered and investigated eheoretical and practical recommendations for the formation of the mechanism of versioning the software environment of development time, which is automated, flexible and universal, and does not depend on specific programming languages, software components and other features of the software environment, are considered, researched and developed. It combines two approaches: universal file-by-file and individual for software products that support internal mechanisms for creating simple and incremental file copies of data. The analysis of the constituent elements of such a mechanism and the functional load of each of them is given, its role in the general structure is substantiated.

STATE VERSIONING, SOFTWARE ENVIRONMENT, SOFTWARE DEVELOPMENT, API, BASH

Введение

Последние несколько лет в IT-индустрии ознаменовались стремительным ростом разнообразных технологий обработки данных, включая средства разработки [1], вследствие развития микросервисной архитектуры [2-8], сервисов распределенной обработки данных [9-13] и безсерверных вычислений, высоконагруженных медиа-сайтов и веб-сервисов [14], а также IoT [15-16]. Вместе с развитием уже существующих создаются все новые и новые программные решения. А значит, непрерывно изменяется и технологический стек разрабатываемого программного обеспечения для конечных пользователей.

Это программное обеспечение сегодня разделено на множество отдельных компонентов для хранения, обработки и выдачи данных, которые подбираются разработчиками в зависимости от ожидаемого результата. Полный набор этих компонентов и формирует программное окружение разработки конечного программного продукта.

При выборе нового программного решения в качестве такого компонента на плечи разработчиков и архитекторов ложится не только проверка его возможностей и определение соответствия конкретной задаче, но и его настройка для повышения эффективности работы через непрерывное экспериментирование с используемым инструментарием, которое идет рука об руку с разработкой целевого программного продукта.

Такое экспериментирование имеет своей целью поиск способов использования преимуществ этих компонентов и нивелирования их недостатков, и включает в себя предварительное, нагрузочное и интеграционное тестирования, позволяя определить эксплуатационные особенности компонента, которые нельзя выявить на статических этапах проектирования и разработки.

В процессе такого экспериментирования разработчиками производится множество изменений в компонентах, которые имеют файлы настроек либо сохраняют в процессе работы свое состояние (набор из пользовательских или само-сгенерированных в процессе работы данных, изменяющих реакцию компонента на внешние запросы). Эти изменения вносятся неконтролируемым образом до момента выявления стабильной рабочей конфигурации, удовлетворяющей требованиям проекта. И их история нигде не сохраняется.

Отсутствие истории изменения настроек или состояния компонента может приводить к критическим

ситуациям и, как следствие, значительному замедлению процесса разработки в случае их утери с обновлением компонента, сбоя на сервере или просто по ошибке. Их восстановление не гарантировано и может занимать очень длительное время из-за того, что разработчик вынужден в таком случае опираться только на свою память.

Безусловно, можно полагаться на полное резервное копирование сервера, предусмотренное всеми современными облачными технологиями и реальными мощностями, производимое обычно раз в неделю или реже. Но за этот период программное окружение и его параметры могут измениться несколько десятков и даже сотен раз.

Единственным альтернативным решением данной проблемы на текущий момент является добавление этих файлов под систему контроля версий (VCS). Но этот подход имеет ряд недостатков. Первый — в процессе создания программного продукта множество сторонних компонентов устанавливаются по мере необходимости и удаляются так же. Что затрудняет централизованное слежение за изменениями. Второй — их файлы настроек и состояния располагаются в различных не связанных между собой местах на сервере. И при значительном количестве таких компонентов может оказаться, что под контроль VCS нужно внести практически весь сервер, что не эффективно. И третий — фиксация изменений производится только вручную.

Поэтому актуальной задачей является разработка механизма хранения подробной истории изменений конфигурации программного окружения для серверов, использующихся во время разработки. Или другими словами — механизма версионирования программного окружения.

Целью этой статьи является разработка и обоснования теоретических и практических рекомендаций относительно формирования механизма версионирования программного окружения времени разработки. Этот механизм должен быть автоматизированным, максимально упрощенным для пользователей и универсальным, то есть минимально зависеть от конкретных компонентов программного обеспечения. А также нуждаться в минимуме сторонних приложений для своей работы, чтобы иметь максимальную скорость развертывания на чистом сервере и минимальную стоимость обслуживания.

Задача статьи – определить необходимые элементы такого механизма, предоставить для каждого из них анализ функциональной нагрузки и обосновать выбор инструментов для их решения.

1. Общий подход к построению механизма

Для решения поставленной задачи необходимо отметить, что механизм версионирования должен работать аналогично системам VCS, то есть создавать версии программного окружения в локальное и удаленное хранилища, позволяющие их просматривать и использовать. Делать это в автоматическом режиме без участия пользователя. Последнее обусловлено тем, что изменения, вносимые разработчиками, администраторами или специалистами DevOps в текущие настройки программного обеспечения сервера, например: nginx, systemd, environment, redis, ... — за период одного рабочего дня могут происходить множество раз. Следить за ними в ручном режиме не возможно и экономически не целесообразно из-за высокой стоимости времени разработчика.

Механизм должен обеспечивать создание версий для групп файлов и директорий с полным сохранением их структуры, начиная от корня, и прав доступа. Следует учесть, что в одну версию могут входить также несвязанные между собой директории и файлы, относящиеся к одному программному обеспечению. Примером может служить система доставки контента Apache Kafka, сохраняющая свои настройки и состояние в различных директориях [17].

Если программный компонент имеет встроенные инструменты создания копий состояния, как, например, СУБД MySQL [18] или Redis [19], в том числе инкрементальной копии, как, например, поисковая система Elasticsearch [20], то в механизме должна присутствовать возможность использовать эти особенности и преимущества.

Создаваемые версии практически всегда содержат конфиденциальные данные, поэтому хранение их во внешних хранилищах данных должно быть безопасным.

Таким образом, цели и подходы у существующих VCS и разрабатываемого механизма одинаковые. Отличие в способе генерации версии и автоматизации этого процесса. Системы контроля версий используются для файлов разрабатываемого программного обеспечения, которые находятся всегда в одной директории, и позволяют создание новой версии только вручную. Разрабатываемый же механизм — для программного окружения, данные которого могут находиться в файлах практически в любой директории сервера или рабочей станции, а также храниться только в оперативной памяти. Кроме того создание версии полностью автоматическое.

Поэтому системы контроля версий не являются конкурентами разрабатываемому механизму, хотя и решают схожую задачу.

С учетом выше сказанного, механизм версионирования должен обладать следующими характеристиками. Копирование должно производиться относительно корня диска, для каждого файла и его родителей должны сохраняться все связанные атрибуты inode: владелец, права доступа и прочие. Результат должен сжиматься современными алгоритмами сжатия (например, использовать хорошо себя зарекомендовавший алгоритм сжатия в реальном времени zstd [21]). А передаваемый на внешние сервисы хранения

данных архив должен быть зашифрован с использованием ассиметричных алгоритмов шифрования.

Причем на самом сервере должен храниться только публичный ключ. Эта мера позволит избежать каких-либо последствий с безопасностью в случае его компрометации. Приватный ключ должен храниться только у разработчика и администраторов. Для автоматизации восстановления версий возможно его хранение также на одном доверенном сервере с повышенными мерами безопасности.

Механизм создания версии состояния приложения может быть двух видов: универсальный и индивидуальный. Первый подразумевает простое пофайловое копирование. Второй — использование программ, специально предназначенных для создания резервной копии состояния определенного программного обеспечения, к которым относятся в частности: базы данных, поисковые системы, менеджеры очередей и т.д. — требующие выполнения определенной последовательности операций для создания копии состояния на диске, часто инкрементальной.

На сервере всегда должна оставаться как минимум одна незашифрованная, но сжатая, версия. Кроме восстановления она служит для сравнения с текущими файлами и директориями при создании новой.

Принципиальная блок-схема работы механизма версионирования настроек и состояния программного обеспечения сервера представлена на рис. 1 ниже.

Процесс создания версии состояния для приложения начинается с определения «универсальный – индивидуальный». В первом случае производится простое клонирование файлов и директорий и пофайловое сравнение для определения присутствия отличий. Во втором — выполняется набор команд, который отвечает за генерацию таких файлов и их клонирование, а также определение того, появились ли изменения с момента создания последней копии.

Если наличие изменений подтверждено, из клонированных inode создается единый архив, который затем зашифровывается и загружается во внешнее хранилище данных.

О результатах работы отправляются уведомления. После чего очищаются ненужные более локальные версии и клонированные файлы.

Рассмотрим далее более подробно каждый важный элемент механизма.

2. Сравнение версий

Если компоненты хранения состояния, имеющие возможность создания инкрементальных резервных копий данных, самостоятельно определяют наличие изменения с момента создания последней копии, то при простом пофайловом копировании рекомендуется использовать стандартные инструменты сравнения.

Рекомендуется для этой цели использовать в достаточной степени гибкий и быстрый компаратор из системы контроля версий Git, как самой распространенной и надежной [22]. Кроме того, можно использовать любой инструмент сравнения файлов и директорий, позволяющий решать эту задачу с достаточной скоростью и эффективностью, в том числе простая программа diff.

Другими словами, инструмент сравнения выбирается исходя из требования, чтобы нагрузка от его выполнения была не заметна пользователю, а время выполнения не было узким местом механизма версионирования. При этом он не должен иметь как ложных срабатываний, так и пропусков срабатывания при наличии изменений.

3. Взаимодействия с API компонентов

Для удобства взаимодействия с компонентами со встроенными инструментами создания копий состояния рекомендуется разработать API-инструментарий на языке оболочки bash или любом другом, поддерживающем сетевое взаимодействие и вызов внешних программ.

Такие компоненты либо имеют клиентские программы, реализующие свой протокол взаимодействия (например, PostgreSQL или Jenkins) или поддерживают REST-взаимодействие через HTTP-протокол (например, Elasticsearch).

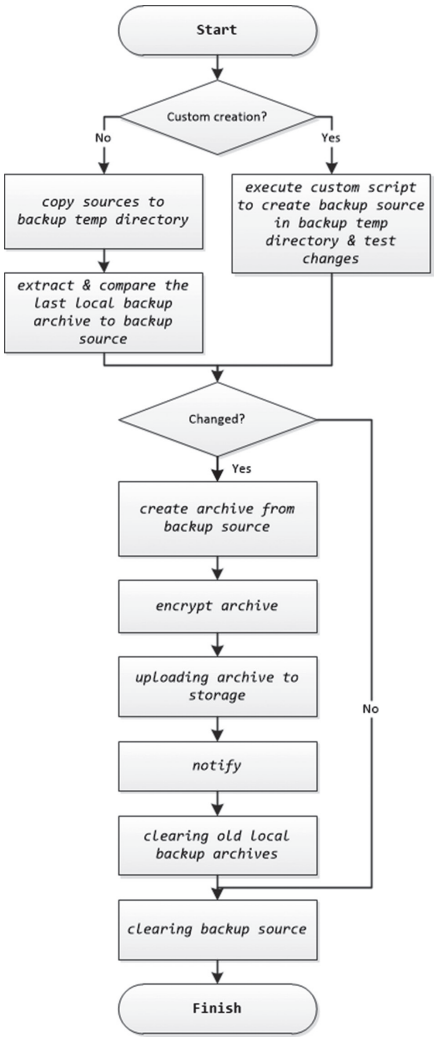


Рис. 1. Блок-схема механизма версионирования

Каждый отдельный модуль API-инструментария может иметь независимый от других внутренний механизм работы, при этом сохраняя общий интерфейс. А свои параметры получать либо полностью из командной строки при вызове, либо также из переменных окружения.

Несмотря на кажущуюся простоту, этот инструментарий должен быть достаточно гибким. Каждый программный продукт имеет свои значительные отличия в API-интерфейсе. Но даже в рамках одного типа REST каждый поддерживает его со своими особенностями — Elasticsearch, например, поддерживает стандарт REST в неполном виде [23]. Кроме того, версии одного продукта могут также иметь свои особенности API-механизма.

4. Хранение версий

Если в качестве локального хранилища может выступать просто директория, то удаленные хранилища достаточно разнообразны по способу организации и хранения.

Для удаленного хранения рекомендуется использовать системы холодного хранения данных. Желательно, несколько независимых.

В качестве таковых могут выступать сервера: Amazon S3, DigitalOcean, свои сервера или даже Dropbox или подобные файловые сервисы.

Последние могут также выполнять важную функцию автоматического копирования версий на несколько независимых носителей информации за счет автоматической фоновой синхронизации удаленного хранилища с локальным, где установлено приложение. Это может оказаться важным, если потребуется получить доступ к версии со стороннего компьютера.

5. Системные сообщения

Не все версии являются одинаково важными и для разработчиков, и администраторов или специалистов DevOps. Одни изменения влияют только на область задач конкретного разработчика, другие — могут оказывать широкое влияние на весь конечный продукт или его часть, за которую ответственны несколько специалистов.

Для разделения по степени важности, а также для удобства и универсальности отправки сообщений, следует использовать несколько каналов связи. Рекомендуется использовать как минимум два:

- только для разработчика,
- для всей группы, участвующей в разработке конечного продукта.

Блок-схема механизма представлена на рис. 2.

Его задача — отправлять персонализированные (только текущему разработчику) и групповые сообщения всем заинтересованным в изменениях в работе сервера лицам. Каналами отправки сообщений могут выступать:

- email,
- Slack, Telegram,
- и прочие.

Электронная почта должна быть обязательно включена для гарантированной доставки, так как ее работа проста, давно отлажена и предсказуема. Кроме того, механизм оповещений должен отвечать и современным требованиям к используемым технологиям — должны быть подключены все необходимые в современной разработке месенджеры.

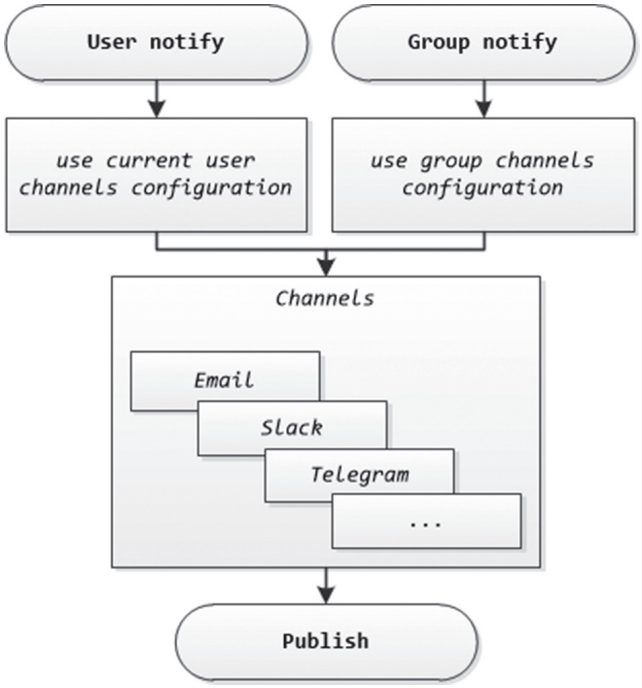


Рис. 2. Блок-схема отправки сообщений

Для группового оповещения лучше всего подходит канал в Slack, но также может быть использована группа в Telegram с использованием Telegram-бота.

Настройки каждого канала рекомендуется определять в переменных окружения. Для перечисленных выше это будут: EMAIL, TELEGRAM_BOT_TOKEN и TELEGRAM_CHAT_ID — определяющие токен Telegram-бота и id чата предназначения, а также SLACK_WEBHOOK_URL — хранящая URL к приложению для публикации сообщения в Slack.

Для удобства отправки одного сообщения в несколько каналов рекомендуется использовать единый центральный механизм (например, bash-скрипт), принимающий в качестве аргументов сообщение и список каналов, по которым оно должно быть отправлено.

Для ускорения работы этого инструментария в качестве центрального узла возможно использовать отдельный сервер на основе менеджера очередей, который принимает сообщение и список каналов связи с адресами, и далее отправляет это сообщение в каждый из них. Таким образом, можно решить проблему задержек при публикации сообщений в достаточно медленные внутренние сервисы или же внешние с негарантированной связью.

Кроме отправки сообщений в процессе создания новой версии программного компонента, этот инструментарий может быть использован также в любых других случаях, требующих внимания разработчика, администраторов или специалистов DevOps.

6. Особенности механизма

Описанное выше аналогичное работе системе контроля версий дополнение дает возможность быстро вернуться к последней или любой более ранней сохраненной копии данных при сбое в работе сервера или при неудачном его обновлении. При восстановлении оно позволяет быстро воссоздать такой же сервер даже в случае недоступности его полной резервной копии.

Применительно к дублированию — такие версии позволяют развернуть сервер на базе новых версий компонентов программного обеспечения с сохранением всех его настроек и состояния. В том числе и без создания полной копии оригинала, т.е. без нагрузки на исходный сервер.

Что может быть полезно, в том числе, при создании тестовой версии production-сервера или обновлении stage-сервера.

Также такой подход к развертыванию среды разработки через использование версий позволяет решить задачу создания нескольких вариантов протестированного окружения. Что минимизирует сбои в работе, а значит, предотвращает непроизводственные финансовые и временные расходы.

7. Практическая реализация

Для проверки возможностей предложенного механизма, он был реализован в виде пакета подпрограмм на языке оболочки bash. В качестве инструмента сравнения и определения наличия изменений в версиях использован компаратор git, в качестве инструмента сжатия — tar с алгоритмом сжатия «xz», в качестве инструмента защиты — gpg с ассиметричным шифрованием.

В качестве локального хранилища — директория «/backups», удаленного — директория в файловом сервисе Dropbox, взаимодействие с API которого осуществлялось при помощи разработанного на языке bash инструментария на базе curl согласно документации сервиса.

В качестве инструмента автоматизации были использованы задачи в cron согласно правилу — одна задача на один программный компонент. Добавление задач было реализовано через универсальную bash-подпрограмму, принимающую в качестве своих аргументов название программного компонента и набор директорий для версионирования. Последнее — только если использовалось простое клонирование файлов.

Версионирование проводилось для следующих программных компонентов: файл переменных окружения «/etc/environment», конфигурации веб-сервера nginx. А также данные из: NoSql СУБД Redis, СУБД MySQL, поисковой системы Elasticsearch. Для первых использовался простой подход файлового клонирования, для вторых — специально разработанный инструментарий на языке оболочки bash, использующий индивидуальные встроенные возможности в каждый из компонентов хранения данных основанных на [18-20].

Для каждого компонента использовалась отдельная директория в локальном и удаленном хранилищах версий. Каждая версия в своем названии имела временную метку с точностью до секунд для легкого определения хронологии.

В качестве каналов системных сообщений использовались email и канал в Telegram. В первом случае использовался пакет утилит mailutils, во втором — специально разработанный инструментарий на языке оболочки bash и curl для взаимодействия с REST-API Telegram согласно документации месенджера.

В процессе полугодовой непрерывной эксплуатации при разработке приложений на PHP/Laravel и Python на низко-производительном сервере разработанный механизм автоматического версионирования программного окружения показал себя как удовлетворяющий требованиям скорости и надежности. Его работа была не заметна пользователям, при этом он не имел ложных срабатываний или пропусков срабатывания при наличии изменений.

Выводы

В работе впервые разработаны и обоснованы теоретические и практические рекомендации относительно формирования механизма версионирования программного окружения во время этапа разработки программного обеспечения, создающего историю изменений программных компонентов.

Этот механизм является автоматизированным, гибким и универсальным, и не зависит от конкретных языков программирования, компонентов программного обеспечения и других особенностей программного окружения.

Доказано, что, несмотря на то, что цели и подходы у систем контроля версий и предложенного механизма одинаковые, они имеют существенные отличия в применении и поэтому не являются конкурентами.

Показано, что такой механизм должен состоять из следующих функциональных элементов: клонирование файлов настроек и состояния, сравнения их с последней версией, сжатия, шифрование и загрузка во внешние хранилища версий, а также отправка уведомлений о появлении новой версии.

Он объединяет два подхода: универсальный пофайловый и индивидуальный для программных продуктов, поддерживающих внутренние механизмы создания простых и инкрементальных файловых копий данных.

Каждый элемент детально описан, показана его функциональная нагрузка и обоснована его роль в общей структуре.

Приведен анализ практической реализации данного механизма для конкретных языков программирования и компонентов программного окружения, разработанного на языке оболочки bash. Отмечено, что в такой реализации его выполнение не заметно пользователям, не имеет ложных срабатываний или пропусков срабатывания при наличии изменений. Кроме того он не нуждается в сторонних приложениях для своей работы.

Показано, что отсутствие истории изменения настроек или состояния компонентов программного обеспечения может приводить к критическим ситуациям и, как следствие, значительному замедлению процесса разработки, а также то, что в результате применения предложенного механизма такие ситуации исчезают. Что ведет к таким преимуществам как усовершенствование процесса разработки, облегчение труда разработчика и сокращение непроизводственных потерь в условиях частого изменения программного окружения.

Список литературы:

[1] *Lee, CY.* Temporal Correlation Analysis of Programming Language Popularity // J. Korean Phys. Soc. 2019. Vol. 75, P. 755–763. https://doi.org/10.3938/jkps.75.755

[2] *da Silva H.H.S., de F. Carneiro G., Monteiro M.P.* An Experience Report from the Migration of Legacy Software Systems to Microservice Based Architecture // 16th International Conference on Information Technology-New Generations (ITNG 2019). Advances in Intelligent Systems and Computing. 2019. Vol 800. https://doi.org/10.1007/978-3-030-14070-0_26

[3] *Rademacher F., Sachweh S., Zündorf A.* A Modeling Method for Systematic Architecture Reconstruction of Microservice-Based Software Systems // Enterprise, Business-Process and Information Systems Modeling. BPMDS 2020, EMMSAD 2020. Lecture Notes in Business Information Processing. 2020. Vol 387. https://doi.org/10.1007/978-3-030-49418-6_21

[4] *Levcovitz, A., Terra, R., Valente, M.T.* Towards a technique for extracting microservices from monolithic enterprise systems // Proceedings of VEM'15. 2015. P. 97–104.

[5] *Munari S., Valle S., Vardanega T.* Microservice-Based Agile Architectures: An Opportunity for Specialized Niche Technologies // Reliable Software Technologies – Ada-Europe 2018. Ada-Europe 2018. Lecture Notes in Computer Science. V. Vol 10873. https://doi.org/10.1007/978-3-319-92432-8_10

[6] *Bucchiarone, A., Dragoni, N., Dustdar, S., et al.* From monolithic to microservices: an experience report from the banking domain // IEEE Softw. 2018. Vol. 35(3), P. 50–55.

[7] *Sorgalla J., Sachweh S., Zündorf A.* Exploring the Microservice Development Process in Small and Medium-Sized Organizations // Product-Focused Software Process Improvement. PROFES 2020. Lecture Notes in Computer Science. 2020. Vol 12562. https://doi.org/10.1007/978-3-030-64148-1_28

[8] *Zolotariov, D.* The distributed system of automated computing based on cloud infrastructure // Innovative Technologies and Scientific Solutions for Industries. 2020. No. 4 (14), P. 47–55. https://doi.org/10.30837/ITSSI.2020.14.047

[9] *Zolotariov, D.* The mechanism for creation of event-driven applications based on Wolfram Mathematica and Apache Kafka // Innovative Technologies and Scientific Solutions for Industries. 2021. No. 1 (15), P. 53–58. https://doi.org/10.30837/ITSSI.2021.15.053

[10] *Mahapatra, T.* Composing high-level stream processing pipelines // Journal of Big Data. 2020. Vol. 7, No. 81. https://doi.org/10.1186/s40537-020-00353-2

[11] *Jung, S., Kim, Y. & Hwang, E.* Real-time car tracking system based on surveillance videos // EURASIP Journal on Image and Video Processing. 2018. Vol. 2018, No. 133. https://doi.org/10.1186/s13640-018-0374-7

[12] *Ismail, A., Truong, HL. & Kastner, W.* Manufacturing process data analysis pipelines: a requirements analysis and survey // Journal of Big Data. 2019. Vol. 6, No. 1. https://doi.org/10.1186/s40537-018-0162-3

[13] *Kim, YK., Kim, Y. & Jeong, CS.* RIDE: real-time massive image processing platform on distributed environment // EURASIP Journal on Image and Video Processing. 2018. Vol. 2018, No. 39. https://doi.org/10.1186/s13640-018-0279-5

[14] *Kolajo, T., Daramola, O. & Adebiyi, A.* Big data stream analysis: a systematic literature review // Journal of Big Data. 2019. Vol. 6, No. 47. https://doi.org/10.1186/s40537-019-0210-7

[15] *Nasiri, H., Nasehi, S. & Goudarzi, M.* Evaluation of distributed stream processing frameworks for IoT applications in Smart Cities // Journal of Big Data. 2019. Vol. 6, No. 52. https://doi.org/10.1186/s40537-019-0215-2

[16] *Ed-daoudy, A., Maalmi, K.* A new Internet of Things architecture for real-time prediction of various diseases using machine learning on big data environment // Journal of Big Data. 2019. Vol. 6, No. 104. https://doi.org/10.1186/s40537-019-0271-7

[17] Kafka – Apache [Електронний ресурс] // Apache. URL: https://kafka.apache.org/documentation/#configuration. – Назва з екрана.

[18] Backup and Recovery Types – MySQL 8.0 Reference Manual [Електронний ресурс] // MySQL. URL: https://dev.mysql.com/doc/refman/8.0/en/backup-types.html. – Назва з екрана.

[19] Redis Persistence – Redis [Електронний ресурс] // Redis. URL: https://redis.io/topics/persistence. – Назва з екрана.

[20] Snapshot and restore – Elasticsearch Guide [Електронний ресурс] // Elastic. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/snapshot-restore.html. – Назва з екрана.

[21] Compression Comparison Benchmarks: zstd vs brotli vs pigz vs bzip2 vs xz etc – Centmin Mod Community Support Forums [Електронний ресурс] // Sysadmin. URL: https://community.centminmod.com/threads/compression-comparison-benchmarks-zstd-vs-brotli-vs-pigz-vs-bzip2-vs-xz-etc.12764/. – Назва з екрана.

[22] 2021 Version Control Software Comparison: SVN, Git, Mercurial – Time doctor [Електронний ресурс] // Time doctor. URL: https://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/. – Назва з екрана.

[23] Update API – Elasticsearch Reference [Електронний ресурс] // Elastic. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-update.html. – Назва з екрана.

Поступила в редколлегию 10.02.2021

УДК 004.627 DOi 10.30837/ bi.2021.1(96).07

О.С. Валлас[1], О.В. Вечур[2]

[1]ХНУРЕ, м. Харків, Україна, oleh.vallas@nure.ua, ORCID iD: 0000-0003-4043-3946

[2]ХНУРЕ, м. Харків, Україна, alexander.vechur@nure.ua, ORCID iD: 0000-0001-9605-1475

МЕТОДИ ПОШУКУ ТА КОДУВАННЯ СХОЖИХ ПОСЛІДОВНОСТЕЙ ДАНИХ В АЛГОРИТМАХ СТИСНЕННЯ ДАНИХ БЕЗ ВТРАТ

Розглянуто методи пошуку та кодування схожих послідовностей даних, та їх використання для покращення алгоритмів стиснення даних без втрат. Досліджено сучасні підходи до пошуку послідовностей з неточним збігом – тривіальні та евристичні методи, індексні методи та методи, що базуються на N-грамах. Розглянуто підходи кодування відмінностей з використанням відстані Левенштейна та Геммінга. Запропонована розширена структура алгоритму стиснення даних. Комбінації вищезазначених методів у складі запропонованої структури було протестовано на двох датасетах – датасеті англійського тексту «enwik8» та комбінованому датасеті «Silesia Corpus». При тестування оцінювались ступінь стиснення, швидкість кодування та декодування, та загальний баланс. У результаті було розроблено нову структуру алгоритмів стиснення даних та виявлено найбільш ефективні комбінації методів для компресії різних типів даних.

СТИСНЕННЯ ДАНИХ БЕЗ ВТРАТ, СХОЖІ ПОСЛІДОВНОСТІ ДАНИХ, ІНДЕКСНІ МЕТОДИ, N-ГРАМИ, ВІДСТАНЬ ЛЕВЕНШТЕЙНА, ВІДСТАНЬ ГЕММІНГА, КОДУВАННЯ ВІДМІННОСТЕЙ

Валлас О.С., Вечур О.В. Методы поиска и кодирования похожих последовательностей данных в алгоритмах сжатия данных без потерь. Рассмотрены методы поиска и кодирования похожих последовательностей данных и их использование для улучшения алгоритмов сжатия данных без потерь. Исследованы современные подходы к поиску последовательностей с неточным совпадением – тривиальные и эвристические методы, индексные методы и методы, основанные на N-граммах. Рассмотрены подходы кодирования различий с использованием расстояния Левенштейна и Хэмминга. Предложена расширенная структура алгоритма сжатия данных. Комбинации вышеупомянутых методов в составе предложенной структуры были протестированы на двух датасетах – датасете английского текста «enwik8» и комбинированном датасете «Silesia Corpus». При тестировании оценивались степень сжатия, скорость кодирования и декодирования, а также общий баланс. В результате была разработана новая структура алгоритмов сжатия данных, и выявлены наиболее эффективные комбинации методов для компрессии различных типов данных.

СЖАТИЕ ДАННЫХ БЕЗ ПОТЕРЬ, ПОХОЖИЕ ПОСЛЕДОВАТЕЛЬНОСТИ ДАННЫХ, ИНДЕКСНЫЕ МЕТОДЫ, N-ГРАММЫ, РАССТОЯНИЕ ЛЕВЕНШТЕЙНА, РАССТОЯНИЕ ХЭММИНГА, КОДИРОВАНИЕ РАЗЛИЧИЙ

Vallas O.S., Vechur O.V. Methods of search and encoding similar data sequences in lossless data compression algorithms. Considered methods for searching and encoding approximate string matchings and their application to lossless data compression algorithms improvement. Studied modern approaches to the search for similar data sequences – trivial and heuristic methods, indexing methods and methods based on N-grams. Considered approaches of encoding differences using Levenshtein and Hamming distances. Proposed an extended structure of the data compression algorithm. Combinations of the above methods as part of the proposed structure were tested on two datasets – the English text dataset «enwik8» and the combined dataset «Silesia Corpus». During testing, ccompression ratio, encoding and decoding speed and overall balance were evaluated. As a result, a new structure of data compression algorithms was developed and the most effective combinations of methods for compression were identified for different data types.

LOSSLESS DATA COMPRESSION, APPROXIMATE STRING MATCHING, INDEXING METHODS, N-GRAMS, LEVENSHTEIN DISTANCE, HAMMING DISTANCE, DIFFERENCE ENCODING

Вступ

Стиснення даних – одне з найстаріших і фундаментальних напрямків у всій сфері інформаційних технологій. На поточний момент обсяг даних в світі оцінюють більш ніж 10^{18} байт і щороку ця кількість збільшується на 30%. Ці дані представляють собою найрізноманітнішу інформацію – текст на різних мовах, зображення, мультимедіа, відеоролики, документи, файли вихідного коду і безліч інших поширених або вузько спеціалізованих форматів. І практично для всіх способів зберігання і типів даних використовується стиснення. Стиснення даних дозволяє значно скоротити обсяг пам'яті, займаний одними або іншими даними і використовується повсюдно у всіх без винятку сферах діяльності людини – від побутових приладів до космічних апаратів.

Усі підходи та алгоритми стиснення даних можна поділити на дві категорії – стиснення даних без втрат та стиснення з втратами. Стиснення без втрат (англ. lossless compression) — метод стиснення даних, при використанні якого закодована інформація може бути повністю відновлена зі стиснутих даних. Навпаки, стиснення з втратами дозволяє лише відновлення даних, які є тільки наближенням до початкових даних. Найчастіше використовується саме стиснення без втрат, адже користувачеві потрібно отримати інформацію саме в такому вигляді, якою вона була до стиснення. Стиснення з втратами

використовується в основному в медіафайлах – зображеннях, аудіо, відео тощо. Тому що в цих файлах часто важливо передати лише загальний огляд і не треба відновлювати інформацію з точністю до окремого пікселю або ноти. В цій роботі будуть розглянуть підходи саме стиснення даних без втрат.

На сьогодні існує дуже багато різних підходів для роботи з даними. Різноманітні алгоритми і структури даних дозволяють робити багатопрофільну обробку та пришвидшити різні операції роботи з даними. Сучасний напрямок обробки природньої мови (NLP) пропонує різні підходи аналізу даних на основі аналізу реальної мови людини. Популярні області аналізу даних та машинного навчання пропонують адаптивні методи будь-якої обробки даних на основі здатності комп'ютерів ітеративно покращувати результат, «навчаючись» на попередніх результатах. Усі ці сфери дають майже нескінченну множину комбінацій, які потенційно можуть бути використані для стиснення даних.

Метою даної роботи є дослідження методів та алгоритмів пошуку схожих послідовностей даних та можливостей застосування цих методів для покращення алгоритмів стиснення даних без втрат. Будуть проаналізовані різні комбінації цих методів у складі існуючих алгоритмів стиснення даних, зокрема алгоритмів родини LZ* [1]. Також будуть розглянуть підходи до оптимального кодування знайдених послідовностей для покращення ефективності стиснення і швидкості кодування та декодування даних. Для аналізу та порівняння результатів буде розроблена метрика та зібрані репрезентативні датасети, представлені різними типами даних. Для автоматизації порівняння буде розроблено набір програмних методів роботи з різними комбінаціями алгоритмів.

1. Базова структура алгоритму стиснення даних без втрат

Практично усі алгоритми стиснення даних без втрат базуються на простій загальній схемі, запропонованій [1], яка отримала назву родина алгоритмів LZ77». Ця схема складається з декількох послідовних етапів, зображених на рисунку 1 (зліва). На першому кроці відбувається пошук закономірностей в даних, далі ці закономірності кодуються так, щоб їх можна було відновити на етапі декодування. В кінці до отриманої послідовності застосовуються додаткові оптимізації, найчастіше – побайтове оптимальне кодування, в результаті якого виходить стиснена послідовність даних. В якості алгоритму оптимального кодування зазвичай вибирають кодування Гаффмана [2] або арифметичне кодування. Обидва алгоритми кодують байти оптимальним способом, тому цей крок не представляє інтересу і використовується однаково практично у всіх алгоритмах стиснення. Основна частина стиснення зосереджена на етапі 2 і 3, а саме в пошуку закономірностей в даних і їх кодуванні. Автори оригінального алгоритму LZ77 пропонують досить примітивні методи пошуку закономірностей – простий пошук однакових послідовностей в деякому фіксованому вікні даних. Знайдені повторення кодуються парами (відстань; довжина), а нові послідовності просто записуються копіюванням.

Рис. 1. Базова (зліва) та розширена (справа) схеми алгоритмів стиснення даних без втрат

Пізніше було запропоновано велику кількість більш розвинутих технік, що поліпшують ефективність оригінального алгоритму LZ77 – розбиття даних на незалежні блоки [3], оптимальне арифметичне кодування отриманої послідовності [4], застосування дельта-фільтрів [5], динамічна оптимізація параметрів пошуку. Сучасна версія алгоритму LZ77 і його ефективні оптимізація розглядаються в [6]. Однак всі ці алгоритми засновані на пошуку і кодування точних збігів послідовностей. У даній роботі ж розглядаються більш складні підходи з використанням кодувань схожих послідовностей, які не обов'язково повністю співпадають.

Вперше використання часткового співпадіння рядків для кодування даних було запропоновано в публікації «Universal data compression based on approximate string matching» [7]. Автори доповнюють вищевказану схему, пропонуючи шукати закономірності в даних шляхом реалізації динамічного словника з підтримкою неточного пошуку. У статті наведено доказ того, що при досить великих вхідних даних, алгоритм завжди сходиться до теоретично оптимального ступеня стиснення. Однак, в даній статті розглядається стиснення з втратами (lossy compression), тому в запропонованій схемі відсутній етап кодування відмінностей між послідовностями.

На основі проведеного аналізу, була розроблена конкретизація оригінальної схеми алгоритму стиснення даних, яка зображена на рисунку 1 (справа). Розроблена схема акцентує увагу на двох основних етапах – пошуку послідовностей з неточним збігом і ефективне кодування відмінностей. В якості останнього етапу було обрано кодування Гаффмана, яке дозволяє оптимально закодувати байти отриманої на попередніх етапах послідовності і має безліч застосувань і ефективних імплементацій у різних алгоритмах стиснення без втрат.

2. Методи пошуку схожих послідовностей даних

У публікації [7] розглядаються два основні підходи до пошуку схожих послідовностей – онлайн і офлайн підхід. Онлайн підхід полягає в ітеративній побудові словника по ходу опрацювання даних. Кодування нових послідовностей відбувається паралельно з побудуванням словника.

Офлайн підхід, на відміну від онлайн підходу, обробляє усі дані заздалегідь і будує так звані індекси схожості між послідовностями тексту. Після того, як індекси побудовані, відбувається кодування усього тексту з фіксованим словником. Онлайн підхід працює швидше і використовує менше пам'яті, але поступається офлайн підходу за ефективністю стиснення.

У роботі «A Comparison of Approximate String Matching Algorithms» [8] були розглянуті неасимптотичні оптимізації, які пришвидшують тривіальний підхід. Асимптотична складність тривіального алгоритму, який обчислює відстань редагування з усіма послідовностями зі словника дорівнює $O(m*n)$, де m – загальна довжина усіх слів словника, а n – довжина слова, яке шукають. Представлені у цій публікації алгоритми асимптотично теж працюють за час, залежний від розміру словника, але на практиці показують значне прискорення. Основна ідея таких алгоритмів полягає у тому, що можна робити пошук у 2 етапи. На першому етапі проводиться швидка статистична перевірка кандидатів, а повна відстань редагування рахується лише на тих кандидатах, які пройшли перший етап. Завдяки тому, що більшість кандидатів значно відрізняються від патерну, повільна перевірка буде обчислена відносно невелику кількість раз. Основний же об'єм словника буде опрацьовано лише на етапі швидкої перевірки. Автори роботи порівняли ефективність основних евристичних алгоритмів на словнику розміром 100000 англійських слів. Результат зображений на рис. 2.

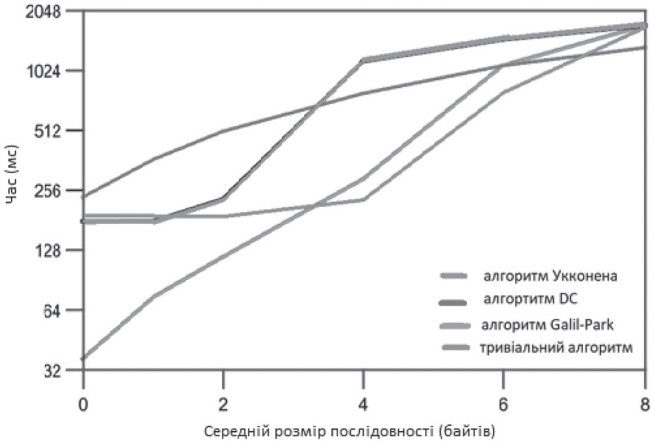


Рис. 2. Ефективність евристичних оптимізацій тривіального пошуку

Алгоритм «Galil-Park» [9] базується на монотонності таблиці пошуку і використовую так звані «посилання переходів», щоб швидко пропускати оброблені раніше послідовності.

Представлений у роботі [10] алгоритм «DC» використовує частоту зустрічання символів у словах словника і відфільтровує слова, частотний розподіл яких сильно відрізняється від шуканого.

Еско Укконен у своїй роботі [11] відзначив, що частотний підхід має значний недолік – він ніяк не враховує відносний порядок слів і має однакове значення для усіх перестановок заданого рядка. Він запропонував разом із частотним розподілом враховувати спільні послідовності, що забезпечило збіг не тільки множини символів, а і їх порядку.

За результатами тестування, проведеного авторами роботи [8] найбільш ефективним серед евристичних алгоритмів виявився метод Укконена, описаний у роботі [11].

Альтернативний підхід до пошуку схожих послідовностей у наборі даних був запропонований у публікації «Approximate string matching using n-grams technique» [12]. Автори роботи погоджуються з тим, що обчислення відстані редагування між поточною послідовністю та усіма іншими послідовностями, наявними у словнику, працює дуже повільно та пропонують поетапний алгоритм, який базується на так званих n-грамах. N-грама – це неперервна підстрока з N послідовних символів, які є складовими частинами рядка. Алгоритм складається з декількох етапів, на кожному з яких залишаються лише ті слова, які містять необхідну кількість n-грам, співпадаючих з n-грамами оброблюваного рядка. Ключовою ідеєю алгоритму є те, що на кожному етапі простір потенційних кандидатів значно зменшується, тому що аби не бути відфільтрованим, кандидат повинен містити не тільки необхідну кількість n-грам поточної довжини, але і всі менші n-грами. Автори стверджують, що достатнього всього 4 етапи навіть для послідовностей великої довжини і експериментально показують, що порівняння відбувається за лінійний час, на відміну від квадратичного у методі обчислення відстані підходом динамічного програмування.

У роботі «Indexing Methods for Approximate Dictionary Searching: Comparative Analysis»[13] представлений детальний аналіз більш сучасних та ефективних методів пошуку схожих послідовностей, які базуються на індексних структурах даних. Завдяки ефективним структурам даних пошуку строк, до яких відносяться суфіксні / префіксні структури, дерева пошуку, строкові автомати, та інші, вирішується основна проблема розглянутих вище методів – необхідність аналізувати увесь словник для пошуку кандидатів. Пошук інформації у індексних структурах може бути виконаний за суб-лінійний час, тобто значно швидше, ніж переглядати усі входження у словник. Але для того, щоб мати можливість робити

запити до таких структур, їх потрібно спочатку побудувати і підтримувати. Деякі методи, зокрема робота [14], пропонують заздалегідь побудувати структури даних на усьому тексті, але це не підходить до задачі компресії, адже під час стиснення необхідно шукати схожі послідовності лише на опрацьовану префіксу даних, а не на усьому тексті. Тому для цієї задачі необхідні «онлайн» індексні структури, тобто ті, які можуть підтримувати два типи запитів:

– додати нову послідовність у структуру;

– знайти усі послідовності, відстань редагування яких до заданої є не більшою за k.

Індексні методи можна поділити на дві категорії – префіксні дерева та методи генерації околиць. Автори публікації [13] аналізують ефективність кожної категорії окремо та визначають алгоритми, які показують кращу ефективність серед усіх індексних методів.

До категорії префіксних дерев відносять алгоритми, які знаходять схожі послідовності шляхом рекурсивного обходу дерева переходів, розглядаючи заміни, вставки та видалення вздовж шляху пошуку. Кловстат та Мондштей у своїй роботі [15] показали алгоритм пошуку, який базується на префіксному дереві «String trie», та довели, що переходячи за посиланнями автомату Левенштейна, цей алгоритм коректно відновіть шлях усіх кандидатів на схожу послідовність. Міхов та Шульц запропонували альтернативну структуру – FB-trie [16]. Основна відмінність полягає у тому, що метод підтримує два дерева – одне побудоване на оригінальному словнику, а інше – на словнику зворотних послідовностей. Це дозволяє комбінувати пошук у двох деревах, порівнюючи одночасно префікси і суфікси. У роботі [17] була запропонована ще одна суфіксна структура – k-errata trie. Вона націлена на оптимізацію найбільш складних випадків, впроваджуючи так звані centroid-шляхи дерева.

Найбільш ефективною категорією індексних структур пошуку вважають методи генерації околиці. Замість того, щоб шукати схожі послідовності у словнику, вони генерують околицю шуканого слова – фактично емулюють усі можливі відмінності, які можуть бути допущені від поточного рядка. Множину таких відмінностей з відстанню редагування не більше за k називають повною k-околицею. Після генерації такої околиці, кожен її елемент може бути дуже ефективно перевірений на наявність у словнику базовими методами, наприклад, хешуванням, або суфіксним деревом. Тривіальний алгоритм генерації такої околиці був вперше представлений Укконеном у роботі [18]. Пізніше Мейєрс у своїй роботі [19] представив так звані конденсовані околиці. Вони представляють собою повні околиці, з яких видалені неоптимальні елементи. Фактично в конденсованій околиці залишають лише елементи, які є суфіксами або префіксами яких-небудь слів зі словника.

Досить широкі тестові дослідження, проведені у роботі [13] показують, що найбільш ефективним алгоритмом серед індексних методів є алгоритм генерації околиць [18], тому саме він і буде протестований у складі алгоритму стиснення даних.

3. Методи кодування відмінностей

Наступним етапом в алгоритмі стиснення даних без втрат є кодування відмінностей. Як відомо, основна мета алгоритмів стиснення – зменшити фінальний розмір стисненого файлу. Використання схожих послідовностей дозволяє не записувати нову послідовність повністю, а записати лише посилання на попереднє входження та саме відмінність. Маючи ці дані, декодер зможе застосувати вказані операції до попереднього рядка і коректно відновити нову послідовність. Тому, не менш важливим етапом є максимально стиснути інформацію стосовно відмінностей, а саме – знайти спосіб закодувати відмінність мінімальною кількістю байт.

Базові алгоритми стиснення даних, такі як LZ77 [1] використовують лише точні збіги, тому їх можна закодувати досить просто – представити у вигляді пари (відстань; довжина). Декодеру потрібно буде просто скопіювати позначену підстроку. Але нам необхідно розглянути більш загальний випадок схожих послідовностей, тому окрім відстані і довжини необхідно також закодувати відмінності. На жаль, на теперішній час ця область розглядалася дуже стисло. Досить тривіальний підхід був запропонований у роботі [7], який базується на використанні відстані Левенштейна. Альтернативний метод, що ґрунтується на кодування відстані Геммінга, запропонували автори роботи [20] для використання у сфери стиснення зображень.

У якості першого методу буде розглянуто кодування відстані Левенштейна. Відстань Левенштейна дорівнює мінімальній кількості операцій, які необхідно щоб отримати з рядка A рядок B. У нашій задачі важлива не тільки кількість, а й самі операції для того, щоб можна було відновити новий рядок. Формально, мається оригінальний рядок A та послідовність операцій додавання, видалення або заміни символів у деяких позиціях. Приклад відстані Левенштейна та операцій для її досяжності зображений на рис. 3.

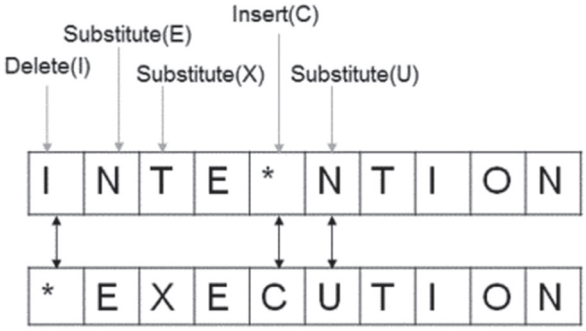


Рис. 3. Приклад перетворення рядка «INTENTION» у EXECUTION» методом Левенштейна

Автори роботи [7] пропонують розбити рядок на групи двох типів – група точного збігу та операція зміни. Для того, щоб групи чергувалися, між двома групами другого типу необхідно вставити пусті групи першого. Чергування груп необхідно для того, щоб забезпечити однакову кількість груп першого і другого типу. Після такого перетворення повне кодування схожого рядка можна зобразити наступною послідовністю: $(shift; length_1; length_2; \supset; length_n; delimeter; operation_1; operation_2; \supset; operation_n)$, де $shift$ – відстань до схожого рядка, $length_i$ – довжина групи повного збігу, $delimeter$ – спеціальний символ, який позначає закінчення опису груп першого типу, а $operation_i$ – операція зміни, яку необхідно зробити після копіювання групи під номером i. У свою чергу операція представляє собою два елементи – тип операції (заміна, додавання, або видалення символу) та новий символ для перших двох типів операцій.

Автори публікації зазначають, що такий підхід показує більш ефективну компресію на відміну від базового підходу. Основною перевагою цього підходу є те, що не потрібно кодувати відстань до кожної групи окремо, а достатньо лише один раз зазначити відстань до початку рядку, а потім просто записувати усі точні збіги один за одним, розділюючи їх операціями другого типу.

У роботі «Image Compression using Approximate Matching and Run Length»[20] пропонується альтернативний метод кодування відмінностей схожих послідовностей. Автори розглядають кодування відмінностей у контексті стиснення зображень, але їх метод можна розширити на компресію будь-яких даних. Основна ідея цього методу – використовувати відстань Геммінга замість відстані Левенштейна. Особливість відстані Геммінга полягає у тому, що вона є звуженням відстані Левенштейна і дозволяє тільки операції заміни символу. Більш того, у своєму оригінальному формулювання ця відстань розглядається лише у полі бінарних рядків, але вона може бути досить просто узагальнена на рядки будь-якого алфавіту. Також у цьому методі пропонується альтернативний спосіб кодування груп з точним збігом та замін. Замість кодування довжини групи, як це було зроблено у попередньому методі, алгоритм кодує одне бітове значення для кожного символу – «1», якщо символ співпадає, та «0» якщо він відрізняється. У разі, коли символ відрізняється, після нього кодується також новий символ, на який його потрібно замінити. Після цього послідовні повторення нулів та одиниць додатково стискаються алгоритмом RLE [21].

Автори публікації стверджують, що на практиці схожі послідовності досить малі, та операція заміни використовується частіше, тому їх алгоритм дає кращі результати ніж попередній. Нижче буде експериментально перевірено обидва запропонованих методи.

4. Метрики оцінювання якості алгоритмів

Для оцінки алгоритмів стиснення даних без втрат враховуються три основні показники:

ступінь стиснення – співвідношення між оригінальним розміром файлу та розміром файлу після стиснення;

швидкість компресії (МБ/секунду) – швидкість роботи алгоритму стиснення;

швидкість декомпресії (МБ/секунду) – швидкість роботи алгоритму відновлення оригінального файлу.

Розрахунок вказаних показників проводиться за наступними формулами:

$$compression\,ratio = \frac{original\ file\ size}{compressed\ file\ size},$$

$$compression\,speed = \frac{original\ file\ size}{compression\,time},$$

$$decompression\,speed = \frac{original\ file\ size}{decompression\,time}.$$

Найважливішим показником звісно є саме ступінь стиснення, тому що він характеризує, наскільки добре алгоритм стиснення здатен зменшити розмір файлу відносно його початкового розміру. В деяких роботах до оригінального розміру файлу також додається розмір програмного коду, який використовується для компресії і декомпресії. Але в нашому випадку тестування буде проводитись на великому об'ємі даних, тому розміром програмного коду можна знехтувати.

5. Огляд існуючих датасетів

У якості текстового датасету був обраний досить популярний датасет enwik8, який містить перші 10^8 символів англійської Вікіпедії. Аналіз датасету показав, що він містить 709,405 унікальних слів. На рис. 4 зображено розподіл 100 найчастіших слів у датасеті.

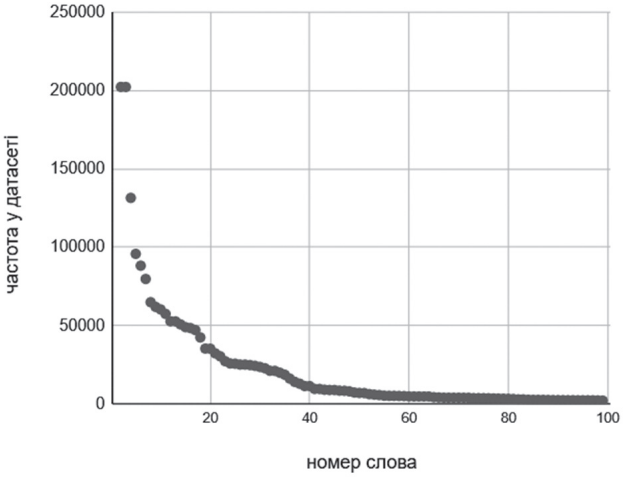


Рис. 4. Розподіл 100 самих частих слів у датасеті

З графіку можна переконатися, що текст містить велику кількість дуже частих слів, які багато повторюються, що посприяє ефективності роботи алгоритму.

В якості загального датасету був обраний розширений датасет Silesia Corpus, який складається з набору файлів популярних типів. Типи файлів у датасеті розділені на 6 категорій – виконувані файли, зображення, мультимедіа, документи, файли розмітки та файли баз даних. Кожна категорія містить декілька файлі різних форматів. Детальний аналіз вмісту датасету наведений у табл. 1.

Таблиця 1

Розподіл даних у датасеті Silesia Corpus

Тип файлів	Формати	Кількість файлів	Загальний розмір, МБ
Виконувані файли	EXE	5	16.8 МБ
Зображення	PNG, JPEG, BMP, TIFF	16	31.1 МБ
Мультимедіа	MP4, WEBM, MPEG, AVI	7	45.6 МБ
Документи	PDF, DOC, DOCX, PPTX	10	4.2 МБ
Файли розмітки	HTML, XML	6	0.7 МБ
Файли баз даних	DB, SQL	3	8.2 МБ

Загалом датасет enwik8 містить 1 файл розміром 100МБ, а датасет Silesia Corpus містить 47 файлів сумарним розміром 106.6 МБ.

6. Опис експериментальних досліджень

Виходячи з аналізу, проведеного у попередніх пунктах, найбільший інтерес представляють три алгоритми пошуку схожих послідовностей даних – алгоритм Укконена [11], що відноситься до евристичних алгоритмів, метод N-грам [12], та індексний метод генерації околиць [18]. Перш за все був реалізований алгоритм Укконена за схемою, описаною у оригінальній публікації. Алгоритм був реалізований на мові програмування C++ та вбудований у загальну реалізацію алгоритму стиснення даних LZ77 [1]. Як визначалося вище, цей алгоритм є евристичним тому його асимптотична складність складає $O\left(N^2 * k\right)$, де N – кількість слів у датасеті. Датасет enwik8 містить приблизно 10 мільйонів слів, тому навіть зі статистичними оптимізаціями реалізований алгоритм працює дуже довго. Для того, щоб отримати реалістичний час, алгоритм був допрацьований за схемою, описаною у [21]. Фактично, датасет був розділений на невеликі блоки і пошук схожої послідовності проводився тільки на останньому блоці. На рис. 5 зображена залежність ступеню стиснення даних від розміру блоку. У якості розмірів блоку були розглянуті ступені двійки, а саме 16, 64, 256, 1024, 4096, 16384, 65536, 262144 та 1048576 байтів. Для кращого сприйняття дані наведені на логарифмічній шкалі. З маленьким розміром блоку алгоритм працював дуже швидко, наприклад опрацювання всього словнику з блоком розміру 16 зайняло усього 5 секунд. Але через малий розмір блоку ступінь стиснення виявилася досить малою. Зі збільшенням розміру блоку датасет стискався краще, але алгоритм працював повільніше. На розміру блоку 1 МБ обробка датасету зайняла 28 хвилин.

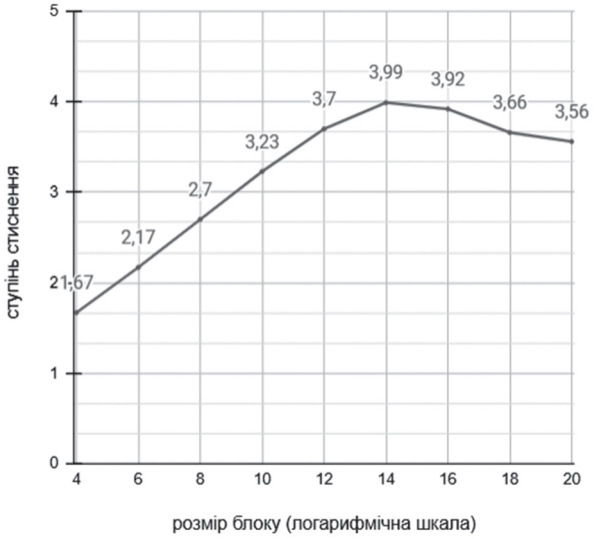


Рис. 5. Ступінь стиснення алгоритмом Укконена залежно від розміру блоку

Цікаво, що починаючи з розміру блоку 2^{16} якість стиснення почала спадати. Це пов'язано з тим, що коли блок дуже довгий, алгоритм знаходить схожі слова досить далеко позаду і кодує велику відстань, що є менш оптимальною, ніж закодувати декілька маленьких ближче чи не кодувати це слово взагалі. Оптимальним розміром блоку було визначено 16 КБ.

Наступним протестованим алгоритмом був метод N-грам [12]. Алгоритм складається з декількох етапів, на кожному з яких перевіряються лише той набір слів, який пройшов попередні перевірки. З цього набору слів обирається підмножина, яка містить достатню кількість N-грам поточної довжини. Автори стверджують, що після четвертого етапу множина потенційних кандидатів стиснеться до дуже малого розміру і усі його елементи можна перевірити стандартною метрикою. Алгоритм був реалізований на мові програмування C++ та протестований на датасеті enwik8. У табл. 2 наведено відповідність між номером етапу та відсотком кандидатів від загальної кількості слів у словнику.

Таблиця 2

Розмір множин кандидатів на етапах методу N-грам

Номер етапу	Відсоток потенційних кандидатів
1-грами	13.32%
2-грами	4.77%
3-грами	2.36%
4-грами	1.29%

Наведені результати були отримані шляхом тестування реалізованого методу N-грам у складі онлайн

схеми алгоритму, описаної у розділі 1. Відсоток був усереднений по всіх словах датасету. Як можна побачити з таблиці, множина кандидатів звужується дуже швидко і після чотирьох етапів залишається лише 1.3% кандидатів порівняно до оригінальної множини. Для пришвидшення алгоритму, так як і в попередньому випадку, датасет був поділений на блоки по 16КБ. Метод N-грам працює швидше евристичного методу, і обробка усього датасету зайняла 105 секунд.

Найбільш сучасним та ефективним методом пошуку схожих послідовностей є метод генерації околиць. У роботі [13] приводиться алгоритм онлайн підтримки словнику та пошуку послідовностей, редакційна відстань до яких не перевищує k. На основі цього алгоритму була протестована залежність розміру околиці від кількості зроблених змін у шуканому слові, результати чого зображені на рис. 6. Можна побачити, що допустимий розмір не повинен перевищувати 5, тому що для більших значені розмір околиці є занадто великим.

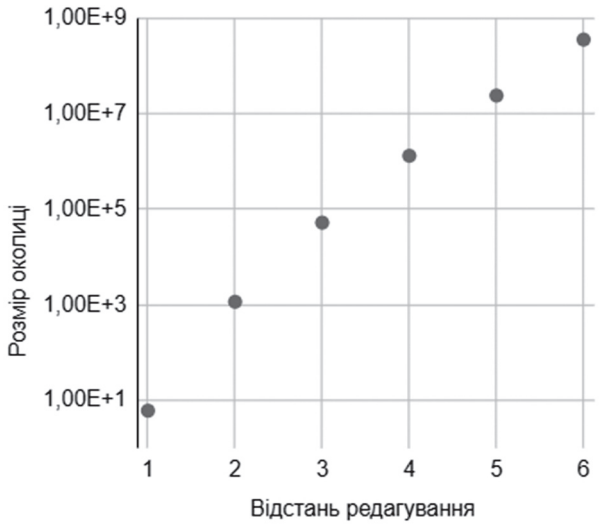


Рис. 6. залежність розміру околиці від відстані редагування

Для виконання перевірки префіксного чи суфіксного збігу буде використане структура даних префіксне дерево, побудоване на множині вже оброблених слів. Для того, щоб використовувати пошук суфіксів, необхідно побудувати два таких дерева – для прямих слів та перевернутих.

У фінальній частині дослідження було проведено тестування комбінацій розроблених методів у складі алгоритму стиснення даних. Для проведення тестування був реалізований застосунок на мові програмування Python. Також була використана утиліта lzbench для автоматизації тестування. Результати проведеного тестування зображені на рис. 7.

Всього було протестоване 6 комбінацій:

– евристичний метод Укконена з кодуванням методом Левенштейна;

– евристичний метод Укконена з кодуванням методом Геммінга;

– метод N-грам з кодуванням методом Левенштейна;

– метод N-грам з кодуванням методом Геммінга;

– індексний метод генерації околиць з кодуванням методом Левенштейна;

– індексний метод генерації околиць з кодуванням методом Геммінга.

На рис. 8 представлені результати тестування на комбінованому датасеті Silesia Corpus. Отримані результати експериментів показали, що найбільш ефективним методом пошуку схожих послідовностей на обох датасетах виявився евристичний метод Укконена, проте він є самим повільним. Цьому алгоритму вдається досягти такої високої ступені стиснення завдяки тому, що він тривіальна перебирає усіх можливих кандидатів. В той же час індексний метод генерації околиць працює майже в п’ять разів швидше, при тому не сильно поступаючись у ступені стиснення. Метод N-грам теж показав конкурентні

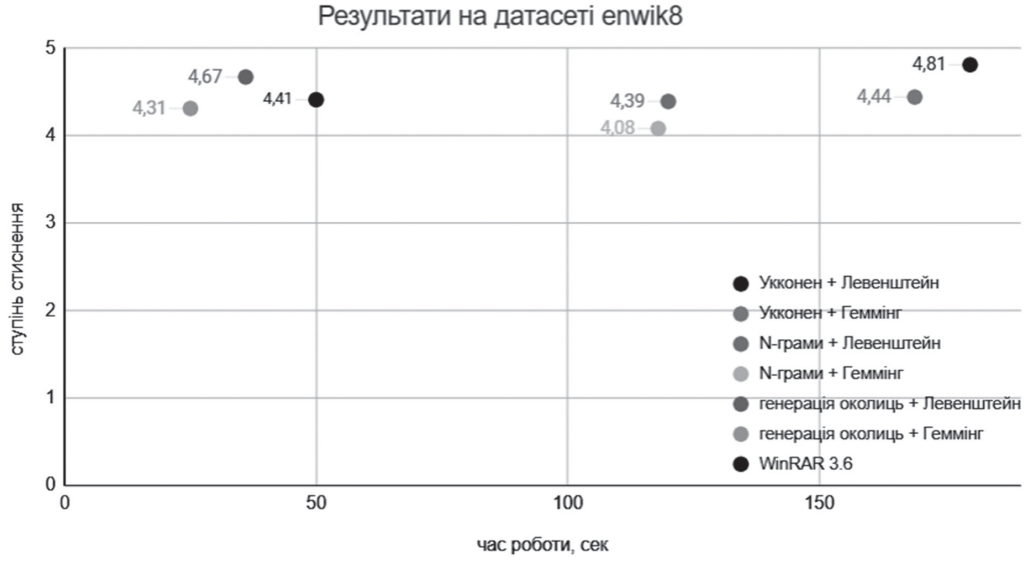


Рис. 7. Результати роботи алгоритмів на датасеті enwik8

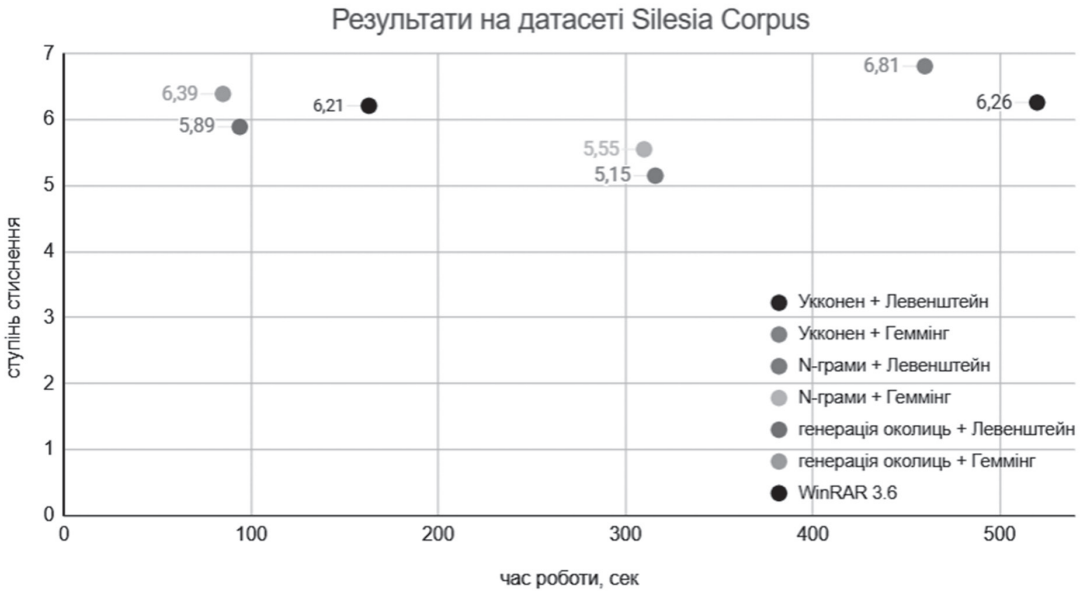


Рис. 8. Результати роботи алгоритмів на датасеті enwik8

результати, проте, поступився методу генерації околиць як по швидкості, так і по ступеню стиснення, що робить його не ефективним на практиці.

З іншого боку, методи кодування відмінностей показали себе неоднозначно. На датасеті enwik8 більш ефективним виявився метод Левенштейна, проте на датасеті Silesia Corpus – метод Геммінга. Це пов'язано з тим, що enwik8 складається зі статей природною мовою, де часто трапляються схожі послідовності з різними варіантами відмінностей – додаванням символу, видаленням, та заміною. Як раз це і забезпечив алгоритм Левенштейна. В той же час датасет Silesia Corpus містить більшість даних машинного формату – виконувані файли, зображення, мультимедіа, тощо. На таких даних частіше трапляються відмінності пов'язані лише с заміною символу, та зустрічаються багато дуже схожих послідовностей. Тому на таких даних метод Геммінга досяг більшого степеню стиснення. Також на обох датасетах підхід кодування методом Гемінга працював швидше, тому що він розглядає меншу кількість операцій і, відповідно, кандидатів.

Також для порівняння розроблених комбінацій з так званою базовою лінією, вони були порівняні з відомим компресором WinRAR версії 3.6. Як можна побачити з графіків, метод Укконена на обох датасетах випереджує WinRAR за ступенем стиснення, а метод генерації околиці – не тільки за ефективностю, а й за швидкістю.

Отже, розглядаючи алгоритм стиснення даних загалом, найкращим варіантом виявився індексний метод генерацій околиць у комбінації з кодуванням Геммінга. У спеціальному випадку роботи з природньою мовою слід використовувати той же метод, але з кодуванням Левенштейна.

Висновки

В ході проведення теоретичного аналізу була запропонована розширена версія алгоритму компресії, досліджені різноманітні підходи до пошуку схожих послідовностей, розглянуті евристичні методи, методі N-грам та індексні методи. Також були проаналізовані методи кодування відмінностей Левенштейна та Геммінга.

На етапі проведення експериментальних досліджень було обрано два репрезентативних датасети: enwik8 – датасет статей англійською мовою, та Silesia Corpus – комбінований датасет різних типів даних. На основі результатів аналізу було визачено три кращі методи пошуку схожих послідовностей – метод Укконена, метод N-грам та метод генерації околиць. Обрані методи були реалізовані та протестовані на визначеному датасеті в комбінації алгоритмами кодування відмінностей, базованих на відстані Левенштейна та Геммінга.

У результаті проведених теоретичних та експериментальних досліджень було обрано дві найбільш ефективні комбінації підходів до стиснення даних без втрат. На датасеті природньої мови кращий результат показав метод генерації околиць у комбінації з кодуванням Левенштейна, який досяг ступеню стиснення 4,67 за найкоротший час. Для комбінованого датасету найефективнішим також виявився метод генерації околиць, але з кодуванням Геммінга, та показав ступень стиснення 6,39.

Перспективним продовженням дослідження є аналіз більш широкого набору методів для пошуку схожих послідовностей (зокрема, нейронні мережі та підходи NLP). Крім того, доцільно проаналізувати комбінації методів на окремих датасетах для різних типів і форматах даних.

Список літератури:

[1] Jacob Ziv, Abraham Lempel. (1977) A Universal Algorithm for Sequential Data Compression IEEE Transactions on Information Theory, pp. 337—343.

[2] Van Leeuwen, Jan (1976) On the construction of Huffman trees, pp. 382–410.

[3] Katz, Phillip W. (1991) String searcher, and compresor using same string patterns.

[4] Ranganathan, N, Henriques, S. (1993) High-speed VLSI designs for Lempel-Ziv-based data compression. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol-40, No.2, pp. 96–106.

[5] E.Jebamalar Leavline ,D.Asir Antony Gnana Singh (2013) Hardware Implementation of LZMA Data Compression Algorithm. International Journal of Applied Information Systems (IJAIS).

[6] Apoorv Gupta, Aman Bansal, Vidhi Khanduja, (2017) Modern Lossless Compression Techniques: Review, Comparison and Analysis.

[7] Ilan Sadeh (2009) Universal data compression based on approximate string matching.

[8] Petteri J, Jorma T, Ukkonen E. (1998) A Comparison of Approximate String Matching Algorithms.

[9] Z. Galil, K. Park (1990) An improved algorithm for approximate string matching, SIAM Journal on Computing, 19, pp. 989–999.

[10] R. Grossi, F. Luccio (2018) Simple and efficient string matching with k mismatches, Information Processing, pp. 113–120.

[11] E. Ukkonen (1992) Approximate string matching with q-grams and maximal matches, Theoretical Computer Science, 92, pp. 191–211.

[12] Narendra Kumar, Vimal Bibhu, Mohammad Islam, Shashank Bhardwaj (2017) Approximate string matching using n-grams technique.

[13] Boytsov L. (2011) Indexing Methods for Approximate Dictionary Searching: Comparative Analysis.

[14] A. Yerokhin, A. Nechyporenko, O. Turuta, A. Babii (2016) A new intelligence-based approach for rhinomanometric data processing 2016 IEEE 36th International Conference on Electronics and Nanotechnology (ELNANO), Kiev, 2016, pp. 198-201.

[15] Klovstad J., Mondshein L. (1975) The CASPERS linguistic analysis system. IEEE Transactions on Acoustics, Speech and Signal Processing 23, pp. 118 – 123.

[16] Mihov S., Schulz K. U. (2004) Fast approximate string search in large dictionaries. Computational Linguistics, 30, 4, pp. 451–477.

[17] Cole R., Gottlieb, L. A., AND Lewenstein, M. (2004) Dictionary matching and indexing with errors and don't cares. In STOC '04: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing. ACM, pp. 91–100.

[18] Ukkonen, E. (1985) Algorithms for approximate string matching. Information and Control 64, 1-3, pp. 100–118.

[19] Myers, E. (1994) A sublinear algorithm for approximate keyword searching. Algorithmica 12, 4/5, pp. 345–374.

[20] Samir Kumar Bandyopadhyay, Tuhin Utsab Paul, Avishek Raychoudhury (2016) Image Compression using Approximate Matching and Run Length.

[21] Robinson, A. H. Cherry, C. (1967) Results of a prototype television bandwidth compression scheme, Proceedings of the IEEE, pp. 356–364.

Надійшла до редколегії 26.03.2021

УДК 004.8

DOi 10.30837/ bi.2021.1(96).08

M.O. Butsenko[1], I.V. Afanasieva[2], N.V. Golian[3], N. Kameniuk[4]

[1]backelor's student of the department of Software Engineering,
Kharkiv National University of Radio Electronics, Ukraine, mykyta.butsenko@nure.ua

[2]associate professor of the department of Software Engineering,
Kharkiv National University of Radio Electronics, Ukraine, iryna.afanasieva@nure.ua

[3]associate professor of the department of Software Engineering,
Kharkiv National University of Radio Electronics, Ukraine, nataliia.golian@nure.ua

[4]master's student of the department of Software Engineering,
Riga Technical University, Latvia, nataliia.kameniuk@gmail.com

A NEURAL NETWORK APPROACH FOR THE AUTOMATIC SELECTION OF A COMPLEX OF REHABILITATION EXERCISES

This article is devoted to solving the problem of automatic selection of a set of rehabilitation exercises during injuries, considering the state of the human cardiovascular system through the use of neural networks. To solve this problem, it was necessary to choose one of two classical approaches – multiclass classification or multilabel classification, each of which solves the problem of data classification through its own algorithm, and use the selected neural network architecture to create a software system. While working on this system, it was also necessary to solve certain problems related to each of these approaches (the need for a large sample due to the large number of exercises that the system should recommend) or a specific approach (inability to select multiple exercises at once – for Multiclass Classification, lower productivity and the number of supported programming languages – for Multilabel Classification).

Samples of different sizes (from 1 million records and more) were used to train the neural network, which were generated through a self-written program that generated a given number of records and wrote them to a .CSV (comma-separated values) file.

ANALYSIS, EXERCISE, MULTICLASS CLASSIFICATION, MULTILABEL CLASSIFICATION, NEURAL NETWORK, SOFTWARE SYSTEM, RECOMMENDATION, CARDIOVASCULAR SYSTEM, INJURY

Буценко М.О., Афанасьева И.В., Голян Н.В., Каменюк Н. Нейросетевой подход автоматического подбора комплекса реабилитационных упражнений. Эта статья посвящена решению проблемы автоматического подбора комплекса восстановительных упражнений при травмах с учетом состояния сердечно-сосудистой системы человека с помощью использования нейронных сетей. Для решения данной задачи необходимо было выбрать один из двух классических подходов – Multiclass Classification или Multilabel Classification, каждый из которых решает проблему классификации данных через собственный алгоритм, и использовать выбранную архитектуру нейронной сети для написания программной системы. Во время работы над этой системой также необходимо решить определенные проблемы, касающиеся каждого из этих подходов (необходимость в большой выборке из-за большого количества упражнений, которые система должна рекомендовать) или конкретного подхода (невозможность выбрать несколько упражнений одновременно – для Multiclass Classification, меньше производительность и количество поддерживаемых языков программирования – для Multilabel Classification).

Для обучения нейронной сети использовались выборки различных размеров (от 1 млн. записей и более), генерировались через самостоятельно написанную программу, которая генерировала заданное количество записей и записывала их в .CSV (comma-separated values) файл.

АНАЛИЗ, УПРАЖНЕНИЕ, МУЛЬТИКЛАССОВАЯ КЛАССИФИКАЦИЯ, МУЛЬТИЛЕЙБЛОВАЯ КЛАССИФИКАЦИЯ, НЕЙРОННАЯ СЕТЬ, ПРОГРАММНАЯ СИСТЕМА, РЕКОМЕНДАЦИЯ, СЕРДЕЧНО-СОСУДИСТАЯ СИСТЕМА, ТРАВМА

Буценко М.О., Афанасьєва І.В., Голян Н.В., Каменюк Н. Нейромережевий підхід для автоматичного підбору комплексу реабілітаційних вправ. Ця стаття присвячена рішенню проблеми автоматичного підбору комплексу відновлювальних вправ під час травм з урахуванням стану серцево-судинної системи людини за допомогою використання нейронних мереж. Для вирішення даної задачі необхідно було обрати один із двох класичних підходів – Multiclass Classification або Multilabel Classification, кожний з яких вирішує проблему класифікації даних через власний алгоритм, та використати обрану архітектуру нейронної мережі для написання програмної системи. Під час роботи над цією системою також необхідно було вирішити певні проблеми, що стосувалися кожного з цих підходів (необхідність у великій вибірці через велику кількість вправ, що система має рекомендувати) або конкретного підходу (неможливість обрати декілька вправ одночасно – для Multiclass Classification, менша продуктивність та кількість підтримуваних мов програмування – для Multilabel Classification).

Для навчання нейронної мережі використовувались вибірки різних розмірів (від 1 млн. записів та більше), що генерувались через самостійно написану програму, що генерувала задану кількість записів та записувала їх у .CSV (comma-separated values) файл.

АНАЛІЗ, ВПРАВА, МУЛЬТИКЛАСОВА КЛАСИФІКАЦІЯ, МУЛЬТИЛЕЙБЛОВА КЛАСИФІКАЦІЯ, НЕЙРОННА МЕРЕЖА, ПРОГРАМНА СИСТЕМА, РЕКОМЕНДАЦІЯ, СЕРЦЕВО-СУДИННА СИСТЕМА, ТРАВМА

1. Introduction

Every day, many people receive injuries of varying severity. In most cases, in order to fully restore the function of the injured part of the body, it is necessary to regularly perform restorative exercises, some of them due to the high intensity can put a heavy strain on the cardiovascular system [1].

Depending on the location and degree of damage, the patient needs restorative exercises of varying intensity.

Making the right list of such exercises can help information about the age and sex of the person. But the fact is that such data are not enough to make such a set of exercises that is guaranteed not to harm the health of the patient.

That is why there is a need to develop a system that provides a set of exercises for home use, namely physical therapy for injuries and fractures, considering the state of the cardiovascular system of the patient and his sex, as well as age.

2. Analysis of competitors and problem statement

There are systems [2, 3, 4] that provide general advice on a set of restorative exercises, but without reference to the state of the cardiovascular system. Simply put, they simply provide sets of restorative exercises.

That is, there is no example of software on the market that could provide a set of exercises, using not only the injury and the number of fractures, dislocations or sprains, but also related indicators of the cardiovascular system and blood tests.

Accordingly, the question arises as to how to analyze such an array of data. For this purpose, it is appropriate to use neural networks [5], as they can provide recommendations with some accuracy, using a sample, which is presented, for example, in the form of a .CSV-file.

An example of the structure of such a file is shown in Figure 1.

It can be seen that the data in it is divided into rows and columns, which in turn are separated by a comma with a space.

3. Multiclass classification

In this case, since we solve the problem of selecting a set of exercises with reference to certain indicators of the cardiovascular system, it is logical to use an approach called Multiclass Classification (it solves the problem of classifying specimens into one of several classes) [6].

The diagram showing the main problem solved by this approach is shown in Figure 2.

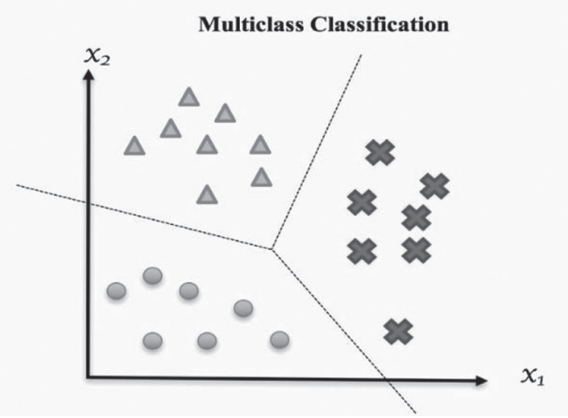


Fig. 2. A diagram illustrating the problem that the Multiclass classification solves

The algorithm responsible for the selection of exercises for recovery works as follows: the entrance is given the age

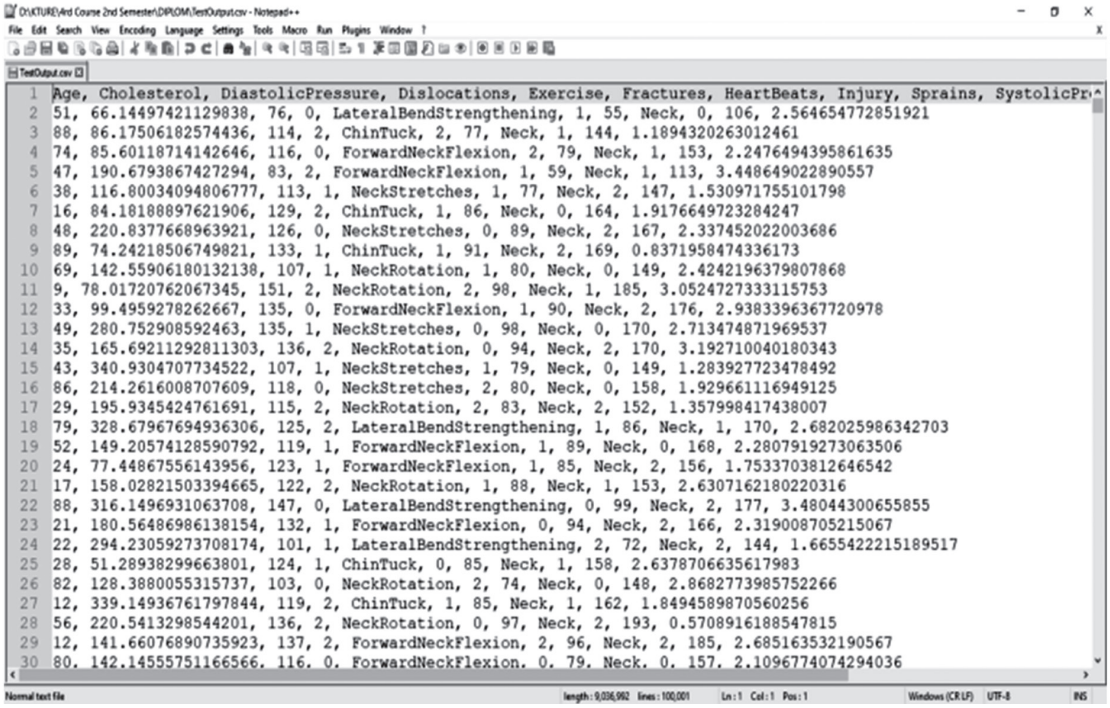


Fig. 1. An example of a .CSV file

and sex of the person, his injury, as well as indicators responsible for the work of his cardiovascular system.

These indicators include::

– systolic and diastolic pressure;

– patient's age (Age);

– as well as key indicators of recent blood tests - cholesterol and triglycerides [7].

The initial data in this case is the result provided by the input data: age, sex, pressure, blood tests, etc. A set of exercises represented as a single object, such as a string in a .CSV file. Together, these data form the training and test samples needed to train the model and form the neural network.

4. Multilabel classification

In addition to the Multiclass Classification approach, we can use the Multilabel classification [8]. An example is shown in Figure 3.

This approach is a generalized version of the multiclass classification, but with one difference — the Multiclass Classification provides only one object (label) as the source data, while the Multilabel classification has no restrictions on the amount of source data.

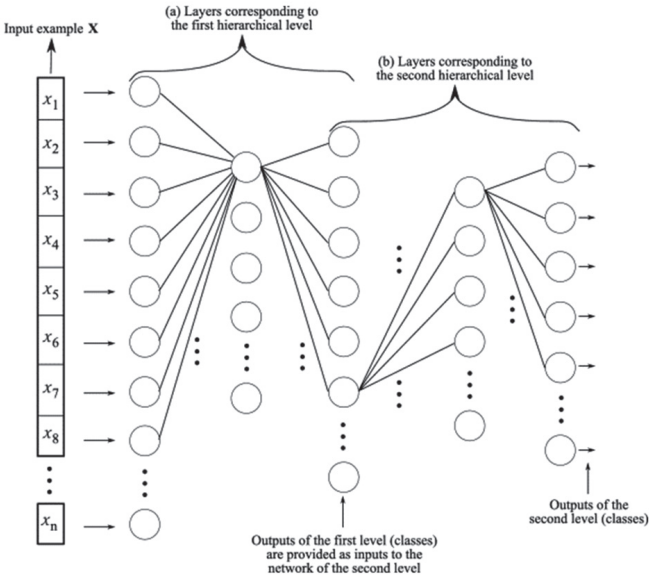


Fig. 3. An example of a Multilabel classification

That is, if the first approach provides one of the precreated complexes, then in the case of Multilabel classification, this complex is formed automatically from individual exercises.

5. Features of approaches

Figure 4 shows the difference between this approach and multiclass classification.

The Multilabel classification approach is potentially more accurate, but is not supported by all programming languages. For example, ML.NET — a framework from Microsoft for the C # programming language — does not currently support Multilabel classification; it is suggested to use Multiclass Classification instead.

Fig. 4. The main difference between Multiclass Classification and Multilabel classification

Also, this approach in most systems, including ours, is less productive, because the recommendation of a set of exercises is a costlier operation in terms of time and resources than the recommendation of a single exercise.

Both of these approaches are relevant to our system, but they have the same drawback — the need for a large sample.

For example, the San Francisco Department of Health Restaurant Table, which lists all inspections with detected violations and their low-medium-high-risk classifications, is posted on the Microsoft website and used to explain the Multiclass Classification using a forecast degree of risk [9], contains approximately 50 thousand records. That is, in order to teach the system to classify violations into three categories with high accuracy — in this case the accuracy is about 100% — a very large data set was needed.

Since there are only more than 3 types of injuries, and each injury requires 3-5 exercises to assemble recovery complexes, the first calculations to ensure high accuracy (90% or more) may require a large sample of tens of millions of records.

Such an array of data can be obtained only by collecting data from a large number of hospitals, or by generating them yourself, referring to scientific advice on the selection of exercises for recovery from injury.

6. Neural network architecture using ML.NET and multiclass classification

At the beginning of the work it is necessary to form a sample, which is used to create a neural network.

The sample size directly depends on the exercises that the system can recommend for recovery — as the number of exercises increases, the sample itself should increase to ensure high accuracy of the recommendation (90% or more).

After its formation it is necessary to pass to creation of a neural network. With the ML.NET framework, this can be done in two ways: through the Model Builder GUI or directly through the API. In the case of Multiclass Classification, it is appropriate to use Model Builder, because it supports the data approach. The principle of its operation is shown in Figures 5 and 6. They show that

when using it ML.NET will independently divide the sample into Test and Train Data and form a neural network; for this he only needs a sample with the specified Features and Label.

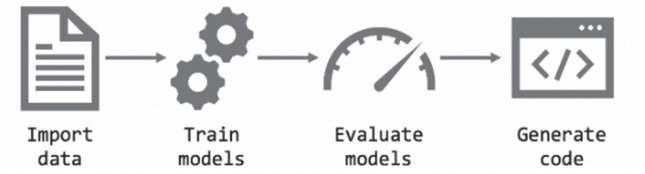


Fig. 5. The principle of ML.NET when using Model Builder

At the Figure 6 we can see also sample distribution on Features and Label.

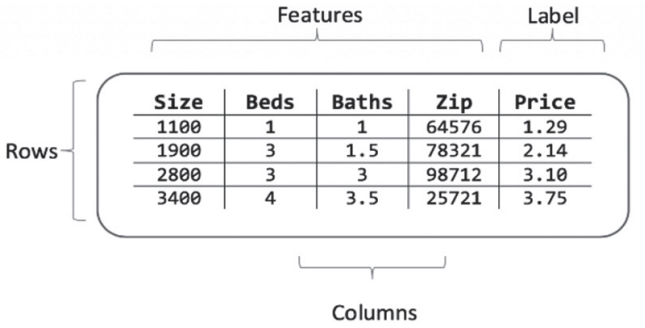


Fig. 6. Sample distribution on Features and Label

The neural network is in the layer of business logic; it must provide high accuracy in order to adequately recommend exercises for rehabilitation after injuries.

The system should expand the sample provided in order to continuously train the neural network.

For ease of use of the neural network, the sample is stored in a separate database table.

The structure (columns) of this table are as follows:

– injury ID (InjuryID);
– patient's age (Age);
– fractures quantity (Fractures);
– dislocations quantity (Dislocations);
– sprains quantity (Sprains);
– cholesterol;
– triglycerides;
– systolic pressure (SystolicPressure);
– diastolic pressure (DiastolicPressure);
– heart beats per minute (HeartBeats);
– exercise ID (ExerciseID).

This table uses trauma and exercise identifiers, as this table is not isolated throughout the system, but is a full participant.

There are other tables in the system, in particular for the same injuries and exercises referred to in the table above, using these identifiers. It is designed so that when deleting any injury or exercise (i.e., records from the main tables), its records that use the identifiers of the deleted entities are not deleted by the system, but are ignored during the selection of recommended injuries for recovery.

In the future, they can be used again if the deleted records are restored while retaining their IDs.

The part of the system that uses the created neural network is presented as a separate project. It receives requests from controllers that use HTTP methods (mainly Get and Post versions), which in turn receive requests from medical devices that collect indicators of the state of the cardiovascular system, or from the emulator of these devices.

The neural network analyzes the obtained indicators and selects the exercise or their complex according to the table above.

After the work is done, it sends a signal to the controller that the selection process has been successfully completed, and the controller sends it to the medical device or emulator. After that, the user is expected to go to the client part of the application to view the result in the form of selected exercises, as well as collected tests for a general understanding of the state of his cardiovascular system.

7. Integration and implementation

The neural network project is integrated into the system at the level of business logic. With the ML.NET feature, you can access non-transferable data types transferred to their current table, which can be written as an optional table that stores files created through files.

The file was generated through an individual algorithm, which randomly generated age and sex (age, gender), then selected successors according to individual medical rules:

– cholesterol;
– triglycerides;
– systolic pressure (SystolicPressure);
– diastolic pressure (DiastolicPressure);
– heart beats per minute (HeartBeats)

Examples of value selection rules are given in the code below. They use the age and sex of the user as initial parameters, and at the output give indicators of cardiovascular condition in the form of systolic, diastolic pressures and heart rate.

```
switch (gender)
    {
      case MaleGender:
        switch (age)
        {
            case >= MinAge and < 20:
                heartParametersTuple.systolicPressure =
NumbersGenerator.GenerateRandomInt(114, 126);
                heartParametersTuple.diastolicPressure =
NumbersGenerator.GenerateRandomInt(70, 83);
                heartParametersTuple.heartBeats =
NumbersGenerator.GenerateRandomInt(60, 80);
                break;
            case >= 20 and < 30:
                heartParametersTuple.systolicPressure =
NumbersGenerator.GenerateRandomInt(120, 131);
                heartParametersTuple.diastolicPressure =
NumbersGenerator.GenerateRandomInt(74, 86);
```

```
                heartParametersTuple.heartBeats =
NumbersGenerator.GenerateRandomInt(50, 90);
                break;
            case >= 30 and < 40:
                heartParametersTuple.systolicPressure =
NumbersGenerator.GenerateRandomInt(124, 133);
                heartParametersTuple.diastolicPressure =
NumbersGenerator.GenerateRandomInt(76, 88);
                heartParametersTuple.heartBeats =
NumbersGenerator.GenerateRandomInt(60, 90);
                break;
}
```

After generating the values, this .CSV file was used as the source tool for learning the system. Figure 7 shows the first step of adding a note to a neural network as a sample for model learning. It is from this sample that the parameters used as properties in the model are selected.

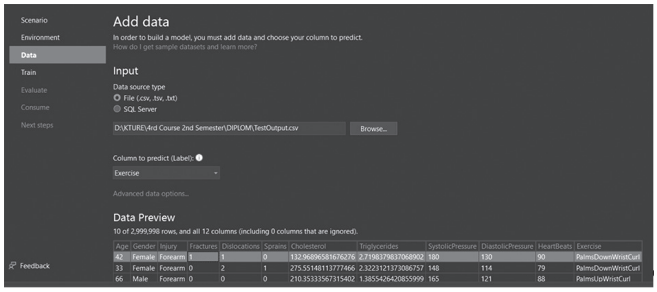


Fig. 7. Initial process of adding data to ML.Net through Model Builder

Figure 8 shows an example estimate that the system uses to train a neural network. The Time to train value increases in proportion to the sample size.

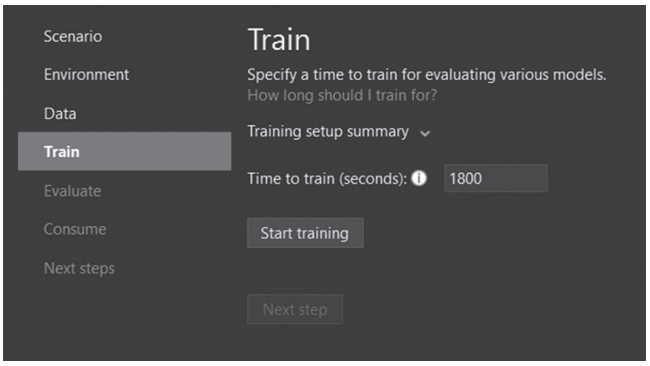


Fig. 8. Time to train estimated by ML.NET and Model Builder

Figure 9 shows the beginning of sampling training. Initially, the accuracy was unsatisfactory (21%), as the most optimized model for training had not yet been selected.

Figure 10 shows the optimal results and model in the selection process.

The accuracy of 91.8% is the value that satisfies the condition of "sufficient accuracy", which was set at the beginning.

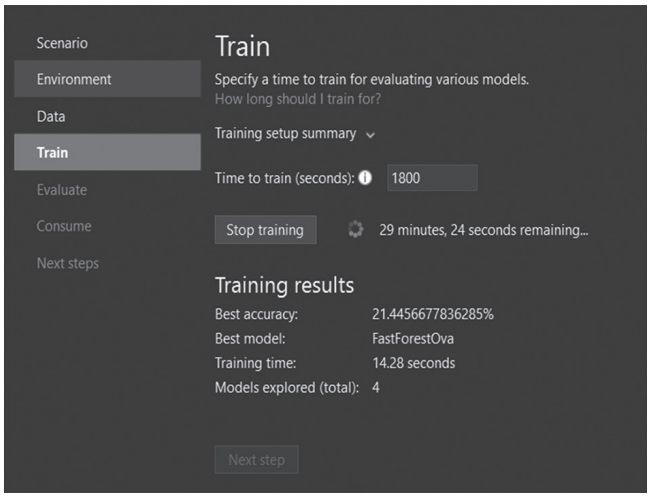


Fig. 9. Non-optimized model and its respective accuracy value

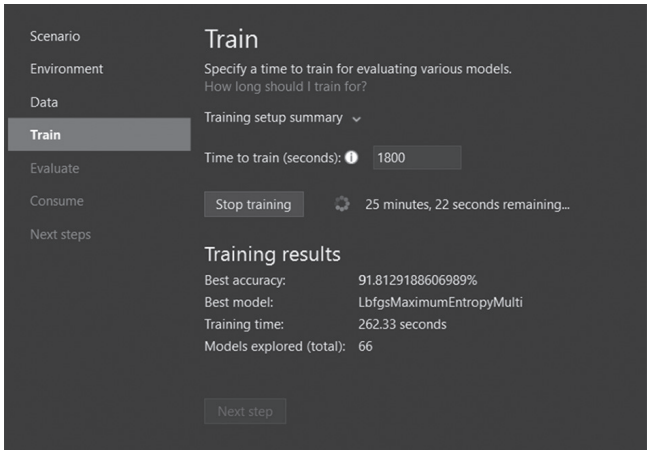


Fig. 10. Non-optimized model and its respective accuracy value

The release of ML.NET and Model Builder released a trained model that provides basic interfaces for testing and initial verification. This principle of learning became the basis for the creation of a neural network based on the principle of Multiclass Classification. In theory, the same approach could work with Multilabel Classification, but at the time of writing, this neural approach was not supported by the ML.NET framework.

8. Conclusions

The paper considers the use of neural network approach for the automatic selection of a complex of rehabilitation exercises during injuries, considering the state of the human cardiovascular system through the use of neural networks. A study is presented to investigate scenarios for applying multiclass and multilabel classification methods to select the most appropriate exercises for each particular case.

It can be concluded that both of these algorithms are appropriate to resolve the issue although each method has its own advantages and disadvantages. It is important to notice that the multiclass classification is considerably simpler and it has a wider list of programming languages and frameworks which natively support the method. On

another side, the multilabel classification can solve more complex problems and can be an appropriate solution when there is a need to select a complex entity which cannot be retrieved by using that the multiclass classification algorithm [10]. In the research, the model built by ML.NET framework was used. It contained 10 properties called features; they were used to describe the injury, and one single property called labels which were used as a respective exercise to heal the injury.

As a point for further research, it is proposed to build the complex multilabel classification model using the same framework. Since ML.NET does not offer any native support for such method, it can be taken as a goal to solve this problem and compare results with the native multiclass classification ones.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Walter R. F. Rehabilitation of Sports Injuries // R. Frontera Walter., 2002.

[2] Physical Therapy Exerises // abayapps. 2019. –https://play.google.com/store/apps/details?id=com.rehabilitation.physicaltherapyexercises.

[3] Injurymap - Effective exercise therapy // Injurymap, 2020. – https://play.google.com/store/apps/details?id=com.injurymap.injurymap.

[4] Knee Pain Relieving Exercises // Dr.Kavin Khatri, 2016. –https://play.google.com/store/apps/details?id=com.technoherpes.kneepain.

[5] Dybowski R. Clinical Applications of Artificial Neural Networks // R. Dybowski., 2001.

[6] Kolo B. Binary and Multiclass Classification // B. Kolo. – Weatherford: Weatherford Press, 2010.

[7] Walker H. K. Clinical Methods: The History, Physical, and Laboratory Examinations // H. K. Walker, W. D. Hall, J. W. Hurst., 1990.

[8] Multilabel Classification: Problem Analysis, Metrics and Techniques // F.Herrera, F. Charte, A. J. Rivera, M. J. del Jesus. – New York: Springer, 2016.

[9] Tutorial: Classify the severity of restaurant health violations with Model Builder // Microsoft, 2019. – https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/health-violation-classification-model-builder.

[10] Shubham J. Solving Multi-Label Classification problems (Case studies included) // Jain Shubham, 2017. –https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification.

The article was delivered to editorial stuff on the 18.01.2021

УДК 004.023 DOi 10.30837/ bi.2021.1(96).09

В.Є. Байдак[1], **О.О. Мазурова**[2], **О.Г. Ворочек**[3]

[1]магістрант кафедри програмної інженерії ХНУРЕ, м. Харків,
vadym.baidak@nure.ua, ORCID ID 0000-0001-8166-181X

[2] кандидат технічних наук, доцент, доцент кафедри програмної інженерії ХНУРЕ, м. Харків,
oksana.mazurova@nure.ua, ORCID ID 0000-0003-3715-3476

[3]кандидат технічних наук, доцент кафедри програмної інженерії ХНУРЕ, м. Харків,
olga.vorochek@nure.uamailto:oksana.mazurova@nure.ua, ORCID ID 0000-0002-9054-9894

РОЗРОБКА КОМБІНОВАНОГО МЕТОДУ ПОБУДОВИ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ОНЛАЙН-МАГАЗИНУ ЕЛЕКТРОННИХ ІГОР

Запропоновано комбінований метод побудови рекомендаційної системи для розширення функціоналу онлайн-магазину електронних ігор. Запропонований метод, що враховує аналітичний профіль користувача та оцінки інших користувачів, дозволить системі надавати більш точні персоналізовані рекомендації. Основою побудови рекомендаційної системи пропонується метод колаборативної фільтрації. Для покращення якості рекомендацій пропонується вибір вхідних даних для колаборативної фільтрації на базі кластеризації користувачів за допомогою методу k-means на основі аналітичного профіля користувача. Розроблений метод апробований під час створення рекомендаційної системи для онлайн-магазину електронних ігор.

ОНЛАЙН-МАГАЗИН, РЕКОМЕНДАЦІЙНА СИСТЕМА, АНАЛІТИЧНИЙ ПРОФІЛЬ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, КЛАСТЕРИЗАЦІЯ, K-MEANS

Байдак В.Е., Мазурова О.А., Ворочек О.Г. Разработка комбинированного метода построения рекомендательной системы для онлайн-магазина электронных игр. Предложен комбинированный метод построения рекомендательной системы для расширения функционала онлайн-магазина электронных игр. Предложенный метод, учитывающий аналитический профиль и оценки других пользователей, позволит системе предоставлять более точные персонализированные рекомендации. Основой построения рекомендательной системы предлагается метод колаборативной фильтрации. Для улучшения качества рекомендаций предлагается выбор входных данных для колаборативной фильтрации на базе кластеризации пользователей с помощью метода k-means на основе аналитического профиля пользователя. Разработанный метод апробирован при создании рекомендательной системы для онлайн-магазина электронных игр.

ОНЛАЙН-МАГАЗИН, РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА, АНАЛИТИЧЕСКИЙ ПРОФИЛЬ, КОЛЛАБОРАТИВНАЯ ФИЛЬТРАЦИЯ, КЛАСТЕРИЗАЦИЯ, K-MEANS

Baidak Vadym, Mazurova Oksana, Vorochek Olga.Development of a combined method for constructing a recommendation system for an online store of electronic games. A combined method for constructing a recommendation system for expanding the functionality of an online store of electronic games is proposed. The proposed method, which takes into account the analytical profile of the user and the assessments of other users, will allow the system to provide more accurate personalized recommendations. The method of collaborative filtration is proposed as the basis for building a recommendation system. To improve the quality of recommendations, it is proposed to select input data for collaborative filtering based on user clustering using the k-means method based on an analytical user profile. The developed method was tested when creating a recommendation system for an online store of electronic games.

ONLINE STORE, RECOMMENDATION SYSTEM, ANALYTICAL PROFILE, COLLABORATIVE FILTRATION, CLUSTERING, K-MEANS

1. Постановка проблеми

На сьогоднішній день, великий потік інформації, доступний людині, мимоволі ініціює роботу з пошуку і фільтрації даних в ньому. Важливим інструментом отримання інформації сучасною людиною є пошукові системи. Однак при відсутності чіткого розуміння конкретного запиту складно знайти необхідну інформацію. Цю проблему частково вирішують рекомендаційні системи (РС), що здатні знаходити об'єкти схожі на те, що подобається або потрібно користувачеві на основі інформації про нього та схожих користувачів. РС аналізують інтереси користувачів і намагаються передбачити, що саме буде найцікавіше для конкретного користувача в даний момент часу [1].

РС є критично важливими в деяких галузях, адже їх ефективне використання може принести величезний дохід або дозволить суттєво виділитися серед конкурентів. РС зазвичай використовуються в сфері комерції на сайтах продажу продуктів. Під продуктом мається на увазі товар або послуга, яку можна запропонувати для ринку, і яка буде задовольняти потреби споживачів [2]. В цій комерційній сфері РС може виявити потреби відвідувачів онлайн-системи і зробити цікаві саме їм пропозиції, збільшуючи дохід системи за рахунок зростання конверсії, середнього чека і частоти повторних покупок [3].

Найважливішою якістю будь-якої РС є точність рекомендацій, які повинні бути персоналізованими і відповідати вподобанням користувачів. РС, яка не володіє такою якістю, не тільки не буде приносити ніякої користі ані користувачам, ані власникам сервісів, а навіть може нанести фінансовий збиток. Таким

чином, існує потреба в розробці методів побудови рекомендаційних систем, які дозолять їм надавати точні персоналізовані рекомендації.

2. Аналіз основних досліджень

Традиційно для побудови РС використовуються підходи на базі колаборативної фільтрації, контентної фільтрації або гібридний підхід. Колаборативна фільтрація рекомендує елементи, визначаючи інших користувачів зі схожими вподобаннями. Контентна фільтрація співвідносить властивості контенту і характеристики користувача. Гібридний підхід поєднує в собі інструменти колаборативної і контентної фільтрації [4].

Колаборативний підхід є найбільш зрілим та реалізується найпоширеніше. Вхідними даними для колаборативної фільтрації зазвичай є набір оцінок усіх користувачів сервісу і це призводить до одного з її основних недоліків, так званої проблеми «білих ворон», що полягає у погіршенні рекомендацій певним категоріям користувачів. Деякі користувачі мають специфічні уподобання, які не узгоджується з уподобаннями інших користувачів [5]. Точність рекомендацій для таких користувачів знижується і оцінки таких користувачів можуть негативно вплинути на точність рекомендацій іншим категоріям користувачів [6].

Якщо користувачів кластеризувати за визначеним аналітичним профілем і використовувати в колаборативній фільтрації для надання рекомендацій данні оцінок користувачів одного кластеру, це дозволить збільшити точність рекомендацій колаборативної фільтрації.

3. Постановка задачі

Отже, була поставлена задача розробити комбінований метод побудови РС, що використовує метод колаборативної фільтрації і забезпечує найбільш повне використання всіх даних користувачів за рахунок використання кластеризації користувачів. Практичну апробацію методу необхідно провести на прикладі побудови рекомендаційної онлайн-системи продажу комп'ютерних ігор, та за її результатами розробити рекомендацій, щодо побудови РС, що найбільш адекватно відображають очікування відвідувачів подібних систем.

4. Розробка комбінованого методу побудови РС

Запропонований комбінований метод передбачає поєднання у собі алгоритмів колаборативної фільтрації для надання рекомендацій і k-means кластеризацію [1] на основі аналітичних профілів користувачів для поліпшення рекомендацій.

Вхідними даними для кластеризації пропонується використовувати характеристики користувачів нормовані до шкали від 1 до h, на основі яких створюються кластери «схожих» користувачів за допомогою алгоритму k-means:

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & \cdots & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & \cdots & \cdots & x_{mn} \end{pmatrix} \overset{k-means}{\Rightarrow} \begin{pmatrix} c_1 \\ c_2 \\ \cdots \\ c_k \end{pmatrix}$$

де x_{ij} – значення j-ої характеристики для i-го користувача, n – кількість характеристик, m – кількість користувачів, c_i – кластер отриманий в результаті роботи алгоритму k-means, k – кількість кластерів.

Після побудови кластерів «схожих» користувачів, для кожного кластера вирішено застосувати алгоритм машинного навчання на основі колаборативної фільтрації (факторизація матриці з градієнтним спуском [7]) для побудови рекомендаційних моделей. Отримані моделі, у свою чергу, будуть використовуватися для прогнозування оцінок продуктів користувачами:

$$\begin{pmatrix} c_1 \\ c_2 \\ \cdots \\ c_k \end{pmatrix} \overset{collaboration\ filtering}{\Rightarrow} \begin{pmatrix} m_1 \\ m_2 \\ \cdots \\ m_k \end{pmatrix}$$

де m_i – модель для побудови рекомендації для i-го кластеру.

Для вибору рекомендованих продуктів з вхідного набору (продукти, які користувач ще не оцінював) буде виконуватися наступне:

– вибір моделі відповідно до кластеру користувача;

– застосування моделі до користувача і кожного продукту з вхідного набору – отримання прогнозованих оцінок продуктів користувачем;

– вибір для рекомендації тих продуктів, прогнозована оцінка яких перевищує певний поріг (зазвичай береться 3,5).

Отже, вибір рекомендованих продуктів можна представити, як:

$$\begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_n \end{pmatrix} + u_{c_i} \overset{m_i}{\Rightarrow} \begin{pmatrix} r_1 \\ r_2 \\ \cdots \\ r_n \end{pmatrix} \overset{r_j > r_{threshold}}{\Rightarrow} \begin{pmatrix} p_f \\ \cdots \\ p_l \end{pmatrix}, \begin{pmatrix} p_f \\ \cdots \\ p_l \end{pmatrix} \in \begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_n \end{pmatrix}$$

де $\begin{pmatrix} p_1 \\ p_2 \\ \cdots \\ p_n \end{pmatrix}$ – вхідний набір продуктів для рекомендацій, n – кількість продуктів у вхідному наборі даних, u_{c_i} – користувач з кластеру c_i, для якого будуються рекомендації, m_i – рекомендаційна модель для кластеру c_i, $\begin{pmatrix} r_1 \\ r_2 \\ \cdots \\ r_n \end{pmatrix}$ – отримані прогнозовані значення оцінок продуктів користувачем, $r_{threshold}$ – поріг рекомендації

(зазвичай береться 3,5), $\begin{pmatrix} g_f \\ \cdots \\ \cdots \\ g_l \end{pmatrix}$ – продукти, обрані для рекомендації користувачу.

Кожен новий користувач повинен бути віднесений до існуючих кластерів, або кластери можуть бути перебудовані. Для нових користувачів рекомендації можуть будуватися на основі найпопулярніших продуктів в рамках його кластеру.

5. Моделювання колаборативної фільтрації для запропонованого методу

Представимо вхідний набір даних для колаборативної фільтрації у вигляді матриці *C*. Рядки матриці *C* відповідають за продукти, а стовпці за користувачів. Оцінки можуть приймати значення від 1 до 5, якщо користувач не оцінював продукт, то відповідна комірка ініціалізується як 0. Приклад матриці *C* наведений на рисунку 1.

	користувачі					
продукти	5	0	3	...	0	3
	0	4	0	...	2	4
	0	0	0	...	0	0
	⋮	⋮	⋮	⋱	⋮	⋮
	0	0	0	...	0	3
	5	0	3	...	0	0

Рис. 1. Матриця оцінок продуктів користувачами

Нехай наступний набір векторів описує *h* характеристик кожного продукту:

$$p^{(1)}, p^{(2)}, \dots, p^{(\text{кількість продуктів})}\ p^i \in \mathbb{R}^h$$

Інший набір векторів описує відношення користувача до кожної з характеристик:

$$u^{(1)}, u^{(2)}, \dots, u^{(\text{кількість користувачів})}\ u^i \in \mathbb{R}^h$$

Отже, вектори $u^{(j)}$ і $p^{(i)}$ мають однакову розмірність, а параметр $u_n^{(j)}$ відображає ставлення користувача *j* до характеристики $p_n^{(i)}$ продукту *i*. У такому випадку оцінку даного продукту цим користувачем можна розрахувати як $\left(u^{(i)}\right)^T p^i$. Надалі будемо позначати кількість користувачів як n_u, а кількість продуктів як n_p. Уявімо, що набір векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ відомий. В такому випадку можна знайти значення вектору $p^{(j)}$ за допомого мінімізації наступного функціоналу:

$$F = \frac{1}{2}\sum_{j:r(i,j)}\left(\left(u^{(j)}\right)^T p^i - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{l=1}^{n}\left(p_l^i\right)^2$$

де $r(i,j) = \begin{cases} 1, \text{якщо } j \text{ користувач оцінив } i \text{ продукт} \\ 0, \text{якщо не оцінив} \end{cases}$,

$\frac{\lambda}{2}\sum_{l=1}^{n}\left(p_l^i\right)^2$ – регулярізація

За допомогою цієї функції розраховується квадратична помилка того, на скільки відрізняються оцінки, які були отримані за допомогою параметрів u і p, в порівнянні зі справжніми значеннями. Отже, після мінімізації функціоналу F, ми отримаємо деякі характеристики $p^{(i)}$ для *i* продукту. Але нам потрібні характеристики для всіх продуктів, в цьому випадку функціонал буде мати наступний вигляд:

$$F\left(p^{(1)}, \dots, p^{(n_p)}\right) = \\ = \frac{1}{2}\sum_{i=1}^{n_p}\sum_{j:r(i,j)=1}\left(\left(u^{(j)}\right)^T p^i - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_p}\sum_{l=1}^{n}\left(p_l^i\right)^2 \quad (1)$$

Розглянемо зворотну ситуацію коли нам відомі характеристики продуктів $p^{(1)}, p^{(2)}, \dots, p^{(n_g)}$. Тоді значення векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ можна знайти за допомогою мінімізації наступного функціоналу:

$$F\left(u^{(1)}, \dots, u^{(n_u)}\right) = \\ = \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(\left(u^{(j)}\right)^T p^i - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{l=1}^{n}\left(u_l^j\right)^2 \quad (2)$$

Однак обидва набори векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ і $p^{(1)}, p^{(2)}, \dots, p^{(n_p)}$ нам невідомі. Існує два способи вирішення даної проблеми – за допомогою алгоритму найменших квадратів, що чергуються (Alternating Least Squares) або за допомогою методу факторизації матриці низького рангу (Low Rank Matrix Factorization) [7].

Можна обчислювати обидва набори u і p одночасно. З функціоналів (1) і (2), можна виявити, що вони обчислюють однакову суму за винятком регуляризації. Перший функціонал обчислює суму помилок для користувачів, які оцінили даний продукт. Другий функціонал для кожного користувача обчислює суму квадратних відхилень для тих продуктів, які були оцінені даним користувачем. Отже, обидва функціонала обчислюють суму відхилень для всіх пар користувач-продукт, які мають значення 1 в таблиці *r*, тобто даний користувач переглянув і оцінив даний продукт. Отримаємо наступний функціонал (функція вартості або функція втрат), який потрібно мінімізувати:

$$F\left(p^{(1)}, \dots, p^{(n_p)}, u^{(1)}, \dots, u^{(n_u)}\right) = \\ = \frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(\left(u^{(j)}\right)^T p^i - y^{(i,j)}\right)^2 + \quad (3) \\ + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{l=1}^{n}\left(u_l^j\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_p}\sum_{l=1}^{n}\left(p_l^i\right)^2$$

Вектори u і p ініціалізуються маленькими випадковими значеннями. Після того, як обчислені всі характеристики, можна отримати наступну матрицю прогнозувань:

$$\begin{bmatrix} \left(u^{(1)}\right)^T p^{(1)} & \left(u^{(2)}\right)^T p^{(1)} & \cdots & \left(u^{(n_u)}\right)^T p^{(1)} \\ \left(u^{(1)}\right)^T p^{(2)} & \left(u^{(2)}\right)^T p^{(2)} & \cdots & \left(u^{(n_u)}\right)^T p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \left(u^{(1)}\right)^T p^{(n_p)} & \left(u^{(2)}\right)^T p^{(n_p)} & \cdots & \left(u^{(n_u)}\right)^T p^{(n_p)} \end{bmatrix} \quad (4)$$

У цій матриці містяться прогнозування оцінок для всіх продуктів всіма користувачами. Наприклад, прогноз оцінки користувача *j* для гри *i* обчислюється як $\left(u^{(j)}\right)^T p^{(i)}$.

Приведемо матрицю (4) до векторизованого вигляду. Для цього введемо матриці *P* і *U*:

$$P=\begin{bmatrix} \cdots & \left(p^{(1)}\right)^T & \cdots \\ \cdots & \left(p^{(2)}\right)^T & \cdots \\ & \vdots & \\ \cdots & \left(p^{(n_p)}\right)^T & \cdots \end{bmatrix}$$

$$U=\begin{bmatrix} \cdots & \left(u^{(1)}\right)^T & \cdots \\ \cdots & \left(u^{(2)}\right)^T & \cdots \\ & \vdots & \\ \cdots & \left(u^{(n_u)}\right)^T & \cdots \end{bmatrix}$$

Матриця *P* містить характеристики усіх продуктів і має розмір n_p на *k*. Матриця *U* містить значення ставлення користувачів до характеристик продукту і має розмір n_u на *k*.

Таким чином, матрицю прогнозування можна подати наступним чином $C = P\left(U\right)^T$.

Матрицю *C* апроксимується за допомогою представлення її як добутку двох матриць. Метою цієї апроксимації є отримання передбачень для ігор, які ще не були оцінені користувачем, тобто нулі в початковій матриці повинні бути замінені на прогнозні значення. Досягти цієї мети допомагає той факт, що мінімізуємий функціонал враховує помилки тільки для тих комірок матриці, для яких вже були проставлені оцінки.

6. Оптимізація градієнтного спуску

Для обчислення наборів векторів $u^{(1)},u^{(2)},\ldots,u^{(n_u)}$ і $p^{(1)},p^{(2)},\ldots,p^{(n_p)}$ пропонується використовувати метод факторизації матриці низького рангу як більш простий в реалізації. Цей метод передбачає вирішення задачі мінімізації функції витрат (3).

Існує декілька алгоритмів призначених для вирішення задачі мінімізації, але одним з найпопулярніших є алгоритм градієнтного спуску. Суть даного алгоритму полягає в тому, що параметри функції втрат оновлюються в протилежному напрямку градієнта даної функції. Тобто, якщо при поточних параметрах функція зростає (градієнт додатний), то її мінімум знаходиться зліва, і додатний градієнт віднімається. Якщо градієнт від'ємний, то мінімум функції знаходиться праворуч і ми віднімаємо від'ємний градієнт:

$$\omega = \omega - \lambda \times \nabla_\omega F\left(\omega\right) \qquad (5)$$

де ω – параметри, $J\left(\omega\right)$ – функція втрат, λ – коефіцієнт швидкості навчання (learning rate).

З формули (5) можна побачити, що параметри оновлюються на величину кратну градієнту. Коефіцієнт швидкості навчання використовується для регуляції швидкості сходження алгоритму та його стабільності. Якщо значення коефіцієнту занадто велике, то алгоритм може розходитися, а якщо занадто маленьке, то алгоритм може сходиться дуже повільно.

Градієнт для векторів $p^{(1)},p^{(2)},\ldots,p^{(n_g)}$ при функції втрат (3) обчислюється наступним чином:

$$\frac{\partial Y}{\partial p_k^{(i)}} = \sum_{j:r(i,j)=1}\left(\left(u^{(j)}\right)^T p^i - y^{(i,j)}\right)u_k^{(j)} + \lambda p_k^{(i)}$$

де λ – коефіцієнт регулярізації.

Для набору $u^{(1)},u^{(2)},\ldots,u^{(n_u)}$:

$$\frac{\partial Y}{\partial u_k^{(j)}} = \sum_{i:r(i,j)=1}\left(\left(u^{(j)}\right)^T p^i - y^{(i,j)}\right)p_k^{(i)} + \lambda u_k^{(j)}$$

де λ – коефіцієнт регулярізації.

Алгоритм градієнтного спуску складається з трьох кроків:

а) задаються початкові значення для параметрів (для колаборативної фільтрації беруться випадкові маленькі значення) і точність розрахунку ε;

б) за формулою (5) обчислюються оновлені значення параметрів;

в) перевіряється умова зупинки алгоритму. Наприклад, алгоритм припиняє роботу, якщо функція втрат змінилася на величину меншу, ніж ε, інакше переходить в пункт б).

Такий алгоритм називається алгоритмом пакетного градієнту (Batch Gradient Descent). Але цей алгоритм не застосовується коли необхідно працювати з великими даними, тому що необхідно робити обчислення для всього набору даних для кожного оновлення. Однак, існує альтернативний алгоритм, який має назву алгоритм стохастичного градієнта (Stochastic Gradient Descent). В цьому алгоритмі оновлення параметрів виконується для кожної пари вхідних даних. Отже, замість формули (5) застосовується формула:

$$\omega = \omega - \lambda \times \nabla_\omega F\left(\omega, x^{(i)}; y^{(i)}\right) \qquad (6)$$

де ω – параметри, $(x^{(i)}; y^{(i)})$ – приклад з вхідного набору даних.

Градієнтний спуск сходиться в локальний мінімум і це є його значним недоліком. Потрапляння в локальний мінімум може істотно погіршити якість прогнозів і оскільки функція втрат (3.3) має безліч локальних мінімумів, то цей недолік є суттєвим. Спробувати вирішити цю проблему можна намагаючись запустити алгоритм декілька разів, щоб параметри були ініціалізовані в такому місці, де можливо уникнути локальні мінімуми. Але на це може піти величезна кількість спроб, враховуючи специфіку

функції втрат, і не має гарантії, що алгоритм зійшовся саме в глобальний мінімум.

Однак, можливо уникнути проблеми локальних мінімумів, оптимізувавши класичний алгоритм градієнтного спуску. До найбільш ефективних і простих оптимізацій відноситься оптимізація, яка носить назву «Імпульс» (Momentum) [7].

Оптимізація типу імпульс потребує заміну формули (6) на формули:

$$v_t = 0.9 \times v_{t-1} - \lambda \times \nabla_\omega F\left(\omega, x^{(i)}; y^{(i)}\right)$$

$$\omega = \omega - v_t$$

Імпульс для кожного параметра накопичується в векторі v. Отже, для мінімізації функції витрат (3) пропонується використовувати описаний оптимізований алгоритм градієнтного спуску.

7. Моделювання кластеризації для розбиття користувачів на групи

Для підвищення ефективності рекомендаційної системи та поліпшення якості рекомендацій пропонується використовувати кластеризацію користувачів. Характеристики користувача повинні бути виражені в чисельних показниках і представлені масивом чисел від 1 до n після нормування. Для кластеризації даних подібного типу найчастіше використовують алгоритм k-means. [8].

Даний алгоритм розділяє безліч вхідних об'єктів на задане число кластерів. Алгоритм починає свою роботу з ініціалізації центрів кластерів (центроїдів):

$$\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$$

де k – кількість кластерів, n – розмірність кожного об'єкта.

Кожен елемент множини відноситься до кластеру з найближчим центром. Наступні етапи повторюються до тих пір, поки алгоритм не зійдеться.

Далі заповнюється вектор c. Цей процес записується наступним чином – for $i = 1$ to l:

$$c^{(i)} = \left\{ p : x^{(i)} - \mu_p{}^2 \le x^{(i)} - \mu_j{}^2 \forall j, 1 \le j \le K \right\} \qquad (7)$$

де $x^{(i)}$ – об'єкт з набору вхідних даних; c – вектор, який містить відповідність між об'єктом $x^{(i)}$ і кластером $c^{(i)}$, до якого він належить; l – кількість об'єктів множини вхідних даних.

З формули (7) виходить, що кожен об'єкт з набору вхідних даних відноситься до кластеру з найближчим центром.

На наступному етапі оновлюється положення центроїдів – for $k = 1$ to K:

$$\mu_k = \frac{1}{S_k} \sum_{x^{(j)}: c^{(j)} = k} x^{(j)} \qquad (8)$$

З формули (8) виходить, що значення центру має безлічі вхідних об'єктів присвоюються кожному центру кластера k, до якого належать ці об'єкти. Якщо не відбулося зміни положення центрів кластерів після поточної ітерації, то алгоритм вважається завершеним.

Однак у k-means алгоритму є ряд недоліків:

– число кластерів визначається заздалегідь;

– не гарантується досягнення глобального мінімуму;

– може відбуватися різне розбиття для однакових вхідних даних при різних початкових положеннях центрів кластерів.

Розглянемо підходи до мінімізації даних недоліків. Введемо функцію втрат для оцінювання якості розбиття:

$$F\left(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K\right) = \frac{1}{m} \sum_{i=1}^{m} x^i - \mu_{c^{(i)}}{}^2$$

Наведений функціонал обчислює суми квадратичних відстаней від центрів кластерів до об'єктів відповідних цим кластерам. Метою алгоритму оптимізації кластеризації є мінімізація даного функціоналу.

Розглянемо перший недолік – число кластерів визначається заздалегідь. Цей недолік можна мінімізувати, використавши метод ліктя. За цим методом розробник самостійно обирає кількість кластерів і оцінює зменшення функції втрат, при збільшенні кількості кластерів.

Другий недолік полягає у відсутності гарантії досягнення глобального мінімуму. Даний недолік можливо мінімізувати за допомогою множинної ініціалізації. Алгоритм k-means ініціалізується різними наборами центроїдів, для кожного набору оцінюється значення функції витрат – обирається той варіант, при якому функція втрат має найменше значення.

8. Використання комбінованого методу побудови рекомендаційної системи в онлайн-магазині електронних ігор

За допомогою розробленого комбінованого методу була побудована рекомендаційна система для онлайн-магазину електронних ігор.

Для аналітичного профілю користувача, на основі якого відбувається кластеризація, були обрані наступні характеристики – вік, стать, улюблені жанри ігор. Для жанру були виділені кількісні характеристики, які можна перенести на користувача, у якого цей жанр є улюбленим. До цих характеристик були віднесені – рівні навичок, екшену, стратегічності, сюжету, зрілості, насилля та освіти. На рис. 2 наведено схему бази даних для онлайн-магазину електронних ігор. Для роботи РС потрібні такі сутності бази даних, як гра, жанр, користувач, оцінка гри користувачем.

Вхідні дані (характеристики користувачів та ігор, оцінки ігор користувачами) для розробленого комбінованого методу були отримані за допомогою API найкрупнішої онлайн-системи продажу ігор – Steam.

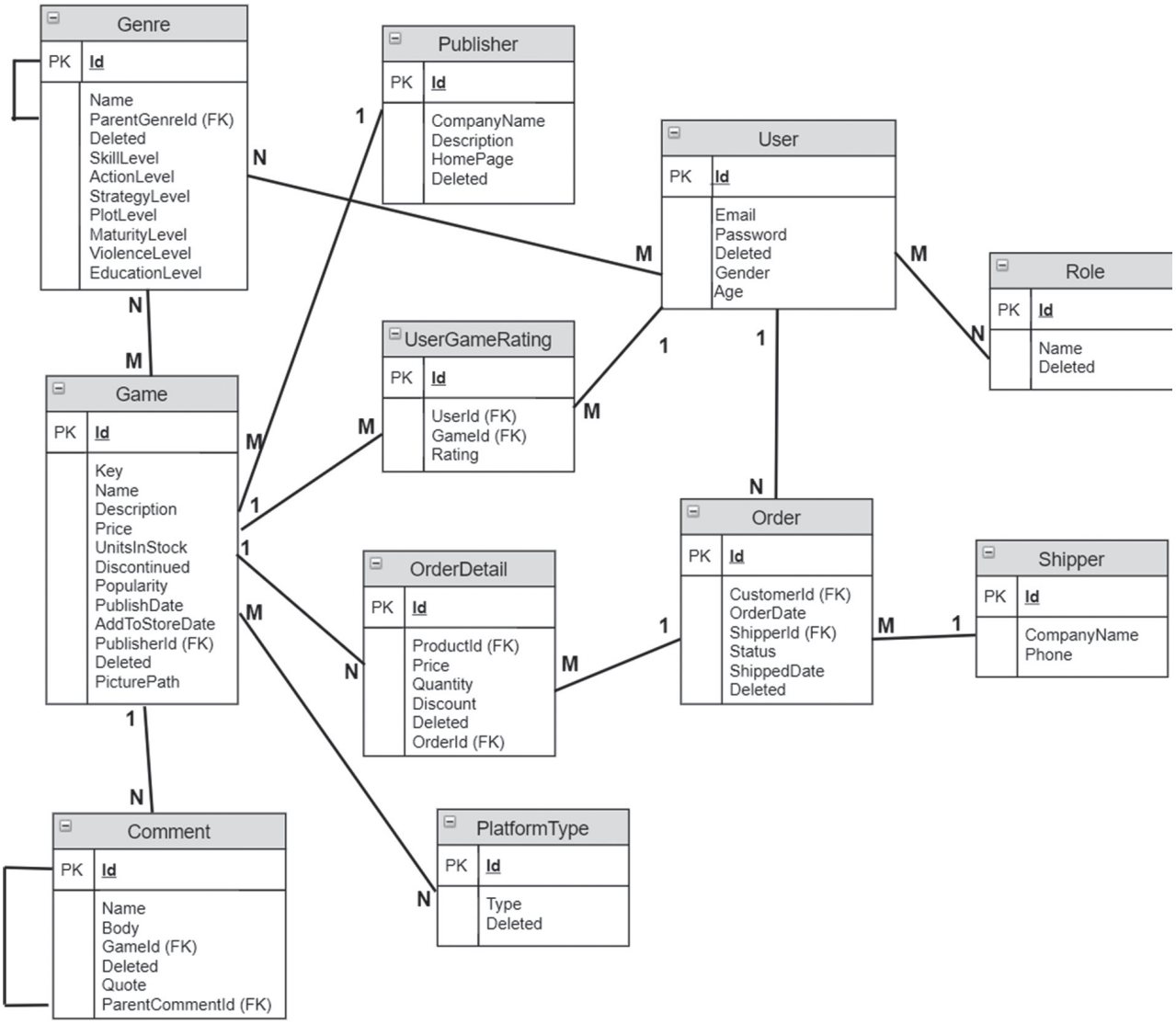


Рис. 2. Схема бази даних онлайн-магазину електронних ігор

Оскільки Steam API не надає інформацію про оцінки ігор користувачами, значення кількості зіграних годин були використані для конвертації у оцінку.

Для оцінки ефективності розробленої на базі запропонованого методу РС та точності рекомендацій РС використовувалася метрика RMSE (Root of Mean Squared Error) [6]. RMSE використовується для вимірювання різниці між передбачуваними значеннями моделі РС та спостережуваними значеннями тестового набору даних. Чим цей показник нижче, тим ефективніша модель і точніші рекомендації.

Для вибору кількості кластерів для розробленої РС була проведена низка експериментів з замірюванням RMSE показника та показника R-квадрат. Показник R-квадрат вказує наскільки дані відповідають моделі. Значення цього показника 0 означає, що дані є випадковими та не можуть відповідати моделі. Значення 1 — що модель точно відповідає даним. Показники RMSE та R-квадрат для різної кількості кластерів наведені у табл. 1.

При кількості кластерів більше 6, показник R-квадрат все більше зменшувався, тому в табл. 1 наведені результати експериментів тільки для кластерів від 2 до 6. Для розробленої РС було обрано 5 кластерів для розбиття, оскільки при цій кількості кластерів РС показує найменший показник RMSE та найбільший показник R-квадрат. Також було проведене порівняння показника RMSE РС, яка використовує колаборативну фільтрацію, та розробленої РС, яка використовує колаборативну фільтрацію на основі методу факторизації матриці низького рангу та k-means кластеризацію для розбиття користувачів на групи за аналітичним профілем. В табл. 2 наведене порівняння показника RMSE для наведених РС для користувачів з різною кількістю оцінок.

Таблиця 1

Показники	Кількість кластерів				
	2	3	4	5	6
RMSE	0.96	0.96	1.11	0.78	1.01
R-квадрат	0.54	0.53	0.48	0.72	0.32

Таблиця 2

RMSE для РС	Користувачі (кількість оцінок)				
	16	**42**	**84**	**112**	**166**
з *k*-means кластеризацією та колаборитивною фільтрацією	0.82	0.88	0.78	0.93	0.99
з колаборитивною фільтрацією	1.53				

RMSE для РС з колаборативною фільтрацією має одне і теж значення для різних користувачів оскільки використовується одна і та сама рекомендаційна модель для всіх користувачів. У випадку РС з k-means кластеризацією, рекомендаційна модель відрізняється в залежності від того до якого кластеру був віднесений користувач. Як можна побачити з табл. 2, показники RMSE для РС з k-means кластеризацією нижчі (в деяких випадках майже в 2 рази) ніж для РС, яка використовує тільки колаборативну фільтрацію. Отже розроблена система будує більш ефективні рекомендаційні моделі і робить точніші рекомендації.

Розроблена рекомендаційна система надає точні персоналізовані рекомендації ігор на основі оцінок цільового користувача та інших користувачів системи з використанням кластеризації користувачів для підвищення точності рекомендацій для користувачів в рамках кластеру.

Розроблена РС реалізовує наступні функції:

– додавання користувачем оцінок та відгуків для ігор;

– створення профілю користувача з інформацією про нього;

– додавання користувачем інформації про улюблений жанр, платформу, видавця;

– кластеризація користувачів на основі профілів користувачів за допомогою методу k-means;

– тренування моделі колаборативної фільтрації для окремих кластерів користувачів;

Розроблена система базується на фреймворці для створення веб-додатків ASP .NET MVC 5 та на бібліотеці машинного навчання ML.NET. Для доступу до реляційних даних використовується СУБД MS SQL Server 2016, ORM Entity Framework та FluentAPI. Інтерфейс програмної системи дозволяє зручно оцінювати ігри, переглядати загальні середні оцінки ігор, історію оцінок та персоналізовані рекомендації.

9. Висновки та перспективи

Розроблений комбінований метод побудови рекомендаційної системи дозволяє підвищити точність персоналізованих рекомендацій (зменшити RMSE показник рекомендаційних моделей) та був покладений в основу рекомендаційної системи онлайн-магазину електронних ігор. Поліпшення рекомендацій зумовлено використання кластеризації користувачів в запропонованому комбінованому методі. Розроблена рекомендаційна система, завдяки поліпшеній якості рекомендацій, буде корисна як користувачам онлайн-магазину електронних ігор, оскільки буде надавати їм релевантні пропозиції і зменшувати час на пошук бажаного, так і власникам онлайн-магазину, оскільки вона може збільшити дохід за рахунок ефективності.

Список літератури:

[1] Isinkaye F.O., Folajimi Y.O., Ojokoh B.A., Recommendation systems: Principles, methods and evaluation, Egyptian Informatics Journal, Volume 16, Issue 3, 2015, Pages 261-273

[2] Chalyi S., Leshchynskyi V Temporal modeling of user preferences in recommender system Chalyi S., Leshchynskyi V. CEUR Workshop Proceedings, 2020, 2711, с. 518-528

[3] Блинков Н. Объем рынка компьютерных игр URL: http://www.it-weekly.ru/market/business/73058.html (дата звернення: 25.01.2021)

[4] Doo Hyodan, Audrey Germain, Geordan Jove Recommendation System for Steam Game Store: An overview of recommender systems URL: https://audreygermain.github.io/Game-Recommendation-System/ (дата звернення: 29.01.2021)

[5] Serhii Chalyi, Volodymyr Leshchynskyi. Method of constructing explanations for recommender systems based on the temporal dynamics of user preferences. «EUREKA: Physics and Engineering».-2020.- Number 3.-p.43-50.

[6] Jena K. C., Mishra S., Sahoo S. and Mishra B. K., Principles, techniques and evaluation of recommendation systems, International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2017, pp. 1-6

[7] Yuan Lu, Jie Yang, Notes on Low-rank Matrix Factorization, URL: https://yangjiera.github.io/pdf/low-rank.pdf (дата звернення: 12.02.2021)

[8] Madushan Dileka Introduction to K-means Clustering URL: https://medium.com/@dilekamadushan/introduction-to-k-means-clustering-7c0ebc997e00 (дата звернення: 12.02.2021)

Надійшла до редколегії 11.03.2021

УДК 004.4; 004.75 DOi 10.30837/ bi.2021.1(96).10

Н.С. Кравець[1], М.С. Чернишов[2]

[1]Кандидат технічних наук, доцент кафедри Програмної інженерії,
Харківський національний університет радіоелектроніки,
natalia.kravets@nure.ua, ORCID iD: 0000-0002-6753-3333

[2]Студент кафедри Програмної інженерії,
Харківський національний університет радіоелектроніки,
mykhailo.chernyshov1@nure.ua, ORCID iD: 0000-0003-1974-730X

ПРОБЛЕМИ РОЗГОРТАННЯ ТА КЕРУВАННЯ МУЛЬТИХМАРНИМИ РІШЕННЯМИ

Проаналізовано доцільність розгортання програмних продуктів у мультихмарі, визначено переваги та недоліки такого підходу, проблеми, пов'язані із використанням сервісів, що надаються кількома хмарними провайдерами. Досліджено, які хмарні рішення є на сьогодні найпопулярнішими, та розглянуто випадки, у яких краще використовувати саме їх, а не конкурентів. Розглянуто платформи мультихмарного управління, орієнтовані на використання організаціями з великою кількістю територіально рознесених відділів та офісів. Визначено, які фреймворки оркестрування хмарних ресурсів підтримують роботу із кількома хмарами, та за допомогою яких технологій ними здійснюється оптимальний вибір хмарного рішення.

МУЛЬТИХМАРА, ХМАРНИЙ ПРОВАЙДЕР, ФРЕЙМВОРК ОРКЕСТРУВАННЯ ХМАРНИХ РЕСУРСІВ, ПЛАТФОРМА МУЛЬТИХМАРНОГО КЕРУВАННЯ

Кравец Н.С., Чернишов М.С. Проблемы развертывания и управления мультиоблачными решениями. Проанализирована целесообразность развертывания программных продуктов в мультиоблаке, определены преимущества и недостатки такого подхода, проблемы, связанные с использованием сервисов, предоставляемых несколькими облачными провайдерами. Исследовано, какие облачные решения являются сегодня самыми популярными, и рассмотрены случаи, в которых лучше использовать именно их, а не конкурентов. Рассмотрены платформы мультиоблачного управления, ориентированные на использование организациями с большим количеством территориально разнесенных отделов и офисов. Определено, какие фреймворки оркестрации облачных ресурсов поддерживают работу с несколькими облаками, и с помощью каких технологий ими осуществляется оптимальный выбор облачного решения.

МУЛЬТИОБЛАКО, ОБЛАЧНЫЙ ПРОВАЙДЕР, ФРЕЙМВОРК ОРКЕСТРАЦИИ ОБЛАЧНЫХ РЕСУРСОВ, ПЛАТФОРМА МУЛЬТИОБЛАЧНОГО УПРАВЛЕНИЯ

Kravets N.S., Chernyshov M.S. Problems of multi-cloud solutions deploying and managing. The expediency of deploying software products in a multi-cloud is analyzed, the advantages and disadvantages of this approach, the problems associated with the use of services by several cloud providers are identified. It was investigated which cloud solutions are the most popular today and the cases in which it is better to use them, rather than competitors are considered. Multi-cloud management platforms are considered, which are focused on the use of organizations with a large number of geographically dispersed departments and offices. It has been determined what cloud resource orchestration frameworks support working with multiple clouds, and which technologies they use to make the best choice of a cloud solution.

MULTI-CLOUD, CLOUD PROVIDER, CLOUD RESOURCE ORCHESTRATION FRAMEWORK, MULTI-CLOUD CONTROL PLATFORM

Вступ

Надзвичайної популярності на сьогоднішній день набули хмарні технології, які дозволяють ефективно вирішувати проблеми, що постають перед компаніями, котрим необхідно розгортати та швидко запускати програмні продукти. Такі рішення часто дозволяють значно заощадити кошти, сплачуючи лише за використані ресурси в тому обсязі, у якому їх необхідно задіяти, а також надають можливість автоматичного масштабування на вимогу використовуваних ресурсів у реальному часі при зміні навантаження.

Зараз існує велика кількість хмарних провайдерів, що надають різноманітні послуги у хмарі: розгортання веб-застосунків, машинне навчання, штучний інтелект, сховища даних тощо. Найбільшу частку на цьому ринку мають Amazon Web Services, MicrosoftAzure та Google Cloud Platform.

Варто зазначити, що кожна із хмар має власний
- набір послуг, що надаються,
- розцінки,
- поширеність у різних регіонах світу.

Тому досить часто при виборі провайдера доводилося знаходити компроміс — яка сукупність характеристик є найбільш прийнятною. Проте, насправді, немає необхідності миритися, наприклад, із відносно високою ціною орендованих віртуальних машин одного із провайдерів, якщо він при цьому має сервіси, що дозволяють значно швидше за конкурентів обробляти великі дані, котрі надходять із розумних пристроїв компанії. Оптимальним рішенням є взаємодія із різними хмарами, при чому із використанням у кожній з них тих сервісів, які найкраще підходять для вирішення конкретної задачі клієнта. Даний підхід отримав назву «мультихмара».

Мультихмари можуть спростити доступ до даних і їх аналіз, скоротити витрати, підвищити безпеку даних. Але розгортання як і керування рішенням у декількох хмарах набагато складніше організувати.

1. Аналіз ефективності використання мультихмари

За даними компанії Gartner [1], 81% компаній, які наразі використовують хмарні рішення, працюють із мінімум двома провайдерами одночасно. Згідно із дослідженням IDC, до 2018 року мультихмарну стратегію планували розвивати 85% IT-відділів великих організацій. Слід зазначити, що до 2015 року цей показник складав 10%. Аналітики IDC описують мультихмару як «організаційну стратегію або архітектурний підхід до проектування складної цифрової послуги, яка включає в себе споживання хмарних послуг від більш ніж одного постачальника хмарних послуг». Це можуть бути безпосередньо конкуруючі хмарні сервіси, такі як сховище даних від декількох постачальників загальнодоступних хмарних послуг або IaaS і SaaS від одного або декількох постачальників хмарних послуг. [2] Використання мультихмари обмежується вартістю та складністю, пов'язаною із забезпеченням узгодженого керування та управління множиною різних варіантів хмарних обчислень. Згідно даних IDC 2020 року 37 відсотків організацій використовують в IT-інфраструктурі мультихмарні рішення (рис. 1). [2]

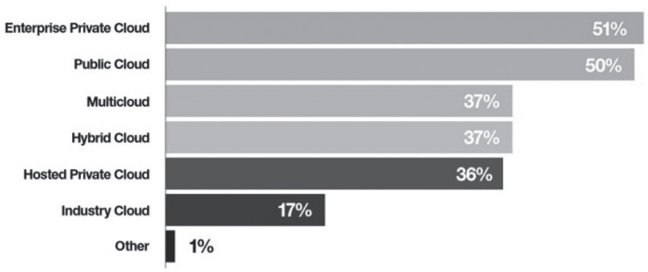


Рис. 1. Використання різних типів хмар в IT-інфраструктурі організацій

Мультихмарні рішення усе частіше використовуються у випадках, коли не вистачає власних потужностей для повноцінного функціонування програмної системи, а розгортання рішення у одній публічній хмарі не є можливим.

Важливою перевагою такого підходу також є незалежність від єдиного провайдера. Банкрутство такого постачальника послуг чи блокування його роботи із політичної чи інших причин, «гра не за правилами» у разі стику власних інтересів із інтересами хмарного провайдера можуть мати катастрофічні наслідки для клієнта.

Тому навіть використання аналогічних за функціоналом сервісів різних постачальників послуг у різних регіонах може виявитися ефективним — у разі припинення роботи одного із них, клієнти веб-ресурсу зможуть використовувати потужності іншого, хоч і з дещо більшою затримкою.

2. Використання сервісів кількох хмарних провайдерів на практиці

До того ж, немає універсальної хмари, яка може задовольнити потреби кожного із клієнтів. Для великої кількості дійсно великих багатофункціональних проектів мультихмарний підхід є єдино можливим. Наведемо приклад. Великий інтернет-магазин, серверна частина якого написана із використанням веб-фреймворку Django, який традиційно гарно інтегрується із СКБД PostgreSQL, і база даних якого вже містить велику кількість даних (тому міграція до нового рішення для підтримки сумісності призведе до переписування невиправдано великих обсягів коду) прагне впровадити найсучасніші технології штучного інтелекту і стати першим застосунком свого типу, що змінить формат взаємодії користувача із інтернет-магазином. Планується проводити верифікацію користувачів за допомогою голосу, здійснювати взаємодію із клієнтом таким чином, наче він спілкується із людиною. Традиційний підхід до роботи із сайтом також має зазнати змін: при вводі запитів до поля пошуку має використовуватися автодоповнення для зменшення кількості дій, що має виконувати користувач, проводитиметься аналіз та виправлення помилок, аби упевнитися у тому, що користувач дійсно знайде те, що мав намір придбати. Так як сайт уже є надзвичайно популярним, то виправданою буде підтримка максимально можливої кількості мов для охоплення якомога більшої аудиторії, при чому взаємодія із системою відбуватиметься у найбільш зручному для них форматі.

Використання однієї хмари у даному випадку є недоцільним. Якщо немає принципової різниці, де розгортати серверну частину за стосунку (хоча, насправді, найзручнішим і найуніверсальнішим рішенням є Amazon EC2), то для розміщення у хмарі бази даних Postgres варто використовувати Microsoft Azure, адже аналоги, наприклад, від Amazon та Google, не надають таких послуг. Коли мова йде про використання штучного інтелекту, то розглянемо рішення від Amazon, Microsoft, Google та IBM. Верифікацію за голосом підтримують лише перші два із них, найбільшу кількість мов дозволяє розпізнавати GoogleCloudAPI, автодоповнення вводу підтримується виключно Azure, а виявлення синтаксичних помилок не надається AWS.

Із проаналізованих вище рішень видно, що є такі задачі, що підтримуються лише частиною хмарних провайдерів, або ж взагалі одним. Варто зазначити, що у вибірці були найвідоміші із них, коли мова йде про більш дрібних представників, то ситуація є ще набагато складнішою. Хоча більшість знайдених рішень для даної проблеми підтримує Azure, проте не для всіх із них дане рішення є оптимальним, тому варто частину сервісів розгорнути із його використанням, а при реалізації інших — звернути увагу на більш конкурентноспроможні аналоги.

Отже, на прикладі було обґрунтовано ефективність та доцільність використання саме мультихмарного підходу. Але окрім вказаних вище переваг, такий підхід містить один найважливіший недолік — складність реалізації. Так як мультихмарність передбачає об'єднання в одну систему сервісів від різних платформ, то не усі із них можуть взаємодіяти один із одним, а значить, і бути інтегрованими в єдину систему. Кожна платформа має власні стандарти. Крім того, використання різних хмар може збільшити комунікаційні затримки, адже доводиться звертатися до кожної із них окремо, що, зрозуміло, відбуватиметься довше, ніж взаємодія із ресурсом у тій самій хмарі. Це означає, що більша швидкодія іншого рішення може бути зведена нанівець затримками комп'ютерної мережі. Різні платформи також надають різні рішення із забезпечення безпеки, тому необхідно визначити, чи не буде система вразливою при використанні сукупності різнорідних рішень. Більшість інструментів розгортання, налаштувань та викликів до застосунків є розрізненими, такими, що досить складно поєднуються, тому фахівцям у галузі інформаційних технологій доводиться великі проміжки часу виділяти для узгодження таких технологій [3].

Таким чином, використання сервісів низки хмарних провайдерів одночасно в рамках однієї системи дозволяє підвищити її ефективність або взагалі робить можливим її функціонування у спроектованому вигляді, проте застосовувати такий підхід потрібно лише тоді, коли це дійсно необхідно, адже інтеграція кількох різнорідних компонентів може призвести до підвищення складності програмної системи в цілому, а також до її вразливості.

3. Порівняння рішень найпопулярніших хмарних провайдерів

Порівняємо переваги популярних публічних хмарних платформ від Amazon, Microsoft, Google та Oracle і таким чином визначимо типи задач, що найефективніше вирішуються кожною із них, на основі чого можна вирішити, яка комбінація хмар є оптимальною для кожного окремо узятого проекту.

Коли мова йде про AWS, то дана платформа пропонує найбільшу кількість хмарних продуктів, так як вона є найстарішою і активно розвивається, пропонуючи клієнтам нові рішення. Обчислювальні сервіси є гарно масштабованими, мають високий рівень безпеки, працюючи у повністю захищеному мережевому середовищі. Серед недоліків варто зазначити досить високу ціну за послуги, що надаються, AWS Glacier має порівняно простішу функціональність, ніж інструменти резервного копіювання інших платформ.

Microsoft Azure гарно оптимізований для вирішення задач, пов'язаних із машинним навчанням та штучним інтелектом. Сервіс Data Lake має багато додаткових функцій для аналітики великих обсягів даних.

Обчислювальні сервери Google Cloud мають більш прозору у порівнянні із аналогами посекундну оплату оренди обчислювальних потужностей, інфраструктура є більш екологічно безпечною та забезпечується ощадливіше використання електроенергії у центрах обробки даних.

Oracle поки надає доступ до відносно малої кількості сервісів, проте такі рішення є досить дешевими, якщо порівнювати, наприклад, із пропозиціями Amazon. До того ж, дана хмара відома надзвичайно оптимальним автономним рішенням DBaaS (база даних як сервіс). База даних Oracle є більш досконалим рішенням, має об'єктно-реляційну модель, що дозволяє зберігати складні бізнес-дані.

AWS, Oracle, GoogleCloudPlatform (GCP) забезпечують автоматичне керування версіями, що наразі не підтримується Azure. AWS та Azure мають велику кількість інструментів моніторингу та керування, в той час коли GCP та Oracle не можуть цим похизуватися [4].

Важливою категорією порівняння хмар є кількість центрів обробки даних (ЦОД) та їх поширеність у різних регіонах планети. Це характеризує швидкість, із якою клієнти зможуть взаємодіяти із платформою, затримки в мережі. З цієї точки зору були проаналізовані хмари AWS, Alibaba Cloud, Azure та GCP. Alibaba Cloud має дуже велику концентрацію data-центрів на Дальньому Сході, насамперед у Китаї, та в Океанії, Azure — у Європі, на півдні Азії та в Австралії, а GoogleCloud та AWS — в Північній Америці. Таким чином, якщо розгортати програмні продукти у зазначених вище регіонах у відповідних хмарах, це буде оптимальним рішенням для обслуговування цільової аудиторії, що там знаходиться [5].

Відомо, що великої популярності зараз набуває машинне навчання. Саме тому його засоби намагаються інтегрувати як у велику кількість вже давно функціонуючих програмних систем, так і у такі, що лише проектуються. Відповідно, користуються попитом і хмарні сервіси, що надають відповідні інструменти. Було розглянуто та проаналізовано рішення машинного навчання чотирьох хмар: AWS, Microsoft Azure, GCP та IBM cloud. У таблиці 1 наведено порівняння ML-сервісів даних платформ.

Таблиця 1

Порівняння сервісів машинного навчання

	AWS	**Azure**	**GCP**	**IBM cloud**
Класифікація	+	+	+	+
Регресія	+	+	+	+
Кластеризація	+	+	+	-
Виявлення аномалій	+	+	-	-
Розмітка даних	+	+	+	+
Надання рекомендацій	+	+	+	-
Ранжування	+	+	-	-
Вбудовані алгоритми	+	+	+	-

Як видно із таблиці, усі основні види послуг, що були виділені для розгляду, підтримуються AWSta Azure, а в IBM-cloud відсутня підтримка більшості з них. 2021 року компанією Amazon було представлено SageMakerStudio — інтегроване середовище розробки для машинного навчання. Цей інструмент дозволяє інтегрувати усі наявні у даній хмарі засоби цієї галузі. Подібні рішення мають і конкуренти: Azure Machine Learning у Microsoft, Google Cloud AutoML у Google, IBM Watson Machine Learning Studioy IBM. Також — крім GCP, Keras — крім Azure. У AWS можна використовувати MXNet, Gluon, Chainer, Torch, у Azure – Spark ML, Microsoft Cognitive Toolkit, у GCP — XGBoost, у IBM-cloud — Spark MLib, XGBoost, PMML, IBM SPSS.

Також було проаналізовано програмні інтерфейси, призначені для обробки людського мовлення у текстовому та аудіоформаті. Результати порівняння наведено у таблиці 2.

Таблиця 2

Порівняння API машинного навчання

	AWS	**Azure**	**GCP**	**IBM cloud**
Розпізнавання мовлення	+	+	+	+
Перетворення тексту в мовлення	+	+	+	+
Виділення ключових фраз	+	+	+	+
Перевірка орфографії	-	+	-	-
Автодоповнення	-	+	-	-
Верифікація за голосом	+	+	-	-
Виділення метаданих	-	-	-	+
Аналіз відношень	-	+	-	+
Синтаксичний аналіз	-	+	+	+
Аналіз особистості	-	-	-	+
Обробка аудіо низької якості	+	+	+	+
Фільтрація неприпустимого вмісту	-	+	+	-
Аналіз настрою	+	+	+	+
Тегування частин мовлення	-	+	+	-
Інструменти для розробки чат-ботів	+	+	+	+

Отримані дані ще раз підтверджують тезу про те, що немає такої хмари, котра підтримувала б усі рішення, які можуть знадобитися при реалізації проекту, і говорять про необхідність використання мультихмари у разі такої потреби. За даними таблиці, Azure пропонує найбільшу кількість інструментів, проте для виділення метаданих та аналізу особистості доведеться використовувати рішення іншої хмари, із розглянутих провайдерів таке рішення має лише IBM.

Також різняться можливості застосовувати розпізнавання мовлення, що надаються провайдерами-лідерами хмарного ринку(Таблиця 3), тому варто ознайомитися із списком мов, що підтримуються кожним рішенням [6].

Таблиця 3

Підтримка хмарними провайдерами розпізнавання мовлення

	AWS	Azure	GCP	IBM cloud
Кількість підтримуваних мов	100+	120	120+	60+
Кількість підтримуваних мов для перекладу	6	60+	100+	48

4. Платформи мультихмарного керування

Коли було здійснено вибір засобів, що використовуватимуться із кожної хмари в межах мультихмари, необхідно подумати про їх узгодження. AWS, Microsoft Azure, GCP та IBM cloud надають можливість створення розподіленого мультихмарного додатка на основі мікросервісів.

У результаті дослідження ринку, було розглянуто декілька платформ мультихмарного керування.

Перша із них — платформа мультихмарного керування від IBM, яка забезпечує централізоване керування хмарами і послугами на підприємстві. Це надає можливість використовувати різні хмари та узгоджено керувати ними в середовищі самообслуговування та співробітництва. Платформа містить у своєму складі інструмент IBM Multicloud Manager[7], він дозволяє контролювати різні кластери Kubernetes як у приватній, так і у загальнодоступній хмарі. Він надає такі зручні інструменти, як єдиний інтерфейс користувача, що дозволяє керувати мультихмарним середовищем в цілому, неперервний контроль за змінами, відкрите середовище, що унеможливлює прив'язку до певного провайдера хмарних послуг. Серед аналогів його вирізняє простота використання, дана платформа чудово підходить для невеликих команд із офісами по всьому світу.

Іншим відомим рішенням є Scalr[8], який розроблений із врахуванням масштабів підприємств. Воно допомагає стандартизувати використання, контролювати витрати та дозволяє користувачам обирати найкращі хмарні сервіси для власних потреб без прив'язки до постачальника. Scalr дозволяє створювати власні політики, що можна налаштовувати, які роблять можливим створення сумісного мультихмарного середовища шляхом автоматизації цих політик. Гарно підходить для великих організацій, яким необхідно керувати великою кількістю відділів та офісів. Відомими клієнтами є Xerox, Samsung, NASA JPL.

Платформа керування хмарами Bunnyshell дозволяє автоматично налаштовувати, масштабувати та розгортати застосунки у декількох хмарних провайдерів. Є можливість перенесення застосунку із однієї хмари у іншу за запитом, а також моделювання міграцій.

5. Платформи оркестрування хмарних ресурсів

Більшість сучасних провайдерів мають власні платформи окрестрування хмарних ресурсів, які застосовуються для керування життєвим циклом ресурсів, починаючи від фази вибору і закінчуючи фазою моніторингу. Але ці продукти найчастіше є пропрієтарними та через комерційні причини не можуть бути портативними. Хоча й існують сучасні рішення для керування конфігурацією (Puppet, Chef, Amazon OpsWorks[10]), які забезпечують підтримку обробки конфігурації ресурсів через хмарні сервіси, проте часто виявляється, що користувачам необхідно розуміти низькорівневі програмні інтерфейси хмарних сервісів для того, щоб створювати та підтримувати складні конфігурації ресурсів.

Поява мультихмарних обчислень ще більше ускладнила складні самі по собі проблеми окрестрування, проте великим платформам окрестрування довелося впроваджувати мультихмарну підтримку, аби бути конкурентноспроможним гравцем на цьому ринку. Нові вимоги передбачають створення нових потужніших засобів окрестрування ресурсів, що охоплюють декілька хмарних адміністративних доменів, здатні впоратися з неоднорідністю базових хмарних ресурсів та сервісів. Оскільки мультихмара передбачає відсутність попередніх домовленостей між постачальниками хмарних послуг, то доводиться користуватися послугами хмарного брокера, який є посередником між постачальниками та споживачами хмарних обчислень.

У [9] запропоновано еталонну архітектуру для фреймворків окрестрування хмарних ресурсів (ФОХР). Складається вона із ряду рівнів. Рівень доступу відповідає за регулювання взаємодії із платформою. Користувачі можуть отримувати доступ до служб із нижніх рівнів за допомогою інтерфейсів командного рядка, веб-API та панелей моніторингу. Рівень керування застосунками контролю за роботою застосунків протягом усього життєвого циклу. Рівень керування ресурсами включає у себе служби (наприклад, служби виявлення, моніторингу), які керують ресурсами протягом усього їх життєвого циклу. Рівень надання ресурсів відповідає за послуги, що виконують основні операції із хмарними ресурсами (створення, запуск, масштабування, зупинка, видалення). ФОХР, що підтримують мультихмарність, мають пропонувати рівень абстракції хмари, який приховуватиме відмінності та уникатиме необхідності індивідуального налаштування постачальника.

На базі розглянутої архітектури можна виділити два види ФОХР: виробничі/комерційні та експериментальні/академічні. Перші із них використовуються у виробничому середовищі постачальниками приватної та публічної хмари, а другі функціонують у дослідницькому середовищі.

Представниками виробничих/комерційних ФОХР, які підтримують мультихмарність, є Cloudify, Brooklyn, Stratos, Alien4Cloud, Terraform. [11, 12, 13, 14]

Cloudify дозволяє розгортати мультихмарні рішення за допомогою вбудованих плагінів. Він також підтримує BYONта використовує TOSCA для сумісності та переносимості. Недоліком є те, що складні сценарії, такі як вертикальне масштабування, не підтримуються «з коробки».

Brooklyn дозволяє розгортати мультихмарні системи в багатьох приватних та публічних хмарах. Використовується Apache jclouds[15] у якості рівня абстракції хмари для забезпечення сумісності. Переносимість досягається за рахунок механізмів повторного та сумісного використання моделі.

У платформі Apache Stratos застосунки зазвичай складаються із наборів катриджів, які представляють опис абстрактних віртуальних машин, на яких розміщуються як бізнес-сервіси, так і сервіси інфраструктури. Для підтримки кількох постачальників також використовується Apache jclouds в якості рівня абстракції хмари. Хмарні ресурси обираються вручну при налаштуванні катриджів.

У межах Alien4Cloud застосунки можна обирати лише за допомогою ручної прив'язки. Як ручні, так і автоматичні робочі процеси можуть використовувати інструменти на основі сценаріїв чи DevOps (Chef[16], Puppet) для керування життєвим циклом застосунку.

Terraform може керувати кількома хмарними провайдерами і навіть залежностями між хмарами за допомогою спеціальних плагінів, які називаються провайдерами. Доступні постачальники для контейнерів Docker і оркестровки контейнерів, а також для зовнішніх хмарних сервісів (наприклад, Amazon RDS). Хмарні ресурси вибираються вручну під час конфігурації, в той час як дії життєвого циклу можуть бути налаштовані за допомогою ініціаторів, виконують сценарії або керуючих конфігурацією (Chef, Puppet, Salt[17]).

Серед експериментальних/академічних ФОХР, які підтримують мультихмарність, можна виділити Cloudiator, Roboconf, INDIGO-DataCloud, MiCADO, MODAClouds, SeaClouds.[18,19,20,21,22,23]

У Cloudiator присутній брокер ресурсів, який відповідає за правильний вибір пропозиції хмари в залежності від бажаних вимог та обмежень стосовно конфігурації віртуальної машини. INDIGO-DataCloud автоматично обирає та оптимізує хмарні ресурси в залежності від SLA і даних моніторингу. Рішення для керування конфігурацією, засноване на ролях Ansible, застосовується як для розгортання програми, так і для створення попередньо налаштованих образів Docker. В межах платформи MiCADO хмарні ресурси вибираються вручну при налаштуванні віртуальних машин. Життєвий цикл додатку обробляється самою MiCADO, яка використовує Occopus і Kubernetes для керування віртуальними машинами і контейнерами відповідно. MODAClouds використовує підхід MDE для підтримки взаємодії між

хмарними провайдерами. Зокрема, MODACloudML являє собою набір розширень UML, що дозволяють розробникам моделювати мультихмарні застосунки за допомогою трьох рівнів абстракції: хмарні незалежні обчислювальні моделі, незалежні від хмарного провайдера моделі і моделі, що залежать від хмарного провайдера. Ці моделі полегшують портативність, оскільки в більшості випадків вони їх можна використовувати багаторазово. Хмарні ресурси можна автоматично обирати й оптимізувати за допомогою Venues4Clouds і SpaceDev4Clouds, а керувати ними можна або за допомогою сценаріїв оболонки, або за допомогою Puppet[9].

Висновки

В межах даної статті розглянуто передумови виникнення мультихмари, переваги, які отримуються при використання хмарних ресурсів декількох провайдерів, такі як розширення функціональної наповненості програмного продукту, коли єдина хмара не має відповідного функціоналу чи він є значно обмеженим, дорожчим чи повільнішим, залежність від провайдера, а також проблеми, що пов'язані із її використанням. Визначено, що головною проблемою постає складність узгодження кількох хмарних сервісів різних постачальників.

Досліджено сучасні рішення, які можуть використовуватися для керування мультихмарними продуктами. Потужними інструментами є платформи мультихмарного керування, наприклад, від IBM, рішення Scalr, Bunnyshell. Іншим видом програмних інструментів є фреймворки окрестрування хмарних ресурсів: виробничі/комерційні (Cloudify, Brooklyn, Stratos, Alien4Cloud, Terraform) та експериментальні/академічні (Cloudiator, Roboconf, INDIGO-DataCloud, MiCADO, MODAClouds, SeaClouds), які мають засоби оркестрування ресурсів, що здатні працювати із рядом хмарних сервісів одночасно для забезпечення коректної роботи застосунку. Також визначено критерії, за якими найпопулярніші постачальники хмарних послуг краще підходять для вирішення поставленої задачі (Azure для обробки великих даних, Oracle для роботи із об'єктно-реляційними базами даних, AWS для зручного швидкого розгортання серверних частин веб-сайтів).

Отже, мультихмарні обчислення мають великі перспективи розвитку, проте використовувати їх потрібно лише у випадку необхідності, аби не зіткнутися із зайвими витратами та втратою часу на необхідні налаштування.

Список літератури:

[1] Smith D., Leong L. Innovation Insight for Multicloud Computing : website.URL: https://www.gartner.com/en/documents/3994448-innovation-insight-for-multicloud-computing (accessed:15.05.2021).

[2] Turner M. J. Worldwide Cloud System and Service Management Software Forecast Update, 2020–2024: Enterprise Investments Rebound : website.URL: https://www.idc.com/getdoc.jsp?containerId=US47079320 (accessed:15.05.2021).

[3] Rethink data: A Seagate technology report with research and analysis by IDC : website.URL: https://www.seagate.com/files/www-content/our-story/rethink-data/files/Rethink_Data_Report _2020.pdf (accessed:15.05.2021).

[4] Aditi Rajan Khot. A Comparative Analysis of Public CloudPlatforms and Introduction of Multi-Cloud. International Journal of Innovative Science and Research Technology. 2020. Volume 5, Issue 9. P. 448–454.

[5] Where Are The Data Centers: AWS, Alibaba Cloud, Azure, GCP. Interconnected: website.URL:https://interconnected.blog/data-center-coverage-aws-alibaba-azure-gcp/ (accessed:15.05.2021).

[6] Lee Y. S. Analysis on Trends of Machine Learning-as-a-Service //International Journal of Advanced Culture Technology. – 2018. – Т. 6. – №. 4. – С. 303-308.

[7] Raj P., Raman A. Automated multi-cloud operations and container orchestration //Software-Defined Cloud Centers. – Springer, Cham, 2018. – С. 185-218.

[8] Mohamed A. M., Abdelsalam H. M. A multicriteria optimization model for cloud service provider selection in multicloud environments //Software: Practice and Experience. – 2020. – Т. 50. – №. 6. – С. 925-947.

[9] Tomarchio O., Calcaterra D., Di Modica G. Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks //Journal of Cloud Computing. – 2020. – Т. 9. – №. 1. – С. 1-24.

[10] Lu M. et al. An orchestration framework for a global multi-cloud //Proceedings of the 2018 Artificial Intelligence and Cloud Computing Conference. – 2018. – С. 58-62.

[11] de Carvalho L. R., de Araujo A. P. F. Performance Comparison of Terraform and Cloudify as Multicloud Orchestrators //2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID). – IEEE, 2020. – С. 380-389.

[12] Carrasco J. et al. Bidimensional cross-cloud management with TOSCA and Brooklyn //2016 IEEE 9th International Conference on Cloud Computing (CLOUD). – IEEE, 2016. – С. 951-955.

[13] Son S. et al. Cloud SLA relationships in multi-cloud environment: models and practices //Proceedings of the 8th International Conference on Computer Modeling and Simulation. – 2017. – С. 1-6.

[14] Calcaterra D. et al. A Comparison of Multi-cloud Provisioning Platforms //CLOSER. – 2019. – С. 507-514.

[15] Toews E., Advocate D. Introduction to Apache jclouds //Apr. – 2014. – Т. 7. – С. 23.

[16] Ebert C. et al. DevOps //Ieee Software. – 2016. – Т. 33. – №. 3. – С. 94-100.

[17] KONG Y. et al. Multi-cloud storage system based on Android platform //Journal of Computer Applications. – 2017. – С. S1.

[18] Baur D. A provider agnostic approach to multi-cloud orchestration. – 2018.

[19] Pham L. M. et al. Roboconf: A hybrid cloud orchestrator to deploy complex applications //2015 IEEE 8th International Conference on Cloud Computing. – IEEE, 2015. – С. 365-372.

[20] Salomoni D. et al. INDIGO-DataCloud: A platform to facilitate seamless access to e-infrastructures //Journal of Grid Computing. – 2018. – Т. 16. – №. 3. – С. 381-408.

[21] Kiss T. et al. Micado—microservice-based cloud application-level dynamic orchestrator //Future Generation Computer Systems. – 2019. – Т. 94. – С. 937-946.

[22] Di Nitto E. et al. Model-driven development and operation of multi-cloud applications: the MODAClouds approach. – Springer Nature, 2017.

[23] Brogi A. et al. SeaClouds: an open reference architecture for multi-cloud governance //European Conference on Software Architecture. – Springer, Cham, 2016. – С. 334-338.

Надійшла до редколегії 23.04.2021

УДК 004.852 DOi 10.30837/ bi.2021.1(96).11

О.Г. Руденко[1], О.О. Безсонов[2], Н.М. Сердюк[3], О.Г. Лебедєв[4], В.О. Лебедєв[5]

[1]Доктор технічних наук, завідувач кафедри комп'ютерних інтелектуальних технологій та систем, Харківський національний університет радіоелектроніки, Україна, oleh.rudenko@nure.ua, ORCID iD: 0000-0003-0859-2015

[2]Доктор технічних наук, професор кафедри комп'ютерних інтелектуальних технологій та систем, Харківський національний університет радіоелектроніки, Україна, oleksandr.bezsonov@nure.ua, ORCID iD: 0000-0001-6104-4275

[3] Кандидат технічних наук, доцент кафедри комп'ютерних інтелектуальних технологій та систем, Харківський національний університет радіоелектроніки, Україна, nataliya.serdyuk@nure.ua, ORCID iD: 0000-0002-0107-4365

[4] Кандидат технічних наук, доцент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Україна, oleh.lebediev@nure.ua, ORCID iD: 0000-0001-5998-0136

[5] Аспірант кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Україна, valentyn.lebediev@nure.ua ORCID id: 0000-0002-0095-7481

БАГАТОКРОКОВИЙ МЕТОД НАВЧАННЯ АДАЛІНИ ЗА НАЯВНІСТЮ СТАЦІОНАРНИХ КОРЕЛЬОВАНИХ ЗАВАД

Розглянуто задачу побудови рекурентної форми багатокрокового алгоритму навчання АДАЛІНИ при наявності стаціонарних корельованих завад. Процес навчання АДАЛІНИ можна значно прискорити, якщо застосувати багатокроковий алгоритм, який використовує обмежене число вимірів, тобто має обмежену пам'ять, та побудований на базі алгоритму поточного регресійного аналізу. Розглянуто задачу побудови рекурентної форми алгоритму поточного регресійного аналізу, що дозволяє здійснювати оцінювання невідомих параметрів при наявності стаціонарних корельованих завад. Отримано основні співвідношення, які описують процеси накопичення нової і скидання застарілої інформації. Показано, що алгоритми, які розглядаються, використовують перетворення випадкового вектора з корельованими складовими в випадковий вектор з некорельованими складовими.

КОРЕЛЯЦІЯ, НАВЧАННЯ, РЕКУРЕНТНИЙ АЛГОРИТМ, КОРЕЛЯЦІЙНА МАТРИЦЯ, ЦЕНТРУВАННЯ, ЗАВАДА, ВКЛЮЧЕННЯ ІНФОРМАЦІЇ, СКИДАННЯ ІНФОРМАЦІЇ

Руденко О.Г., Бессонов А.А., Сердюк Н.Н., Лебедев О.Г., Лебедев В.О. Многошаговый метод обучения АДАЛИНЫ при наличии стационарных коррелированных помех. Рассмотрена задача построения рекуррентной формы многошагового алгоритма обучения АДАЛИН при наличии стационарных коррелированных помех. Процесс обучения АДАЛИН можно значительно ускорить, если применить многошаговый алгоритм, который использует ограниченное число измерений, то есть имеет ограниченную память и построен на базе алгоритма текущего регрессионного анализа. Рассмотрена задача построения рекуррентной формы алгоритма текущего регрессионного анализа, что позволяет осуществлять оценку неизвестных параметров при наличии стационарных коррелированных помех. Получены основные соотношения, описывающие процессы накопления новой и сброса устаревшей информации. Показано, что рассматриваемые алгоритмы используют преобразования случайного вектора с коррелированными составляющими в случайный вектор с некоррелированными составляющими.

КОРРЕЛЯЦИЯ, ОБУЧЕНИЕ, РЕКУРРЕНТНЫЙ АЛГОРИТМ, КОРРЕЛЯЦИОННАЯ МАТРИЦА, ЦЕНТРОВКА, ПОМЕХА, ВКЛЮЧЕНИЕ ИНФОРМАЦИИ, СБРОС ИНФОРМАЦИИ.

Rudenko O.G., Bezsonov O.O., Serdiuk N.M., Lebediev O.G., Lebediev V.O. Multi-step method of ADALINE learning in the presence of stationary correlated noises. The problem of constructing a recurrent form of a multistep learning algorithm ADALIN in the presence of stationary correlated noise is considered. The ADALIN learning process can be significantly accelerated if we apply a multi-step algorithm that uses a limited number of measurements, that is, it has a limited memory and is based on the current regression analysis algorithm. The problem of constructing a recurrent form of the current regression analysis algorithm is considered, which makes it possible to estimate unknown parameters in the presence of stationary correlated noise. The main relationships are obtained that describe the processes of accumulation of new and dumping of obsolete information. It is shown that the algorithms under consideration use transformations of a random vector with correlated components into a random vector with uncorrelated components.

CORRELATION, LEARNING, RECURRENT ALGORITHM, CORRELATION MATRIX, CENTERING, INTERFERENCE, INCLUDING INFORMATION, RESETTING INFORMATION.

Вступ

АДАЛІНА (адаптивний лінійний елемент) став першою лінійною нейронною мережею, запропонованою Уїдроу Б. і Хоффом М.Є., яка представляла альтернативу персептрону [1]. Згодом даний елемент і алгоритм його навчання знайшли досить широке застосування в задачах ідентифікації, управління, фільтрації і т.п. Алгоритм навчання Уїдроу – Хоффа є алгоритмом Качмажа рішення систем лінійних алгебраїчних рівнянь. Властивості даного алгоритму при вирішенні задачі ідентифікації досить повно описані в [2]. В роботі [3] регуляризований алгоритм Качмажа (Уїдроу-Хоффа) застосовувався для навчання АДАЛІНИ в завданні оцінювання нестаціонарних параметрів і отримано відповідні оцінки швидкості збіжності та точності.

Слід зазначити, що процес навчання АДАЛІНИ можна значно прискорити, якщо замість одно крокового алгоритму Уїдроу – Хоффа (Качмажа) застосувати багатокроковий алгоритм, який використовує обмежене число вимірів, тобто має обмежену пам'ять, та побудований на базі алгоритму поточного регресійного аналізу [4-6].

Зазвичай при побудові нейромережевих моделей і їх дослідженні передбачається, що присутні в вимірах завади не корельовані. Наявність же корельованих завад призводить до ускладнення процедури оцінювання. Тому часто корельованістю завад нехтують, отримуючи при цьому свідомо неоптимальні результати.

Рекурентне оцінювання при корельованих завадах розглядалося в [7-9]. В [7] запропонована рекурентна форма МНК, в [8] вивчалася процедура типу стохастичної апроксимації, коефіцієнти якої на кожному кроці обираються оптимальними в сенсі мінімізації суми апостеріорної дисперсії оцінок. В роботі [9] показано, що обидва ці способи мають багато спільного і було проведено їх експериментальне порівняння, яке показало, що в разі дуже сильної кореляції РМНК має більшу обчислювальну стійкість.

Метою даної роботи є отримання рекурентної форми багатокрокового алгоритму навчання за умов стаціонарних корельованих завад.

1. Побудова рекурентної форми алгоритму поточного регресійного аналізу

Розглянемо задачу ідентифікації об'єкта, що описується рівнянням

$$Y_n = X_n c^* + \Xi_n, \tag{1}$$

де $Y_n = (y_1, y_2, ... y_n)^T$ – вектор вихідних сигналів;

$X_n^T = (x_1, x_2, ..., x_n)^T = \left(X_{n-1}^T x_n\right)$ – матриця вхідних сигналів;

$c^* = (c_1^*, c_2^*, ..c_N^*)^T$ – вектор параметрів, що оцінюються;

$\Xi_n = (\xi_1, \xi_2, ..., \xi_n)^T$ – вектор завад.

Коваріаційна матриця D_n порядку n завади ξ_{n+1} має вид

$$D_n = M\left\{\Xi_n \Xi_n^T\right\} = \begin{bmatrix} d_{1,1} & d_{1,2} ... & d_{1,n-1} & d_{1,n} \\ d_{2,1} & d_{2,2} ... & d_{2,n-1} & d_{2,n} \\ ... & ... & ... & ... \\ d_{n,1} & d_{n,2...} & d_{n,n-1} & d_{n,n} \end{bmatrix} = \begin{bmatrix} D_{n-1} & d_{n-1} \\ d_{n-1}^T & d_{nn} \end{bmatrix}, \tag{2}$$

де $d_{ij} = M\{\xi_i \xi_j\}; ....$

$d_{n-1}^T = (d_{n,1}, d_{n,2}, ..., d_{n,n-1}) = M\left\{\xi_n \Xi_{n-1}^T\right\}.$

Як відомо [10], застосування оцінки

$$c_n = \left(X_n^T X_n\right)^{-1} X_n^T Y_n \tag{3}$$

до моделі з корельованими завадами дає оцінки, дисперсії яких будуть занижені.

Гаусівська-марківська оцінка (МНК), що отримується мінімізацією квадратичного функціоналу, має вигляд

$$c_n = \left(X_n^T D_n^{-1} X_n\right)^{-1} X_n^T D_n^{-1} Y_n \tag{4}$$

Алгоритм поточного регресійного аналізу, що має вигляд

$$c_{n|L} = (X_{n|L}^T X_{n|L})^{-1} X_{n|L}^T Y_{n|L}, \tag{5}$$

$$Y_{n|L} = \begin{pmatrix} Y_{n-1|L-1} \\ --- \\ y_n \end{pmatrix} = \begin{pmatrix} y_{n-L+1} \\ --- \\ Y_{n|L-1} \end{pmatrix}, \tag{6}$$

де $L \times 1$ – вектор;

$$X_{n|L} = \begin{pmatrix} X_{n-1|L-1} \\ --- \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{n-L+1}^T \\ --- \\ X_{n|L-1} \end{pmatrix}, \tag{7}$$

$(L \times 1) \times N$ – матриця;

був запропонований в роботі [4], а в [5] розглянута модифікація цього алгоритму, що використовує механізм забування минулого інформації (згладжування). Тут $L = const (L \geq N)$ - пам'ять алгоритму.

Особливістю алгоритмів з $L = const$ є те, що використовувані при побудові оцінок матриці і вектори спостережень на кожному кроці оцінювання формуються наступним чином: в них включається інформація про новоприбулі виміри і виключається інформація про найбільш старі. Залежно від того, як формуються ці матриці і вектори (додається спочатку нова інформація, а потім виключається застаріла, або ж спочатку виключається застаріла, а потім додається нова) можливі дві форми оцінки.

Зупинимося на цьому докладніше.

За аналогією з гаусівсько-марківською оцінкою (4) може бути запропонована оцінка

$$c_{n|L} = (X_{n|L}^T D_{n|L}^{-1} X_{n|L})^{-1} X_{n|L}^T D_{n|L}^{-1} Y_{n|L}, \tag{8}$$

де

$$D_{n|L}=\begin{bmatrix} d_{n-L+1,n-L+1} & d_{n-L,n-L+2}\cdots & d_{n-L+1,n-1} & d_{n-L+1,n} \\ d_{n-L+2,n-L+1} & d_{n-L+2,n-L+2} & d_{n-L+2,n-1} & d_{n-L+2,n} \\ \cdots & \cdots & \cdots & \cdots \\ d_{n,n-L+1} & d_{n,n-L+2} & d_{n,n-1} & d_{n,n} \end{bmatrix}=$$

$$=\begin{bmatrix} D_{n-1,L-1+1} & \vdots & d_{n-1} \\ - & - & - \\ d_{n-1}^T & \vdots & d_{n,n} \end{bmatrix}, \qquad (9)$$

$$d_{n-1}^T=\left(d_{n,n-L+1},d_{n,n-L+2},...,d_{n,n-1}\right)=M\left\{\xi_n\Xi_{n-1|L-1}^T\right\}$$

У зв'язку з тим, що матриця $D_{n|L}$ має блокове представлення, то справедливо [11]

$$D_{n|L}^{-1}=$$

$$=\begin{bmatrix} D_{n-1,L-1}^{-1}+\dfrac{D_{n-1,L-1}^{-1}d_{n-1}d_{n-1}^TD_{n-1,L}^{-1}}{\alpha_n} & \vdots & -\dfrac{D_{n-1,L-1}^{-1}d_{n-1}}{\alpha_n} \\ \cdots & \cdots & \cdots \\ -\dfrac{d_{n-1}^TD_{n-1,L-1}^{-1}}{\alpha_n} & \vdots & \dfrac{1}{\alpha_n} \end{bmatrix}, \qquad (10)$$

де $\alpha_n=d_{nn}-d_{n-1}^TD_{n-1,L-1}^{-1}d_{n-1}$.

Аналогічні співвідношення і використовуються в [7] для отримання рекурентної форми МНК.

Розглянемо випадок стаціонарних завад ξ.

Нормована коваріаційна матриця для стаціонарної завади з дисперсією σ_ξ^2 має вигляд

$$R_{n|L}=D_{n|L}/\sigma_\xi^2=\begin{bmatrix} 1 & \rho_1\cdots & \rho_{n-L} & \rho_{n-L+1} \\ \rho_1 & 1\cdots & \rho_{n-3} & \rho_{n-L} \\ \cdots & \cdots & \cdots & \cdots \\ \rho_{n-L+1} & \rho_{n-L}\cdots & \rho_1 & 1 \end{bmatrix}, \qquad (11)$$

де ρ_i – значення нормованої кореляційної функції процесу ξ.

С урахуванням цього оцінка (8) набирає вигляду

$$c_{n|L}=(X_{n|L}^TR_{n|L}^{-1}X_{n|L})^{-1}X_{n|L}^TR_{n|L}^{-1}Y_{n|L}, \qquad (12)$$

Введемо $(n-m+1)$ – вимірний вектор

$$r_{m,n}^T=\left(\rho_n,\rho_{n-1},...\rho_m\right), \quad (m\le n).$$

Тоді матриця $R_{n|L}$ може бути представленою в блочному виді

$$R_{n|L}=D_{n|L}/\sigma_\xi^2=\begin{bmatrix} R_{n-1|L-1} & r_{n-1,1} \\ r_{n-1,1}^T & 1 \end{bmatrix}, \qquad (13)$$

де $r_{n-1,1}^T=\left(\rho_{n-1},\rho_{n-2},...\rho_1\right)$, а зворотна матриця $R_{n|L}^{-1}$ буде обчислюватися так:

$$R_{n|L}^{-1}=$$

$$=\begin{bmatrix} R_{n-1|L-1}^{-1}+\dfrac{R_{n-1|L-1}^{-1}r_{n-1,1}r_{n-1,1}^TR_{n-1|L-1}^{-1}}{1-r_{n-1,1}^TR_{n-1|L-1}^{-1}r_{n-1,1}} & -\dfrac{R_{n-1|L-1}^{-1}r_{n-1,1}}{1-r_{n-1,1}^TR_{n-1|L-1}^{-1}r_{n-1,1}} \\ -\dfrac{r_{n-1,1}^TR_{n-1}^{-1}}{1-r_{n-1,1}^TR_{n-1|L-1}^{-1}r_{n-1,1}} & \dfrac{1}{1-r_{n-1,1}^TR_{n-1|L-1}^{-1}r_{n-1,1}} \end{bmatrix}. \qquad (14)$$

2. Процедура додавання нової інформації

Припустимо, що на $(n-1)$-му такті отримана оцінка

$$c_{n-1|L}=\left(X_{n-1|L}^TR_{n-1|L}^{-1}X_{n-1|L}\right)^{-1}X_{n-1|L}^TR_{n-1|L}^{-1}Y_{n-1|L} \qquad (15)$$

Прихід нової інформації (додавання нового виміру) призводить до обчислення оцінки, яка за аналогією з (15) може бути записана наступним чином:

$$c_{n|L+1}=(X_{n|L+1}^TR_{n|L+1}^{-1}X_{n|L+1})^{-1}X_{n|L+1}^TR_{n|L+1}^{-1}Y_{n|L+1}, \qquad (16)$$

де

$$Y_{n|L+1}=\begin{pmatrix} Y_{n-1|L} \\ --- \\ y_n \end{pmatrix}=\begin{pmatrix} y_{n-L+1} \\ --- \\ Y_{n|L} \end{pmatrix} \text{ – вектор } (L+1)\times 1. \qquad (17)$$

Введемо позначення

$$K_{n|L+1}^{-1}=\left(X_{n|L+1}^TR_{n|L+1}^{-1}X_{n|L+1}\right);$$
$$K_{n-1|L}^{-1}=\left(X_{n-1|L}^TR_{n-1|L}^{-1}X_{n-1|L}\right); \qquad (19)$$
$$K_{n|L}^{-1}=\left(X_{n|L}^TR_{n|L}^{-1}X_{n|L}\right).$$

$$\alpha_{j,n}=\rho_j-r_{j+1,n+j-1}R_{n-1|L-1}^{-1}r_{1,n-1} \quad (j=0,1,...) \qquad (20)$$

та обчислимо $K_{n|L+1}^{-1}$

$$K_{n|L+1}^{-1}=X_{n-1|L}^TR_{n-1|L}^{-1}X_{n-1|L}+$$
$$+\frac{X_{n-1|L}^TR_{n-1|L}^{-1}r_{n-1,1}r_{n-1,1}^TR_{n-1|L}^{-1}X_{n-1|L}}{\alpha_{n,0}}-$$
$$-\frac{x_nr_{n-1,1}^TR_{n-1|L}^{-1}X_{n-1|L}}{\alpha_{n,0}}-\frac{X_{n-1|L}^TR_{n-1|L}^{-1}r_{n-1,1}x_n^T}{\alpha_{n,0}}+ \qquad (21)$$
$$+\frac{x_nx_n^T}{\alpha_{n,0}}=K_{n-1|L}^{-1}+x_n^*x_n^{*T},$$

де $x_n^*=\dfrac{x_n-X_{n-1|L}^TR_{n-1|L}^{-1}r_{n-1,1}}{\sqrt{\alpha_{n,0}}}$.

Аналогічно обчислимо

$$X_{n|L+1}^TR_{n|L+1}^{-1}Y_{n|L+1}=X_{n-1|L}^TR_{n-1|L}^{-1}Y_{n-1|L}+x_n^*y_n^*, \qquad (22)$$

де $y_n^*=\dfrac{y_n-Y_{n-1|L}R_{n-1|L}^{-1}r_{n-1,1}}{\sqrt{\alpha_{n,0}}}$.

Додаючи до обох частин (15) $x_n^*x_n^{*T}c_{n-1|L}$

$$K_{n-1|L}^{-1}c_{n-1|L}+x_n^*x_n^{*T}c_{n-1|L}=$$
$$=X_{n-1|L}^TR_{n-1|L}^{-1}Y_{n-1|L}+x_n^*x_n^{*T}c_{n-1|L}$$

і віднімаючи (15) з (16) (з урахуванням властивостей $K_{n-1|L}^{-1}$ и $X_{n|L+1}^TR_{n|L+1}^{-1}Y_{n|L+1}$), отримуємо

$$K_{n|L+1}^{-1}\left(c_{n|L+1}-c_{n-1|L}\right)=x_n^*\left(y_n^*-c_{n-1|L}^Tx_n^*\right)$$

або

$$c_{n|L+1}=c_{n-1|L}+K_{n|L+1}x_n^*\left(y_n^*-c_{n-1|L}^Tx_n^*\right), \qquad (23)$$

де

$$K_{n|L+1}=K_{n-1|L}-\frac{K_{n-1|L}x_n^*x_n^{*T}K_{n-1|L}}{1+x_n^{*T}K_{n-1|L}x_n^*}. \qquad (24)$$

3. Процедура відкиданні застарілої інформації

При відкиданні застарілої інформації, отриманої на $n-L+1$-у кроці, ми приходимо від оцінки $c_{n|L+1}$ до оцінки $c_{n|L}$. Для отримання відповідних правил корекції оцінки поступимо таким чином.

Скористаємося блоковим поданням ковариаційної матриці $R_{n|L+1}$

$$R_{n|L+1}=\begin{bmatrix} 1 & \rho_1 & \rho_2 & \rho_{n-L+1} \\ \rho_1 & 1 & \rho_1 & \rho_{n-L+1,n} \\ \dots & \dots & \dots & \dots \\ \rho_{n-L+1} & \rho_{n-L} & \rho_{n-L-1} & 1 \end{bmatrix}=\begin{bmatrix} 1 & r_{n-1,1}^T \\ r_{n-1,1} & R_{n|L} \end{bmatrix}, \quad (25)$$

де $r_{n-1,1}^T=(\rho_{n-L+1},\rho_{n-L},\dots,\rho_{n-1})$, та представленням зворотної матриці $R_{n|L+1}^{-1}$ у вигляді

$$R_{n|L+1}^{-1}=\begin{bmatrix} \dfrac{1}{\alpha_{n-L+1,0}} & -\dfrac{r_{n-L+1}^T R_{n,L}^{-1}}{\alpha_{n-L+1,0}} \\ -\dfrac{R_{n|L}^{-1} r_{n-L+1}}{\alpha_{n-L+1,0}} & R_{n,L}^{-1}+\dfrac{R_{n,L}^{-1} r_{n-L+1} r_{n-L+1}^T R_{n,L}^{-1}}{\alpha_{n-L+1,0}} \end{bmatrix}, \quad (26)$$

де $\alpha_{n-L+1,0}=1-r_{n-L+1,1}^T R_{n,L}^{-1} r_{n-L+1,1}$.

В цьому випадку

$$\begin{aligned} K_{n|L+1}^{-1}&=\left(X_{n|L+1}^T R_{n|L+1}^{-1} X_{n|L+1}\right)= \\ &=\frac{x_{n-L+1}x_{n-L+1}^T}{\alpha_{n-L}}-+\frac{X_{n|L}^T R_{n|L}^{-1} d_{n-L+1} d_{n-L+1}^T R_{n|L}^{-1} X_{n|L}}{\alpha_{n-L}}- \\ &-\frac{x_{n-L+1} d_{n-L+1}^T R_{n|L}^{-1} X_{n|L}}{\alpha_{n-L+1}}-\frac{X_{n|L}^T R_{n|L}^{-1} d_{n-L+1} x_{n-L+1}^T}{\alpha_{n-L+1}}+ \\ &+X_{n|L}^T R_{n|L}^{-1} X_{n|L}=K_{n|L}^{-1}+x_{n-L+1}^* x_{n-L+1}^{*T}, \end{aligned} \quad (27)$$

де $x_{n-L+1}^*=\dfrac{x_{n-L+1}-X_{n|L}^T D_{n|L}^{-1} r_{n-L+1,1}}{\sqrt{\alpha_{n-L+1,0}}}$.

Аналогічно маємо

$$X_{n|L+1}^T R_{n|L+1}^{-1} Y_{n|L+1}=X_{n|L}^T R_{n|L}^{-1} Y_{n|L}+x_{n-L+1}^* y_{n-L+1}^*, \quad (28)$$

де $y_{n-L+1}^*=\dfrac{y_{n-L+1}-Y_{n|L} D_{n|L}^{-1} r_{n-L+1,1}}{\sqrt{\alpha_{n-L+1,0}}}$.

Віднімемо з обох частин (16) $x_{n-L+1}^* x_{n-L+1}^{*T} c_{n|L+1}$. Тоді

$$\begin{aligned} &\left(\left(X_{n|L+1}^T R_{n|L+1}^{-1} Y_{n|L+1}\right)^{-1}-x_{n-L+1}^* x_{n-L+1}^{*T}\right)c_{n|L+1}= \\ &=X_{n|L+1}^T R_{n|L+1}^{-1} Y_{n|L+1}-x_{n-L+1}^* x_{n-L+1}^{*T} c_{n|L+1}. \end{aligned} \quad (29)$$

Враховуючи, що

$$\left(X_{n|L}^T R_{n|L}^{-1} X_{n|L}\right)c_{n|L}=X_{n|L}^T R_{n|L}^{-1} Y_{n|L}, \quad (30)$$

віднімемо з (30) співвідношення (16) (с урахуванням виразів для $K_{n|L}^{-1}$ та $X_{n|L}^T R_{n|L}^{-1} Y_{n|L}$)

$$K_{n|L}^{-1}\left(c_{n|L}-c_{n|L+1}\right)=x_{n-L+1}^* x_{n-L+1}^{*T} c_{n|L+1}-x_{n-L+1}^* y_{n-L+1}^*,$$

звідки

$$c_{n|L}=c_{n|L+1}-K_{n|L} x_{n-L+1}^*\left(y_{n-L+1}^*-c_{n|L+1}^T x_{n-L+1}^*\right), \quad (31)$$

але $K_{n|L}^{-1}=K_{n|L+1}^{-1}-x_{n-L+1}^* x_{n-L+1}^{*T}$.

Тому

$$K_{n|L}=K_{n|L+1}+\frac{K_{n|L+1} x_{n-L+1}^* x_{n-L+1}^{*T} K_{n|L+1}}{1-x_{n-L+1}^{*T} K_{n|L+1} x_{n-L+1}^*}. \quad (32)$$

Таким чином, алгоритм буде мати вигляд (перші два співвідношення описують включення нової інформації, а наступні два - відкидання застарілої)

$$c_{n|L+1}=c_{n-1|L}+K_{n|L+1} x_n^*\left(y_n^*-c_{n-1|L}^T x_n^*\right), \quad (33)$$

$$K_{n|L+1}=K_{n-1|L}-\frac{K_{n-1|L} x_n^* x_n^{*T} K_{n-1|L}}{1+x_n^{*T} K_{n-1|L} x_n^*}, \quad (34)$$

$$c_{n|L}=c_{n|L+1}-K_{n|L} x_{n-L+1}^*\left(y_{n-L+1}^*-c_{n|L+1}^T x_{n-L+1}^*\right), \quad (35)$$

$$K_{n|L+1}=K_{n|L}+\frac{K_{n|L+1} x_{n-L+1}^* x_{n-L+1}^{*T} K_{n|L+1}}{1-x_{n-L+1}^{*T} K_{n|L+1} x_{n-L+1}^*}. \quad (36)$$

Якщо ж спочатку відкидається застаріла інформація, а потім включається нова, то алгоритм набуває вигляду

$$\begin{aligned} c_{n-1|L-1}&=c_{n-1|L}- \\ &-K_{n-1|L-1} x_{n-L+1}^*\left(y_{n-L+1}^*-c_{n-1|L}^T x_{n-L+1}^*\right); \end{aligned} \quad (37)$$

$$K_{n-1|L-1}=K_{n-1|L}+\frac{K_{n-1|L} x_{n-L+1}^* x_{n-L+1}^{*T} K_{n-1|L}}{1-x_{n-L+1}^{*T} K_{n-1|L} x_{n-L+1}^*}; \quad (38)$$

$$c_{n|L}=c_{n-1|L-1}+K_{n|L} x_n^*\left(y_n^*-c_{n-1|L-1}^T x_n^*\right); \quad (39)$$

$$K_{n|L}=K_{n-1|L-1}-\frac{K_{n-1|L-1} x_n^* x_n^{*T} K_{n-1|L-1}}{1+x_n^{*T} K_{n-1|L-1} x_n^*}; \quad (40)$$

Тут

$$x_{n-L+1}^*=\frac{x_{n-L+1}-X_{n|L}^T D_{n|L}^{-1} r_{n-L+1,1}}{\sqrt{\alpha_{n-L+1,0}}}; \quad (41)$$

$$x_n^*=\frac{x_n-X_{n-1|L}^T D_{n-1|L}^{-1} r_{n-1,1}}{\sqrt{\alpha_{n,0}}}; \quad (42)$$

$$y_{n-L+1}^*=\frac{y_{n-L+1}-Y_{n|L} D_{n|L}^{-1} r_{n-L+1,1}}{\sqrt{\alpha_{n-L+1,0}}}; \quad (43)$$

$$y_n^*=\frac{y_n-Y_{n-1|L} D_{n-1|L}^{-1} d_{n-1}}{\sqrt{\alpha_n}}. \quad (44)$$

4. Особливості оцінювання для випадку стаціонарних завад

Розглянемо деякі особливості оцінювання для випадку стаціонарних завад.

З урахуванням введених позначень перетворений вихідний сигнал y_n^* може бути записаний так

$$y_n^*=y_n-r_{1,n-1}^T R_{n-1|L-1}^{-1} Y_{n-1|L-1}. \quad (45)$$

Визначимо добуток $r_{1,n-1}^T R_{n-1|L-1}^{-1} Y_{n-1|L-1}$, користуючись блоковим поданням векторів і матриць, що входять до нього.

У зв'язку з тим, що

$$r_{n-1,1}^T=\left(\rho_{n-1}:r_{n-2,1}^T\right), \quad Y_{n-1}^T=\left(Y_{n-2|L-1}^T:y_n\right),$$

$$R_{n-1|L-1}=\begin{bmatrix} R_{n-2|L-2} & r_{n-2,1}^T \\ r_{n-2,1} & 1 \end{bmatrix},$$

а зворотна матриця обчислюється аналогічно (14)

$$R^{-1}_{n-1|L-1} = \tag{46}$$

$$= \begin{bmatrix} R^{-1}_{n-2|L-2} + \dfrac{R^{-1}_{n-2|L-2} r_{n-2,1} r^T_{n-2,1} R^{-1}_{n-2|L-2}}{1 - r^T_{n-2,1} R^{-1}_{n-2|L-2} r_{n-2,1}} & -\dfrac{R^{-1}_{n-2|L-2} r_{n-2,1}}{1 - r^T_{n-2,1} R^{-1}_{n-2|L-2} r_{n-2,1}} \\ -\dfrac{r^T_{n-1,1} R^{-1}_{n-1}}{1 - r^T_{n-2,1} R^{-1}_{n-2|L-2} r_{n-2,1}} & \dfrac{1}{1 - r^T_{n-2,1} R^{-1}_{n-2|L-2} r_{n-2,1}} \end{bmatrix},$$

то

$$r^T_{1,n-1} R^{-1}_{n-1|L-1} Y_{n-1|L-1} = \left(\rho_{n-1} : r^T_{n-2,1}\right) \times$$

$$\times \begin{bmatrix} R^{-1}_{n-2|L-2} + \dfrac{R^{-1}_{n-2|L-2} r_{n-2,1} r^T_{n-2,1} R^{-1}_{n-2|L-2}}{\alpha_{n-1,0}} & -\dfrac{R^{-1}_{n-2|L-2} r_{n-2,1}}{\alpha_{n-1,0}} \\ -\dfrac{r^T_{n-2,1} R^{-1}_{n-2|L-2}}{\alpha_{n-1,0}} & \dfrac{1}{\alpha_{n-1,0}} \end{bmatrix} \times$$

$$\times \begin{pmatrix} Y_{n-2|L-2} \\ \dots \\ y_{n-1} \end{pmatrix} = r^T_{n-2,1} R^{-1}_{n-2|L-2} Y_{n-2|L-2} + \tag{47}$$

$$+ \left(\frac{\rho_{n-1} - r^T_{n-2,1} R^{-1}_{n-2|L-2} r_{n-2,1}}{\alpha_{n-1,0}} \right) \times \left(y_{n-1} - r^T_{n-2,1} R^{-1}_{n-2|L-2} Y_{n-2|L-2} \right),$$

де $\alpha_{n-1,0} = 1 - r^T_{n-2,1} R^{-1}_{n-2|L-2} r_{n-2,1}$. Але

$$\rho_{n-1} - r^T_{n-2,1} R^{-1}_{n-2|L-2} r_{n-2,1} = \alpha_{n-1,1},$$

$$y_{n-1} - Y^T_{n-2|L-2} R^{-1}_{n-2|L-2} r_{n-2,1} = y^*_{n-1}.$$

Тому остаточно маємо

$$Y^T_{n-1,1} R^{-1}_{n-1|L-1} r^T_{n-1,1} = Y^T_{n-2,L-2} R^{-1}_{n-2|L-2} r^T_{n-2,1} + \frac{\alpha_{n-1,1}}{\alpha_{n-1,0}} y^*_{n-1}. \tag{48}$$

Після підстановки (48) в (46) отримуємо

$$y^*_n = y_n - \frac{\alpha_{n-1,1}}{\alpha_{n-1,0}} y^*_{n-1} - Y^T_{n-2,2} R^{-1}_{n-2|L-2} r_{n-2,1}. \tag{49}$$

Добуток $Y^T_{n-2|L-2} R^{-1}_{n-2|L-2} r_{n-2,1}$, в свою чергу, обчислюється рекурентно за формулою (47). Таким чином, змінна y^*_n обчислюється наступним чином:

$$y^*_n = y_n - \sum_{k=1}^{n-1} \frac{\alpha_{n-k,k}}{\alpha_{n-k,0}} y^*_{n-k}. \tag{50}$$

Коефіцієнти $\alpha_{n-k,k}$, що входять в даний вираз, допускають рекурентне обчислення. Для цього визначимо величини $r^T_{i,j} R^{-1}_l r_{k,m}$.

У зв'язку з тим, що згідно з прийнятими позначеннями

$$r^T_{i,j} = \left(\rho_i : r^T_{i-1,j}\right), \; r^T_{k,m} = \left(\rho_k : r^T_{k-1,m}\right),$$

а матриця R^{-1}_l обчислюється відповідно до (43), то

$$r^T_{i,j} R^{-1}_l r_{k,m} = \left(\rho_i : r^T_{i-1,j}\right) \times$$

$$\times \begin{bmatrix} \dfrac{1}{\alpha_{l-1,0}} & -\dfrac{r^T_{l-1,1} R^{-1}_{l-1}}{\alpha_{n-1,0}} \\ -\dfrac{R^{-1}_{l-1} r_{l-1,1}}{\alpha_{l-1,0}} & R^{-1}_{l-1} + \dfrac{R^{-1}_{l-1} r_{l-1,1} r^T_{l-1,1} R^{-1}_{l-1}}{\alpha_{l-1,0}} \end{bmatrix} \begin{pmatrix} \rho_k \\ \dots \\ r_{k-1,m} \end{pmatrix} = \tag{50}$$

$$= r^T_{i-1,j} R^{-1}_{l-1} r_{k-1,m} + \frac{\left(\rho_i - r^T_{i-1,j} R^{-1}_{l-1} r_{l-1,1}\right)\left(\rho_k - r^T_{k-1,m} R^{-1}_{l-1} r_{l-1,1}\right)}{\alpha_{l-1,0}}.$$

Тут $\alpha_{l-1,0} = 1 - r^T_{l-1,0} R^{-1}_{l-1} r_{l-1,0}$.

Це дозволяє записати для любого моменту часу

$$r^T_{n+k-1,k} R^{-1}_{n-1|L-1} r_{n+m-1,m} =$$

$$= r^T_{n+k-2,k+1} R^{-1}_{n-2|L-2} r_{n+m-2,m+1} -$$

$$- \frac{\left(\rho_{n+k-2} - r^T_{n+k-2,k} R^{-1}_{n-2|L-2} r_{n-1,1}\right)\left(\rho_{n+m-2} - r^T_{n+m-2,m} R^{-1}_{n-2|L-2} r_{n-1,1}\right)}{\alpha_{n-2,0}} = \tag{51}$$

$$= r^T_{n+k-2,k} R^{-1}_{n-2|L-2} r_{n+m-2,m} - \frac{\alpha_{n-1,k} \alpha_{n-1,m}}{\alpha_{n-2,0}}.$$

Підставляючи (51) в (47) та ітеруючи, отримуємо

$$\alpha_{n,j} = \rho_j - \sum_{k=1}^{n-1} \frac{\alpha_{n-k,k} \alpha_{n-k,j+k}}{\alpha_{n-k,0}}. \tag{52}$$

Таким чином, змінна y^*_n, використовувана в алгоритмі, може бути визначена за формулами (50), (52) без обчислення зворотної матриці R^{-1}_{n-1}.

Зупинимося детальніше на властивостях перетвореного процесу y^*_n. Розглянемо центровану складову процесу y^*_n.

$$\overline{y}^*_n = y^*_n - M\left\{y^*_n\right\} = \xi_n - \Xi^T_{n-1|L-1} R^{-1}_{n-1|L-1} r_{n-1,1}. \tag{53}$$

Обчислимо $M\left\{\left(\overline{y}^*_n\right)^2\right\}$:

$$M\left\{\left(\overline{y}^*_n\right)^2\right\} = M\left\{\left(\xi_n - \Xi^T_{n-1|L-1} R^{-1}_{n-1|L-1} r_{n-1,1}\right)^2\right\} =$$

$$= M\left\{\xi^2_n\right\} - 2M\left\{\xi_n \Xi^T_{n-1|L-1} R^{-1}_{n-1|L-1} r_{n-1,1}\right\} + \tag{54}$$

$$+ M\left\{r^T_{n-1,1} R^{-1}_{n-1|L-1} \Xi_{n-1|L-1} \Xi^T_{n-1|L-1} R^{-1}_{n-1|L-1} r_{n-1,1}\right\}.$$

Внаслідок симетричності матриці R^{-1}_{n-1}

$$r^T_{n-1,1} R^{-1}_{n-1|L-1} \Xi_{n-1|L-1} = \Xi^T_{n-1|L-1} R^{-1}_{n-1|L-1} r_{n-1,1}.$$

Так як $M\left\{\Xi_{n-1} \xi_n\right\} = r_{n-1,1}, M\left\{\Xi_{n-1} \Xi^T_{n-1}\right\} = R^{-1}_{n-1}$, то після взяття операції математичного сподівання отримуємо

$$M\left\{\left(\overline{y}^*_n\right)^2\right\} = 1 - r^T_{n-1,1} R^{-1}_{n-1|L-1} r_{n-1,1} = \alpha_{n-1,0}, \tag{55}$$

тобто параметр $\alpha_{n-1,0}$ визначає дисперсію процесу y^*_n.

Аналогічно визначимо значення взаємної кореляційної функції між завадою ξ_{n-j} $(j = 1,2,\dots,n-1)$ та центрованою складовою y^*_n

$$M\left\{\xi_{n+j} \overline{y}^*_n\right\} =$$

$$= M\left\{\xi_{n+j} \xi_n - \Xi^T_{n-1|L-1} R^{-1}_{n-1|L-1} r_{n-1,1} \xi_{n+j}\right\} =$$

$$= \rho_j - r^T_{n-1,1} R^{-1}_{n-1|L-1} r_{n-j+1,j+1}.$$

Звідси слід, що коефіцієнти $\alpha_{n,j}$ являють собою значення взаємної кореляційної функції центрованої складової сигналу y^*_n і завади ξ_n.

Обчислимо $M\left\{\overline{y}^*_n \overline{y}^*_{n-1}\right\}$.

Скориставшись виразом (54) і тим, що $\Xi^T_{n|L} = \left(\Xi^T_{n-1|L-1} : \xi_n\right)$, отримуємо

$$M\left\{\overline{y}^*_n \overline{y}^*_{n-1}\right\} = M\left\{\left(\xi_n - \Xi^T_{n-1|L-1} R^{-1}_{n-1|L-1} r_{n-1,1}\right) \overline{y}^*_{n-1}\right\}.$$

Визначимо рекурентну формулу для перерахунку вектора $\Xi^T_{n-1|L-1} R^{-1}_{n-1|L-1} r_{n-1,1}$, що входить в даний вираз,

використовуючи блочне представлення співмножників, які входять до нього. В цьому випадку

$$\Xi^T_{n-1|L-1}R^{-1}_{n-1|L-1}r_{n-1,1}=\left(\Xi^T_{n-1|L-1}:\xi_n\right)R^{-1}_{n-1|L-1}\begin{pmatrix}\rho_{n-1}\\ \dots\\ r_{n-2,1}\end{pmatrix},$$

де $R^{-1}_{n-1|L-1}$ визначається формулою (46). Після нескладних перетворень маємо

$$\Xi^T_{n-1|L-1}R^{-1}_{n-1|L-1}r_{n-1,1}=\Xi^T_{n-2|L-2}R^{-1}_{n-2|L-2}r_{n-2,1}+$$

$$+\left(\xi_{n-1}-\Xi^T_{n-2|L-2}R^{-1}_{n-2|L-2}r_{n-2,1}\right)\frac{\alpha_{n-1,1}}{\alpha_{n-1,0}}.$$

В наших позначеннях

$\xi_{n-1}-\Xi^T_{n-2|L-2}R^{-1}_{n-2|L-2}r_{n-2,1}=\overline{y}^*_{n-1}$. Таким чином,

$$\Xi^T_{n-1|L-1}R^{-1}_{n-1|L-1}r_{n-1,1}=\Xi^T_{n-2|L-2}R^{-1}_{n-2|L-2}r_{n-2,1}+\overline{y}^*_{n-1}\frac{\alpha_{n-2,1}}{\alpha_{n-2,0}}. \quad (56)$$

Тоді

$$M\left\{\overline{y}^*_n\overline{y}^*_{n-1}\right\}=$$

$$=M\left\{\left(\xi_{n-1}-\Xi^T_{n-2|L-2}R^{-1}_{n-2|L-2}r_{n-2,1}-\overline{y}^*_{n-1}\frac{\alpha_{n-1,1}}{\alpha_{n-1,0}}\right)\overline{y}^*_{n-1}\right\}=$$

$$=M\left\{\left(\xi_n\overline{y}^*_{n-1}-\Xi^T_{n-2|L-2}R^{-1}_{n-2|L-2}r_{n-2,1}\overline{y}^*_{n-1}-\left(\overline{y}^*_{n-1}\right)^2\frac{\alpha_{n-1,1}}{\alpha_{n-1,0}}\right)\right\}=$$

$$=-M\left\{\xi_n\Xi_{n-2|L-2}\right\}R^{-1}_{n-2|L-2}r_{n-2,1}+ \quad (57)$$

$$+r^T_{n-2,1}R^{-1}_{n-2|L-2}M\left\{\Xi_{n-2|L-2}\Xi^T_{n-2|L-2}\right\}R^{-1}_{n-2|L-2}r_{n-2,1}=$$

$$=-r^T_{n-2,1}R^{-1}_{n-2|L-2}r_{n-2,1}+r^T_{n-2,1}R^{-1}_{n-2|L-2}r_{n-2,1}=0.$$

Звідси впливає, що

$$M\left\{\left(\Xi^T_{n-2|L-2}R^{-1}_{n-2|L-2}r_{n-2,1}\overline{y}^*_{n-1}\right)\right\}=0,$$

тобто

$$M\left\{\Xi_{n-2|L-2}\overline{y}^*_{n-1}\right\}=0. \quad (58)$$

Враховуючи (46), (48), (58), маємо

$$M\left\{\overline{y}^*_n\overline{y}^*_{n-2}\right\}=$$

$$=M\left\{\left(\xi_n-\Xi^T_{n-2|L-2}R^{-1}_{n-2|L-2}r_{n-2,1}-\overline{y}^*_{n-1}\frac{\alpha_{n-2,1}}{\alpha_{n-1,0}}\right)\overline{y}^*_{n-2}\right\}=$$

$$=M\left\{\left(\xi_n\overline{y}^*_{n-1}-\Xi^T_{n-3|L-3}R^{-1}_{n-3|L-3}r_{n-3,1}-\overline{y}^*_{n-2}\frac{\alpha_{n-2,1}}{\alpha_{n-2,0}}-\overline{y}^*_{n-1}\frac{\alpha_{n-1,1}}{\alpha_{n-1,0}}\right)\overline{y}^*_{n-2}\right\}.$$

Після взяття операції математичного сподівання так, як було зроблено вище, отримуємо

$$M\left\{\overline{y}^*_{n-1}\overline{y}^*_{n-2}\right\}=0,$$

$$M\left\{\xi_n\overline{y}^*_{n-2}\right\}=\alpha_{n-2,1},$$

$$M\left\{\left(\overline{y}^*_{n-2}\right)^2\right\}=\alpha_{n-2,0}.$$

Внаслідок (57) $r^T_{n-3,1}R^{-1}_{n-3}\ M\left\{\Xi_{n-3}\overline{y}^*_{n-2}\right\}=0$ і остаточно маємо

$$M\left\{\overline{y}^*_n\overline{y}^*_{n-2}\right\}=0.$$

Обчислюючи $M\left\{\overline{y}^*_n\overline{y}^*_{n-3}\right\}$, $M\left\{\overline{y}^*_n\overline{y}^*_{n-4}\right\}$ і т.д., переконуємося, що всі ці математичні сподівання добутків дорівнюють нулю. Отже, центрована складова перетвореного процесу, яка визначається співвідношенням (57), являє собою дискретний білий шум, тобто співвідношення (44) описує перетворення випадкового вектора з корельованими складовими в випадковий вектор з некорельованими складовими.

Висновки

Таким чином, розглянуто задачу побудови рекурентної форми алгоритму поточного регресійного аналізу, що дозволяє здійснювати оцінювання невідомих параметрів при наявності стаціонарних корельованих завад.

Отримано основні співвідношення, які описують процеси накопичення нової і скидання застарілої інформації. Показано, що алгоритми, які розглядаються, використовують перетворення випадкового вектора з корельованими складовими в випадковий вектор з некорельованими складовими.

Конфлікт інтересів

Автори не заявляють конфлікту інтересів.

References

[1] Widrow B. Adaptive switching circuits, / B. Widrow, M. Hoff // IRE WESCON Convention Record. Part 4. New York: Institute of Radio Engineers, 1960. – P.96–104.

[2] Либероль Б.Д. Исследование сходимости одношаговых адаптивных алгоритмов идентификации. / Б.Д. Либероль, О.Г. Руденко, А.А. Бессонов // Проблемы управления и информатики. – 2018. –, 5. – С.19 – 32.

[3] Руденко О.Г. Регуляризованный алгоритм обучения адалины в задаче оценивания нестационарных параметров / О.Г. Руденко, А.А. Бессонов // Управляющие системы и машины. – 2019. –1. – С.22 – 30.

[4] Перельман И.И. Оперативная идентификация объектов управления./ И.И. Перельман //–М.: Энергоиздат,1982. –272 с.

[5] Руденко О.Г. Модифицированный алгоритм текущего регрессионного анализа в задачах идентификации и прогнозирования. / О.Г. Руденко, И.Д. Теренковский, А. Штефан, Г.А. Ода // Радиоэлектроника и информатика. – 1998. –№ 4(05). – С.58–61

[6] Rudenko O.G. Adaline robust multistep training algorithm / O.G. Rudenko, O.O. Bezsonov // Control Systems and Computers. – 2020. – № 3. – P.15–27

[7] Аведьян Э.Д. Рекуррентный метод наименьших квадратов при коррелированных помехах / Э.Д. Аведьян // Автоматика и телемеханика . –1975.-5.-С.67-75

[8] Медведев Г.А. Рекуррентное оценивание при помощи коррелированных наблюдений / Г.А. Медведев // Автоматика и телемеханика . 1974.-5.-С.110-116

[9] Медведев Г.А. О рекуррентных оценках по коррелированным наблюдениям / Г.А. Медведев, Г.А. Хацкевич // Автоматика и телемеханика .-1979.-8.-С.69-75

[10] Джонстон Дж. Эконометрические методы./ Дж. Джонстон - М.: Статистика,1980. -444с.

[11] Фаддеев Д.К. Вычислительные методы линейной алгебры. / Д.К. Фаддеев, В.Н. Фаддеева – М.: Физматгиз,1963. – 735 с.

Надійшла до редколегії 29.01.2021

УДК 37 DOi 10.30837/ bi.2021.1(96).12

Я.О. Гончар[1], О.В. Вечур[2]

[1]ХНУРЕ, м. Харків, Україна, yaroslav.honchar@nure.ua, ORCID iD: 0000-0002-8637-8025

[2]ХНУРЕ, м. Харків, Україна, alexander.vechur@nure.ua, ORCID iD: 0000-0001-9605-1475

ОСНОВНІ ПРИНЦИПИ СТВОРЕННЯ ТА ДІЯЛЬНОСТІ МУЛЬТИМЕДІЙНИХ НАВЧАЛЬНИХ СИСТЕМ

Розглянуто проблеми і запропоновано рішення для покращення в освітньому процесі інформаційного обміну шляхом індивідуалізації, локалізації та уніфікованості понять. Знайдено проблеми й рішення щодо швидкості навчання: використання ефективного апаратного забезпечення, порядок подання контенту і його структурування. Досліджено мотиваційну складову при розвитку та зацікавленості вивчення знань. Запропоновано принципи створення взаємодії при роботі з навчальним контентом як здобувачів освіти, так і редакторів. З'ясовано можливі варіанти систем перевірки рівня знань для особистої оцінки. Розглянуто проблеми отримання освітнього ступеню при повністю автоматизованій дистанційній перевірці та наведено рекомендацію із створення процесу тестування для такого присвоєння.

ДИСТАНЦІЙНЕ НАВЧАННЯ, ІНДИВІДУАЛЬНЕ НАВЧАННЯ, МОТИВАЦІЯ, ОСВІТА, ПРОГРАМНА СИСТЕМА, ТЕСТУВАННЯ

Гончар Я. А., Вечур А. В. Основные принципы создания и деятельности мультимедийных обучающих систем. Рассмотрены проблемы и предложены решения для улучшения в образовательном процессе информационного обмена путем индивидуализации, локализации и унифицированности понятий. Найдено проблемы и решения по поводу скорости обучения: использование эффективного аппаратного обеспечения, порядок представления контента и его структурирование. Исследована мотивационная составляющая при развитии и заинтересованность изучения знаний. Предложены принципы создания взаимодействия при работе с учебным контентом как учащихся, так и редакторов. Выяснено возможные варианты систем проверки уровня знаний для личной оценки. Рассмотрены проблемы получения образовательного степени при полностью автоматизированной дистанционной проверке и приведены рекомендации по созданию процесса тестирования для такого присвоения.

ДИСТАНЦИОННОЕ ОБУЧЕНИЕ, ИНДИВИДУАЛЬНОЕ ОБУЧЕНИЕ, МОТИВАЦИЯ, ОБРАЗОВАНИЕ, ПРОГРАММНЫЕ СИСТЕМЫ, ТЕСТИРОВАНИЕ

Honchar Y. O., Vechur O. V. Basic principles of creation and operation of multimedia teaching systems. Problems and solutions were considered for improvement of information exchange in the educational process by individualization, localization, and unification of terms. Problems and solutions to the speed of learning were found: the use of effective hardware, the order of content presentation and its structuring. The motivational component in the development and interest in the study of knowledge has been studied. The principles of creating interaction when working with educational content of both students and editors were proposed. Possible options for knowledge assessment systems for personal assessment have been identified. The problems of obtaining an educational degree with fully automated remote testing are considered and a recommendation for creating a testing process for such assignment is given.

DISTANCE LEARNING, EDUCATION, INDIVIDUAL LEARNING, MOTIVATION, SOFTWARE SYSTEM, TESTING

Вступ

Досить поширеною у світі є практика довготривалого навчання. Приблизно до 6-ти років виконується виховання батьками та вихователями в дошкільних закладах, потім до 18-ти років – шкільне навчання, а на додачу ще пропонують закінчити й заклад вищої освіти, що бере на себе ще мінімум 4 роки. При очікуваній середній тривалості життя в 70 років це майже третина всього виділеного часу, і навіть це якщо закінчувати лише один ЗВО.

Є також різноманітні інтернет-ресурси та програмні системи для навчання, але для присвоєння освітнього ступеня вони здебільшого не вповноважені.

Чи дійсно ми правильно розподіляємо час на навчання? Чи ефективно витрачаємо його для здобуття саме необхідних знань у необхідній кількості? Як насамперед людський час, так і вся навчальна інфраструктура наразі є досить дорогоцінними, щоб їх ресурс марно витрачати. Освітній та науковий процеси досить сповільнені через зайві та неправильні процедури, що в них виконуються. Проблема ефективності навчання є дуже актуальною.

У цій роботі пропонується проаналізувати наявний освітній процес, віднайти існуючі проблеми та дослідити можливі рішення на наявність переваг та недоліків. Також пропонується розглянути принципи створення навчальних систем відповідно до виявлених проблем та запропонованих рішень та визначити норми для систем перевірки рівня знань з курсів.

1. Розгляд і пропозиції вирішення проблем інформаційного обміну

Кожен індивідуум має власний механізм сприйняття, обробки, запам'ятовування й пригадування інформації. Це ускладнює визначення швидкості вчителя, при якій навчання буде найефективнішим. З одного боку, досить повільне надання навчального матеріалу, як це здебільшого й відбувається через обмеженість швидкості сприйняття звукової і текстової

інформації, провокує бездіяльність учня або забування знань через тривалі формулювання, що зменшує його концентрацію у навчанні. З іншого боку — завелика швидкість вчителя не дозволяє учневі встигнути запам'ятати всі знання, а отже і потенційно залежну від них інформацію. Тому навчання має бути індивідуальним.

Важливою складовою точного навчання є використання однієї мови – тієї, якою найкраще володіє здобувач освіти. У навчальних системах дозволяється і навіть потрібна можливість викладання декількома мовами задля локалізації і доступності, але при цьому повинна бути можливість пройти курс однією мовою. Навіть при вивченні іноземних мов, які в даному випадку споріднюються із об'єктами навчання і не враховуються як мови передачі основної теорії, саме правила в такому випадку мають бути якомога зрозумілішими, а тому повинні бути передані мовою, якою учень найлегше сприймає інформацію та запам'ятовує, утворюючи правильні зв'язки асоціативності. При недотриманні такої практики виникають наступні проблеми:

– повне або часткове нерозуміння навчальної теорії, що сповільнює процес, зменшує мотивацію продовжувати через дискомфорт;

– неправильне розуміння теорії, яка може хибно використовуватися в подальшому навчанні та в повсякденному житті;

– зменшення концентрації через паралельне виконання перекладу теорії, особливо якщо та надається вголос у реальному часі (відео або реальним викладачем).

Такі ж проблеми можуть проявлятися навіть при навчанні рідною мовою, хоча й значно рідше. В основному таке трапляється при вживанні великої кількості синонімів, що є досить поширеною практикою для літератури, де намагаються уникати повторів і тавтологій. Але загальновживане не обов'язково є корисним, тож хоча матеріал і стає звучати мелодійніше, це порушує всі асоціативні зв'язки між об'єктами, що обговорюються. Таке явище накладає наступні умови для коректної передачі інформації:

– отримувач має знати, що вживані синоніми є дійсно ними, а не різними поняттями;

– джерело інформації використовує саме синоніми для усунення повторів, а не інші поняття, що мають якісь відмінності від об'єктів обговорення;

– отримувач має самостійно прив'язувати такі поняття до їх об'єктів обговорення, що тягне за собою пригадування значення нововживаних слів та пошук подібних об'єктів, про які наразі йде мова, для встановлення зв'язку, що це саме вони обговорюються, а не змінюється тема.

Якщо здобувач освіти та джерело знань працюють у потоковому режимі дотримуючись такого принципу, то їм доведеться відволікатися на виконання наведених умов, що не дає можливості сконцентровано навчатися. Тому знання мають бути уніфіковані. В ідеальному випадку кожен об'єкт навчання повинен мати лише одну назву без вживання синонімів. І навіть якщо утворюватимуться повтори, це не примушуватиме суб'єктів навчання марно витрачати зусилля на відновлення асоціативності переданих об'єктів.

2. Розгляд і пропозиції вирішення проблем швидкості навчання

Крім розглянутих проблем недоотримання інформації або втрати її правильності і точності при передачі є й інші проблеми, що впливають на швидкість. У цьому розділі розглядаються проблеми, де інформація передається без проблем, але досить повільно засвоюється.

Пошукові системи та стандартні бази знань на кшталт вікіпедії не є ефективними для навчання, а лише впорядковують ці знання. Так, вони надають інформацію про знання, що запитується, але в такому випадку учень не знатиме про знання, якщо не знатиме взагалі про їх існування, принаймні найменування. Тож для систем навчання доцільно формувати курси та програми із переліком таких найменувань і бажано з їх описом, який може бути уточнений учнем особисто з використанням пошукової системи. Системи мають подавати знання у логічному порядку, а не просто бути індексатором.

Інформація про об'єкт навчання має відповідати на базові питання, які можуть виникати щодо нього згідно його властивостей. Крім цього знання варто подавати не просто абзацом тексту, а чітко розбиваючи текст на властивості щодо об'єкту знань, окремо їх виділяючи. Для пришвидшення освоєння необхідно виконувати систематизацію теорії, використовуючи порівняльні таблиці, наглядні схеми (рисунки) з якомога ширшою та детальнішою демонстрацією об'єкта навчання, анімації, що наглядно демонструють життєвий цикл цього об'єкта та прив'язкою до них інтерактивності, що дозволить учню змінити параметри середовища або об'єкта для спостереження змін і кращого розуміння показаної системи. Тому знання мають бути структурованими

Для ефективнішого навчання здобувач освіти має виконувати у відсотковому відношенні операцій із навчання якомога більше, ніж операцій видобування теорії. Зайві операції відволікають учня й можуть зменшувати його концентрацію. Тож сильно пов'язана між собою інформація має подаватися дуже близько – на одному екрані, для проведення аналогій та іншого аналізу здобувачем відповідно до його методів.

Попередній підхід може утворювати досить великі структури теорії, насамперед таблиці і схеми, тож і екран для них має бути немалим. Так, теорія може бути зображена й на малому екрані, проте її перегляд потребуватиме багато дій із переміщення, збільшення й

зменшення. Тож тут варто або рекомендувати більший дисплей, або розміщувати пов'язану теорію так, щоб та була нормального розміру та вміщалася в дисплей.

Швидкість відгуку системи напряму впливає на швидкість навчання. По-перше, тривалість відгуку сповільнює роботу користувача, як і в будь-якій іншій системі. Тож при розробці систем варто або використовувати високопродуктивні програмні рішення або використовувати більш швидкі машини для серверів. У разі якщо затримки спричинені клієнтським застосунком, то оптимізація роботи сервера не дасть якихось позитивних результатів і можна виконувати лише оптимізацію програмного коду, що виконується на клієнтському обладнанні, яке не підлягає заміни. Рекомендованим є також спрощення інформаційного обміну між веб-додатком та сервером:

– використання ефективних (проте безпечних) правил кешування;

– використання алгоритмів стиснення даних, що можуть застосовуватися за замовчуванням на відповідних форматах зображень, відео та аудіо;

– зменшення кількості послідовних запитів до сервера із допустимим збільшенням розміру даних.

По-друге, користувач не просто бажає виконати якісь обробки даних, а використовує систему в навчальних цілях, тож йому варто утримувати певні обсяги даних у власній пам'яті, здебільшого у буферній та короткочасній, бо в довготривалій зберігається вже вивчена інформація і тому може використовуватися лиш для підкріплення буферної та короткочасної. Внаслідок цього простежується не лінійна, а більша складність вивчення матеріалу відносно затримок системи.

По-третє, на час затримок користувач використовує свою буферну й короткочасну пам'яті в інших цілях, наприклад спостереження за оновленням сторінки або принаймні переглядом анімації завантаження даних (що вважається кращою практикою ніж взагалі відсутність повідомлень про стан системи), які можуть досить сильно завантажувати буферну пам'ять, а тому й провокувати її випадкове очищення, що ускладнює процес. Як рішення може виступати часткове оновлення сторінок та зменшення виконання анімацій, бо мозок реагує здебільшого на зміни об'єктів сприйняття, а не на їх присутність [1].

3. Розгляд і пропозиції вирішення проблем мотивації та зацікавленості в навчанні

Згідно піраміди Абрагама Маслоу [2], творчість, пізнання і самовираження є близькими до вершини потреб людини, тож для її досягнення потрібно задовольнити низку первинних потреб.

Потребами першої необхідності є їжа, відпочинок, сон і подібні. Хоча системі навчання не передбачено їх задовольняти, вона не має заважати це виконувати, інакше у користувачів виникатимуть негативні ефекти що можуть бути в діапазоні від просто зменшеної концентрації, адже увага приділятиметься незадоволеним потребам, до взагалі виникнення ненависті до системи за тривалої відсутності вільного часу на їх задоволення. Тому навчання має проводитися за гнучким графіком, коли учень сконцентрований здебільшого на ньому. Має бути можливість призупинити навчання в будь-який час та продовжити із місця зупинки пізніше.

Далі за потребами слідує безпека, і хоча знову освітні системи за це не відповідають, вони не мають нести загрозу користувачу. Мають виконуватися умови кібербезпеки:

– конфіденційність персональних даних та дій;

– безпека інтернет-підключення та дійсність сертифікату сайту.

Також необхідна безпека здоров'я, яка може бути визначена:

– відображенням рекомендацій та нагадувань при користуванні системою (щодо розміру екрану та відстанню до нього від очей, яскравості, утриманням положення спини, рівня гучності звуку динаміку);

– усуненням низькоконтрастних об'єктів графічного інтерфейсу;

– відображення об'єктів нормального для зручного перегляду розміру;

– усунення різкої і частої зміни кольорів анімацій великих ділянок інтерфейсу;

– усунення шумів у звуках та перепадів гучності, яка задана відповідно до системних налаштувань.

Задоволення соціальних потреб може відбуватися в різних напрямках:

– оцінка й коментування розглянутої теорії, її обговорення з іншими учасниками, що теж дійшли до такої або пройшли її;

– створення форумів із учасниками освітнього процесу для різноманітних обговорень та спільної наукової діяльності;

– створення чатів за навчальними курсами та дисциплінами із можливостями переписування, говоріння та демонстрації робіт.

Для задоволення потреби в повазі та самооцінці система може:

– демонструвати, що саме навчився робити здобувач освіти;

– узагальнювати, як учень розвивався від початку роботи із системою;

– формувати рейтинги.

Далі ми підходимо до потреб у пізнанні, що й задовольняються навчальними системами. Проте ця потреба знаходиться досить високо у піраміді і вільний графік у довгостроковій перспективі може навіть зашкодити, бо може виникнути проблема із небажанням повертатися до навчання, спричинене забуттям фундаментальної теорії, що не пригадувалася тривалий час. Для цього пропонується 2 рішення.

Перше полягає в наданні можливості побачити «вижимку» з навчального матеріалу або якимось

чином отримувати швидку зрозумілу довідку за забутими поняттями прямо у процесі навчання.

Друге рішення полягає у стабільності графіку, тобто постійному нагадуванні, що варто вчитися, або у формуванні графіку самим учнем за наданими принципами часового менеджменту. Це дозволить зменшити тяжкість повернення здобувача освіти до активного процесу навчання, якщо той раптом захопився іншими справами і забув за систему, а повернення може здаватися йому складним.

Піраміда Маслоу хоч і має закріплені рівні цінностей, але в різних особистостей вони можуть бути переставлені і кардинально відрізнятися, тож варто також розглянути й подальші потреби – естетичні й творчі.

Щодо естетичних потреб мультимедійні навчальні системи повинні мати захопливу подачу матеріалу, тобто гарний інтерфейс і якісний звук, що може бути навіть виражений озвученням контенту. Це особливо важливо для здобувачів освіти малому віку, які потребують цікавість від системи настільки високу, що дасть змогу працювати саме з нею, а не виконувати якусь іншу діяльність. Старші здобувачі хоч і можуть мати інші стимули для вивчення, наприклад для освоєння професії, проте їм теж бажано подавати якісний мультимедійний матеріал. Крім того, що інтерфейс має працювати без візуальних помилок та відповідати гарним показникам ергономіки, його елементи повинні бути наближені до ігрового стилю, що передбачає використання широкого діапазону кольорів, різноманітних форм елементів, анімацій і звуків.

До мультимедійних наукоємних ресурсів для навчання можуть слугувати й аудіокниги, але варто оцінювати їх реальну ефективність: частка сприйняття звукової інформації відносно усієї становить менше 10%. Навіть якщо порівнювати викладання вчителем та прослуховування книги з можливістю прокручування й паузи, аудіо є досить вузьким каналом зв'язку, тим паче що легко може бути зашумлений середовищем, де подальше навчання буде досить ускладнене.

На заміну аудіокнигам можна поставити відеоуроки, які теж можна дивитися у вільний час, але цього разу застосовується візуальний і звуковий канали, що в рази покращує ефективність, а тому й швидкість навчання. Проте тепер варто розуміти, що це ще потокова інформація, а тому в ідеальному випадку її швидкість має бути однаковою із швидкістю сприйняття здобувачем освіти. Також важливо зауважити, що пошук у такому матеріалі ускладнений, бо потребує розпізнавання відео та аудіо або окремої від них подачі у візуально-статичному форматі (схеми, таблиці), які в свою чергу ставлять під сумнів існування відео через тяжкість локалізації багатьма мовами, навіть створення субтитрів, які натомість сприймаються тяжче. Тож ефективна система навчання може мати відеокурси, але ефективніша буде та, що матиме більше інтерактивності у своїх матеріалах.

Наступними у потребах є самовираження та творчість, які здебільшого подано за принципом зовнішнього примусу, тобто коли здобувача освіти змушують виконати певні завдання. І як не дивно, це працює в залежності від особистості – вона або ігнорує їх, або примушено і без великого ентузіазму в рамки поставлених вимог.

Для зменшення імовірності відторгнення учня від завдання не рекомендується зразу навантажувати його складнощами, а спершу навчити його виконувати елементарні та розбивати складні завдання на елементарні.

Якщо до твору таки є якісь вимоги, то їх варто встановлювати перед творчим процесом та вони мають бути доступні під час нього. Зовсім не рекомендується давати завдання, рішення яких не розглядалося при навчанні, бо в такому випадку крім поганого результату можна очікувати зменшений інтерес.

Самовираження має правильно оцінюватися, щоб учасники освітнього процесу розуміли, чому саме виставлена оцінка відповідає твору. Якщо ця оцінка виставляється лише у вигляді балу, то має бути присутня чітка градація, яка частина балу за що відповідає. Якщо чітка градація не можлива, то оцінювання має бути якомога точнішим, адже замала оцінка може показатися здобувачеві як марнотратство часу, а завищена може утворювати хибну самовпевненість, що можна видавати результат невисокої якості для великих досягнень. Такий процес має велику психологічну складову, тож варто враховувати результативність і взагалі активність учня на дисципліні з часом.

Зацікавленість також падає, коли нав'язують або взагалі примушують щось вивчати. Учень розуміє, що не всі предмети йому знадобляться, тож не вивчає їх, принаймні ефективно. Далі поширена ланцюгова реакція – під тиском великої кількості завдань той намагається втекти від зобов'язань, через що трапляються ситуації, коли здобувач освіти втрачає стимул вивчати хоч щось.

Для вирішення такої дилеми та запобігання потрапляння в неї пропонується давати користувачу системи вибір, що саме той може вивчити, надавати довідку, що він після курсу знатиме, що зможе робити та які перспективи його очікують. При перевантаженому справами щоденного графіку особистість шукає марні справи, які можна усунути із щоденного вжитку, тож варто відображати реальну цінність навчальних дисциплін, а не просто наполегливо нав'язувати їх для вивчення.

4. Принципи введення ролей редакторів та створення їм відповідного функціоналу

Найголовнішими користувачами технічних інструментів навчальної системи є вчителі та викладачі. Саме вони мають найбільший пріоритет зі створення навчального контенту так, як є компетентними фахівцями в цій предметній області.

Також при роботі з інструментами може виникнути необхідність у створенні візуального контенту, який можуть створювати інші користувачі — художники та дизайнери. Їх потреба може бути неминучою, бо не всі фахівці мають здібності якісно візуалізувати те, що вони знають та хочуть відобразити. Система може не передбачати універсальних редакторів медіаконтенту, проте його розміщення в якості навчального матеріалу має бути доступним. У разі створення відео може залучатися більший спектр різноманітних фахівців (сценаристів, режисерів, аніматорів), проте і їх користування системою може зводитися лиш до публікації відео.

При розміщенні щойно сказаних медіаматеріалів варто пам'ятати, що вони не передбачають великої інтерактивності. Для ефективного навчання вона наразі необхідна, тож варто створювати об'єкти інтерактивної взаємодії:

– 3D моделі об'єктів, із можливим виділенням їх складових;

– набору об'єктів системи, із якими можна взаємодіяти напряму;

– анімовані системи, в яких можна задавати різноманітні параметри та спостерігати відповідні зміни.

Також варто розуміти, що такі об'єкти можуть виглядати як окремі програмні рішення, які мають пройти всі фази розробки програмного забезпечення від формування вимог до тестування й підтримки. Тож у навчальних системах технічні інструменти для створення інтерактивних систем можуть бути подані у наступних варіантах:

– звичайна вставка програмного коду або інших вихідних об'єктів результату зовнішньої відносно даної системи розробки програмного забезпечення із реалізацією в даній системі можливості їх відтворення здобувачу освіти;

– створення редактору для подальшого створення таких інтерактивних моделей для введення в них об'єктів, задання їх властивостей і функцій їх взаємодії з якомога меншим або взагалі відсутнім програмним кодом.

При варіанті із вставками окремо створених файлів та коду система навчання має перевіряти ті на предмет потенційних загроз (шкідливого коду), перевіряти сумісність версій наданих даних і відтворювача, та взагалі правильно реагувати на надані їй дані, коректно відображати кінцевому користувачу. Це рішення хоч і виглядає простішим, проте несе за собою наступні проблеми:

– безпека цілісності системи та даних користувача при запуску стороннього коду;

– ризики, пов'язані із зовнішнім редактором (ліцензування, підтримка, помилки та інші);

– правильне відтворення поставлених об'єктів;

– сумісність зовнішнього редактора із внутрішнім відтворювачем;

– необхідність у фахівцях, що можуть повноцінно працювати із зовнішнім редактором та правильно інтегрувати у навчальну систему.

Інакше можна ризикнути і обрати більш затратний варіант, а саме створити вбудований редактор для подібних анімацій, в якому вирішується більшість проблем попереднього варіанта, проте натомість можуть виникати інші:

– якість інтерактивних моделей, створених таким чином, залежить від якості редактора;

– створення редактора повинно проходити принаймні базові фази з розробки програмного забезпечення (формування вимог, проектування, розробка, тестування, підтримка);

– створення редактора є досить затратним за часом і ресурсами процесом, і може не відобразити його реальну вартість, бо потенційний прибуток може виявитися меншим у порівнянні із використанням інших аналогів.

Для кращого вибору більш сприятливого підходу можна створювати комбіновані програмні рішення та спостерігати за їх актуальністю, прибутковістю, якістю навчання та простотою процесів.

5. Загальні принципи створення взаємодії при роботі з навчальним контентом

Крім необхідних функціональних вимог, які рекомендовано внести у сучасну навчальну систему, варто також використати і супутні підходи для ще кращої взаємодії із навчальним контентом.

Насамперед це робота з редагуванням навчального матеріалу, що потребує використання персистентних структур даних [3] та забезпечує скасування операцій, їх відновлення або навіть перехід до певної версії контенту.

Перше, що можна реалізувати – це скасування та відновлення операцій. І хоча в сучасних браузерах та API настільних додатків така функція підтримується на текстових даних, для інших же типів варто реалізувати самостійно, чітко визначивши, які дії включатимуться в одну операцію (наприклад, дії виділення та взяття у фокус певного елемента здебільшого приєднують до основних дій). Функціональність має бути ретельно перевіреною, бо є тою, що напряму впливає на введені дані користувача і в складних системах може бути досить комплексною та непередбачуваною.

Версії контенту можна організувати у наступних варіантах:

– збереження версій напряму користувачем;

– автоматичне збереження контенту з можливим утриманням лише обмеженої кількості таких збережень;

– збереження змін між версіями для їх можливого подальшого відтворення;

– автоматичне збереження змін, спричинених кожною операцією.

Рекомендується залишити користувачу можливість зберігати певні версії контенту для їх подальшого точного відновлення. Автоматичне збереження контенту можна застосовувати для уникнення раптових втрат даних.

Рекомендується все-таки використовувати збереження змін а не всього контенту, адже при досить великій системі такі збереження потребуватимуть великих вимог до файлової системи. Збереження саме змін можна виконувати і при вже розглянутому автоматичному збереженні. Варіант збереження кожної операції є більш уніфікованим і дозволяє скасовувати окремі операції, проте вимагає великої активності сервера та мережевого підключення. Тут варто обрати варіант, який краще підходить до системи, відповідно до розміру її бази знань, кількості активних користувачів та редакторів, обчислювальних можливостей сервера.

Краще не видавати напоказ те, що наразі редагують фахівці, тож пропонується рішення збереження чорнового варіанту контенту окремо від опублікованого та виконувати публікацію змін за ініціативи редактора.

Системні редактори контенту можна створити за принципом «WYSIWYG» («Що ви бачите, те й отримаєте»), при цьому уникати втілення стандартних форм введення даних, що потенційно несуть несумісність між інтерфейсами даних форми та самого контенту.

Для ефективнішого управління програмною системою рекомендується реалізувати використання гарячих клавіш. Частина таких здебільшого реалізована за замовчуванням, проте специфічні і відносні до предметної області варто створити самостійно. Якщо доступних клавіатурних скорочень у кожен момент часу невелика кількість, то їх можна виводити користувачу разом зі списком доступних дій. Спершу йому буде складніше освоювати систему, проте згодом це зменшить затримки у користуванні, що підвищить ефективність.

6. Створення узагальнень та тренувальні перевірки знань

Після вивчення курсу варто пригадати всю необхідну теорію та виконати перевірку.

Для повторного пригадування буде досить неефективним проглянути весь матеріал заново. Швидшим варіантом може представлятися перегляд у більш скороченій версії. Тож для запуску процедури пригадування всього вивченого та утримання у пам'яті варто зініціювати пригадування якомога більшої предметної області курсу за якомога менший час.

Скорочення змісту курсу для повторень та узагальнень варто проводити індивідуально, в залежності від надмірності наявної теорії для користувача, тобто саме того, що здобувач уже вивчив.

Навчання можна організувати циклічно, тобто спершу подати найбільш детальну інформацію про об'єкти навчання, а потім проводити навчання все з меншим і меншим обсягом тих же даних. Ці ітерації досить ефективно поєднуються з перевіркою, адже після кожної перевірки можна подавати навчальний матеріал у відповідному до рівня знань ступеню вивчення.

Хоча ітеративний процес повторного вивчення досить гарно описаний, в реальності така процедура займе досить багато часу в залежності від ефективності запам'ятовування та швидкості забування. Також варто наперед визначати, який мінімальний ступінь навченості може бути прохідним, тобто таким, при якому вважається, що навчання пройдено успішно: якщо задати замалий показник, утворюватимуться недостатньо кваліфіковані кадри, а при високому – зростає кількість ітерацій і період навчання, а тому зменшується і кількість випускників.

При досить великій кількості теорії в курсі всю її може бути важко утримати в пам'яті або ефективно пригадувати, тож варто подати скорочений варіант або очікувати гірші результати. Якщо ж навпаки, даних малий обсяг, то при досить частих перевірках здобувач освіти може механічно запам'ятати всі відповіді і таким же механічним чином пройти перевірку та здобути гарну оцінку, хоча сам курс не відкривав, а лиш проходив тест.

Тут ми підійшли до проблем, що можуть виникнути при тестуванні. Для такої процедури рекомендується використовувати виключно автоматизовані рішення. Застосування людських ресурсів у процес перевірки вносить людський фактор, який може досить сильно впливати на якість оцінювання:

– упереджене оцінювання відповідно до стиля виконання творчої роботи;

– оцінювання робіт відносно власних критеріїв, які різняться у працівників, що у той же час є індивідуумами;

– ігнорування допущених помилок або їх фіксування навіть за їх реальної відсутності.

Але навіть при автоматизованій перевірці не все так просто. Якщо така виконана виключно у вигляду тесту, то користувач може запам'ятати всі відповіді на всі питання без формування будь-яких логічних зв'язків у теорії курсу, ба навіть взагалі без його розгляду, просто механічно проходячи тест раз за разом, окремо записуючи, які відповіді є правильними. На це можна навіть створити бота, що самостійно проходить тести, поки не отримає максимальний результат. А якщо такий буде опублікований, то кожен зможе здобути відмінну оцінку, не відкриваючи курс і навіть ні разу самотужки не проходячи тест.

На протидію цьому можна звісно підключити протиботові сервіси на кшталт reCAPTCHA, але рекомендовано вдосконалити саме систему перевірки. Така має містити завдання, рішення яких досить довго виконуватимуться звичайним перебором. Для ботів це лише питання часу, тож такий перебір варто

ускладнити так, щоб і автоматизований підбір тривав досить довго – місяці, а то й роки.

Звичайні рішення можуть пропонувати велику кількість варіантів відповідей з можливістю вибору більше одного (кількість не зазначається), але такі тести можуть потенційно нести і точки зупинки для здобувачів освіти, бо ті не розумітимуть, де саме вони помилилися і що їм варто вивчити.

Незвичайний підхід передбачає створення величезної кількості питань з уживанням різних синонімів, форм слів, стилю подання та інших видів зміни речень так, щоб ті не змінювали власного значення. Проте варто розуміти, що хоча це й ускладнює автоматизовану здачу курсу, така затія може досить ускладнити розробку тестувальної системи та навіть підвищити кількість помилок саме в ній.

7. Принципи перевірки здобувачів освіти для присвоєння освітнього ступеню

Розглянуті тренувальні перевірки можна вдосконалити до попередньо наведеного рівня, проте навіть в такому стані систему не модна використовувати для присвоєння дійсного освітнього ступеню. Так, вони можуть зменшувати імовірність автоматизованої здачі, проте достатньо посадити пройти перевірку іншу людину, що є експертом за даним курсом. У цієї проблеми є рішення, хоча кожне з наведених може мати свої проблеми і вразливості.

Найпростішим варіантом можна порівняти успішність проходження тренувальних перевірок та здобуття ступеня, особливо формуючи динаміку та швидкість навчання. Проте отримані за цим принципом результати матимуть імовірнісну характеристику, а не абсолютно точну, тож застосовувати можна несучи певні ризики із можливими виключними випадками.

Складнішим може бути аналіз активності користувача на курсі та перевірках: використання клавіатурних скорочень, траєкторії руху вказівника, швидкість реакції та інші. Проте це теж несе імовірнісну характеристику, хоча й більш точну, але ризиковану.

Може бути більш формальне рішення із використанням іншої автентифікації. Найбільш репрезентативною можна навести відеоідентифікацію, тобто під час проходження перевірки здобувач освіти буде під постійним наглядом для передбачення підміни. Можуть застосовуватися сканування відбитків пальців, обличчя, очей та інші, проте вони можуть бути переведені віртуально здобувачем освіти (далі –кандидатом) взагалі на інше робоче місце. В той же час особа, що нелегально підміняє кандидата на тестуванні (далі – експерт) може мати віддалений доступ до робочого стола кандидата, тож і відповідатиме замість нього, хоча на камері буде все ще той самий кандидат.

Звісно, перевірку можна ускладнити скануванням запущених програм на пристрої користувача і відслідковування використання технології віддаленого робочого столу. Але і на цей захист є протидія – використання віртуальної машини або контейнера [4]. В такому випадку кандидат може на своїй основній машині запустити віртуальну із копією операційної системи. Навчальну систему запустити у віртуальному середовищі і більше нічого там не відкривати. Запущені програми на основній (хостовій) машині не відображаються в списку у віртуальній машині, тож перелік усіх запущених програм виявляється частково прихованим, а тому метод хоч і може мати певний позитивний результат (виявлення порушників), він не гарантує абсолютну чесність отриманого академічного ступеню.

Виходячи із проведеного дослідження, повністю автоматизованого і водночас дистанційного рішення із перевірки знань наразі не передбачається. Проте це не значить, що реалізовувати таку систему недоцільно, адже її проведення можна здійснити аудиторно за наявності реальних спостерігачів та використання комп'ютерного обладнання, що не має зовнішніх інтерфейсів і доступу до інтернету, щоб обмежити користувача від несанкціонованих дій.

Висновки

Отже, в ході науково-дослідницького процесу в цій роботі було виявлено наявні проблеми в сучасних системах освіти: проблеми передачі інформації, проблеми з використання неефективних засобів, що не дають навчатися на повну швидкість і загальні проблеми мотивації та зацікавленості в навчанні. Були запропоновані відповідні рішення та рекомендації із уникнення проблем у навчальній системі.

На основі проведеного аналізу проблем предметної області та сформованих рішень було визначено основні принципи зі створення навчальних систем: які можуть бути види користувачів та який функціонал їм може надаватися, функціональні особливості, які покращать взаємодію із системою.

Наостанок розглянуто додаток до навчальної системи – надано варіанти створення системи перевірки знань для самоконтролю, описані проблеми, пов'язані із повністю автоматизованими дистанційними тестувальними системами, та запропоновано якомога точнішу систему перевірки із частковим відхиленням від автоматизації.

Список літератури:

[1] *R. N. Haber, M. Hershenson*. The psychology of visual perception, 2d ed. Holt: Rinehart & Winston, 1980.

[2] *A. H. MASLOW*. A Theory of Human Motivation. Brooklyn: Psychological Review, 1943, pp. 370-396.

[3] *C. Okasaki*. Purely Functional Data Structures. Ph.D thesis, School of Computer Science, Carnegie Mellon University, 1996.

[4] S. Soltesz, H. Poetzl, E. Marc, Fiuczynski, A. Bavier, L. Peterson. Container-based operating system virtualization: a scalable high-performance alternative to hypervisors. EuroSys 07: Proceedings of the 2007 conference on EuroSys, ISSN 0163-5980, pp. 275-287.

Надійшла до редколегії 09.02.2021

УДК 004.652.4 DOi 10.30837/ bi.2021.1(96).13

Т.К. Михневич[1], **О.О. Мазурова**[2]

[1]магістрант кафедри програмної інженерії ХНУРЕ,
м. Харків, tetiana.mykhnevych@nure.ua, ORCID ID 0000-0001-6377-2145

[2] кандидат технічних наук, доцент, доцент кафедри програмної інженерії ХНУРЕ,
м. Харків, oksana.mazurova@nure.ua, ORCID ID 0000-0003-3715-3476

ДОСЛІДЖЕННЯ МЕТОДІВ ПІДТРИМКИ ТЕМПОРАЛЬНОСТІ В РЕЛЯЦІЙНИХ БАЗАХ ДАНИХ

Проведено аналіз області побудови темпоральних інформаційних систем та виявлені існуючі проблеми, які потребують рішення. Запропоновано математичну модель представлення темпоральних даних з урахуванням дійсного та транзакційного часу. Розглянуто способи підтримки темпоральності у реляційних СУБД, виявлені їх переваги та недоліки. Наведено результати експериментального порівняння реалізацій підтримки темпоральності на рівні бази даних та на рівні додатку з використанням реляційних систем управління базами даних MS SQL Server та Oracle. Порівняння проводилося за метриками часу виконання різнотипних запитів та об'єму диску, що використовувався. На основі результатів дослідження методів підтримки темпоральності розроблено рекомендації стосовно організації та роботи з темпоральними моделями даних при використанні реляційних СУБД.

БІТЕМПОРАЛЬНА МОДЕЛЬ ДАНИХ, ДІЙСНИЙ ЧАС, РЕЛЯЦІЙНА СУБД, ТЕМПОРАЛЬНА МОДЕЛЬ ДАНИХ, ТЕМПОРАЛЬНІСТЬ, ТРАНЗАКЦІЙНИЙ ЧАС, ЧАСОВА МІТКА.

Михневич Т.К. , Мазурова О.А. Исследование методов поддержки темпоральности в реляционных базах данных. Проведен анализ области построения темпоральных информационных систем и выявлены существующие проблемы, требующие решения. Предложена математическая модель представления темпоральных данных с учетом действительного и транзакционного времени. Рассмотрены способы поддержки темпоральности в реляционных СУБД, выявлены их преимущества и недостатки. Приведены результаты экспериментального сравнения реализаций поддержки темпоральности на уровне базы данных и на уровне приложения с использованием реляционных систем управления базами даннях MS SQL Server и Oracle. Сравнения проводились на основании метрик времени выполнения разнообразних запросив и занятого объема диска. На основе результатов исследования методов поддержки темпоральности разработаны рекомендации по организации и работе с темпоральными моделями данных при использовании реляционных СУБД.

БИТЕМПОРАЛЬНАЯ МОДЕЛЬ ДАННЫХ, ДЕЙСТВИТЕЛЬНОЕ ВРЕМЯ, РЕЛЯЦИОННАЯ СУБД, ТЕМПОРАЛЬНАЯ МОДЕЛЬ ДАННЫХ, ТЕМПОРАЛЬНОСТЬ, ТРАНЗАКЦИОННОЕ ВРЕМЯ, ВРЕМЕННАЯ МЕТКА.

Mykhnevych T., Mazurova O. Research of Methods of Temporality Support in Relational Databases. The analysis of temporal information systems construction area is performed and the existing problems that need to be solved are identified. There was proposed a mathematical model for the presentation of temporal data, considering the valid and transactional time. Methods of supporting temporality in relational DBMS are investigated, their advantages and disadvantages are revealed. The stages of temporal database design are proposed. The results of an experimental comparison of the implementations of temporality support at the database level and at the application level using the relational database management systems MS SQL Server and Oracle are presented. Comparisons were made based on metrics of the execution time of various requests and the occupied disk space. Recommendations are developed for organizing and working with temporal data models when using relational DBMS based on an study of temporality support.

BITEMPORAL DATA MODEL, CURRENT TIME, RELATIONAL DBMS, TEMPORAL DATA MODEL, TEMPORALITY, TRANSACTION TIME, TIMESTAMP.

Вступ

На сьогодні реляційні бази даних (БД) поступово втрачають свої позиції основної концепції зберігання даних. Традиційно реляційна СУБД замінює старі значення атрибутів новими та не бере до уваги попередні стани об'єктів. Такий підхід може бути доречним не в усіх системах. Існує велика кількість предметних галузей, де зміна кожного об'єкта є критичною. У таких випадках необхідно обробляти дані, що змінюються, накопичувати історію їх зміни та видавати стан системи на будь-який момент часу. Однією із сфер, де застосовуються темпоральні моделі даних, є системи, що забезпечують підтримку вибору споживача, надаючи персоналізований перелік товарів та послуг. Вони дозволяють побудувати часову модель поведінки користувачів, що описує еволюцію вимог користувача до товарів, пропонованих системою [1]

Темпоральні (або часові) дані – це будь-які дані, що пов'язані з певними датами або проміжками часу. Для управління та зберігання таких даних краще були б пристосовані темпоральні СУБД. Але незважаючи на те, що підтримка темпоральності має свої переваги та актуальність, досі не існує єдиного стандарту реалізації темпоральних БД та майже відсутні дійсно темпоральні СУБД.

Майже всі додатки, що використовують реляційні БД (РБД), є темпоральними за своєю суттю. У них зберігаються дані, що змінюються з плином часу, але в мові запитів до РБД SQL не має адекватної і ефективної підтримки роботи саме з темпоральними даними.

Тому майже у всіх БД підтримка роботи з темпоральними даними забезпечується зусиллями програмістів. Відповідно темпоральні принципи в цих системах реалізуються по різному та не завжди ефективно, що потребує додаткового дослідження таких методів підтримки темпоральності.

1. Аналіз існуючих методів підтримки темпоральності

Зазвичай побудова БД з елементами темпоральності засновується на реляційних СУБД (РСУБД), як певна надбудова над нею [2]. Цей спосіб є найбільш простим та доступним як для розробників, так і для користувачів, адже на користь реляційних СУБД свідчать:

– успішність їх використання впродовж кількох десятиліть та надійність;

– гарна підтримка виробника, постійне вдосконалення та розвиток;

– найкраща підтримка цілісності даних;

– наявність загальноприйнятих стандартів та інше.

Але у способі використання РСУБД з підтримкою темпоральності можна виділити і наступні проблеми:

– відсутність єдиного стандарту для реалізації підтримки темпоральності; мова запитів SQL не пристосовані для роботи з темпоральними даними [2];

– РСУБД не підтримує семантику часового поля відношень, не контролює коректність його значень.

Окрім того, існує ціла низка характеристик, яким повинні задовольняти засоби підтримки темпоральності:

– специфікація періодів;

– таблиці з дійсним/транзакційним часом;

– темпоральна поведінка UPDATE / DELETE;

– темпоральні обмеження цілісності;

– предикати та функції для періодів.

Отже, можна зробити висновок, що реалізація темпоральних СУБД на базі реляційних є найбільш ефективною на сьогоднішній момент, але потребує глибокої переробки моделі даних з урахуванням темпоральної технології та особливостей обраних для реалізації РСУБД.

З іншого боку, в стандарті SQL:2011 [4] з'явилась важлива нова функціональність створювати та керувати темпоральними таблицями. Але у стандарті є і певні обмеження, а саме:

– періоди – це лише відкриті-закриті інтервали [X, Y); але це не новий тип даних, по факту це два стовпця для позначення початку та кінця періоду;

– у таблицях може бути щонайбільше один транзакційний і один дійсний період часу;

– немає понять "NOW", "UC", "infinity", "empty";

– багато відмінностей між таблицями періодів транзакційного та дійсного часу;

– обмежена підтримка формулювання темпоральних запитів.

Та навіть цей стандарт реляційні СУБД не підтримують повністю, а лише частково. При цьому темпоральні принципи в цих системах використовуються неефективно:

– неекономно використовується дисковий простір;

– низька ефективність взаємодії з темпоральними об'єктами;

– погана розширюваність БД;

– висока трудомісткість розробки БД;

– складність підтримки цілісності даних;

– складність організації доступу до даних;

– проблеми в забезпеченні конфіденційності даних.

Отже, можна виділити декілька принципових підходів до підтримки темпоральності під час використання РСУБД:

– реалізація темпоральной підтримки на рівні БД за допомогою вбудованої підтримки (вибір СУБД з максимальною кількістю темпоральних функцій);

– реалізація підтримки на рівні БД шляхом додавання додаткових стовпців, тригерів, функцій, збережених процедур тощо; фактично, розробка нової моделі надання темпоральних даних;

– реалізація підтримки на рівні додатку.

2. Постановка задачі

Метою роботи є дослідження методів підтримки темпоральності в реляційних БД шляхом їх розробки та практичної реалізації, проведення серії експериментів та формування рекомендації щодо їх ефективного використання.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– провести аналіз проблемної області побудови темпоральних інформаційних систем та методів підтримки темпоральності в реляційних БД;

– розробити нову модель та методи підтримки темпоральності в реляційних БД;

– спроектувати та реалізувати програмні рішення для проведення дослідження розроблених методів;

– спланувати та провести експериментальне дослідження розроблених методів та розробити відповідні рекомендації.

3. Математична модель надання темпоральних даних

Один з підходів, що досліджується, передбачає розробку, фактично, нової моделі надання темпоральних даних на базі реляційної моделі.

Отже, для забезпечення роботи РСУБД з темпоральними даними необхідна реалізація трьох основних концептів:

– темпоральних типів даних;

– видів часу;

– темпоральних тверджень.

В ході аналізу [5] виділено 3 темпоральні типи даних, що зберігають час:

– момент часу – точка на часовій осі (наприклад, CURRENT_DATE, 24 вересня 9 година ранку); цей тип даних існує і в звичайній моделі даних: майже кожна РСУБД має оператори роботи з ним, для його зберігання достатньо одного стовпця (див. рис. 1);

– інтервал – це довжина часу, тривалість відома, кінець і початок – ні;

– період – це протяжність часу, тривалість якого відома, початок і кінець визначаються двома моментами часу; період може бути закритим (включені обидва моменти часу), відкрито-закритим (один момент часу виключений), відкритим (обидва моменти часу виключені); саме цей тип даних створює багато труднощів у роботі з темпоральними моделями даних, необхідно підтримувати операції над множинами для часу (перетинання, об'єднання, різницю), обмеження цілісності, адекватність моделі даних [5].

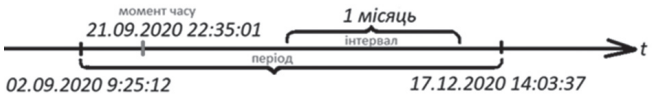


Рис. 1. Темпоральні типи даних

В ході аналізу [5] виділено 3 основних типи часу:

– час, визначений користувачем (неінтерпретований час);

– дійсний час (час, у який факт є правильним у модельованій реальності);

– транзакційний час (коли факт був записаний у БД).

В ході аналізу [5] виділено 3 базових твердження з часовим орієнтуванням:

– поточне: зараз;

– послідовне: у кожний момент часу;

– без послідовності: ігнорування часу.

Отже, три набори з трьох понять і створюють основу зберігання та управління темпоральними даними. Оскільки ці поняття є чужими для мови запитів SQL, часом виникають різні невідповідності та заміни, які кожен постачальник СУБД створив для рішення проблем темпоральності даних. Тому для реалізації темпоральних БД (ТБД) ці достатньо чіткі поняття повинні бути перетворені у громіздкі вирази та структури у SQL та схемах РБД.

Оскільки час є багатовимірним [4], то моделі даних можуть підтримувати жодного, один, два або більше вимірів:

– модель-знімок: не підтримує жодного з вимірів;

– модель з дійсним часом: підтримує лише дійсний час;

– модель з транзакційним часом: підтримує лише транзакційний час;

– бітемпоральна модель: підтримує дійсний та транзакційний час.

Мітки транзакційного часу надають інформацію про час зміни даних, виправлення помилок, а мітки дійсного – про зміну параметрів модельованого світу. Отже, дійсний і транзакційний час є ортогональними (див. рис. 2). Тому, залежно від предметної галузі та поставленої мети може бути достатньо реалізації або одного з цих типів, або двох відразу.

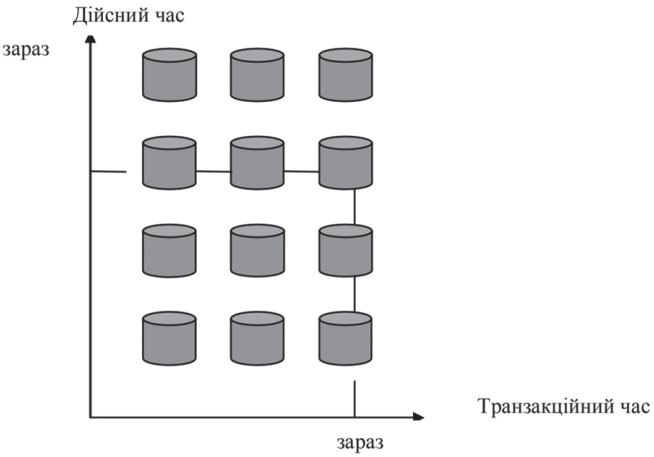


Рис. 2. Взаємовідношення між дійсним та транзакційним часом

Перейдемо до формального опису моделі темпоральних даних. На базі традиційного уявлення про модель даних [6], що складається з дозволеної організації даних, обмежень цілісності та множини операцій на об'єктами, розширена темпоральна модель (РТМ) може бути представлена, як

$$MT = (DT, OT, CT),$$

де DT – дані, OT – операції, CT – обмеження цілісності; але всі компоненти залежать від часу.

Для додання такої темпоральності була використана відкрита модель з абстрактним ідентифікатором об'єкта (Object Abstract Identify, AOID) [6], яка дозволяє будь-який об'єкт уявити у вигляді реляційного відношення. Життєвий відрізок об'єкта будемо описувати через життєві відрізки всіх його властивостей, визначених у різних відношення, що мають часові атрибути Tstart і Tend, та визначають відповідно час початку і закінчення життєвого відрізка.

Отже, об'єкт $O \in DT$ РТМ буде характеризуватися унікальним абстрактним ідентифікатором OID і набором властивостей A і може бути представлений у вигляді:

$$O = (OID, A).$$

У свою чергу набір властивостей A можна розділити на дві підмножини: множина статичних властивостей A^s, що не змінюється з часом, та множина динамічних властивостей A^d, що змінюються з часом, при цьому:

$$A = A^s \cup A^d,\ A^s \cap A^d = \varnothing.$$

Загальний вигляд РТМ представлено у вигляді ER-діаграми (див. рис. 3).

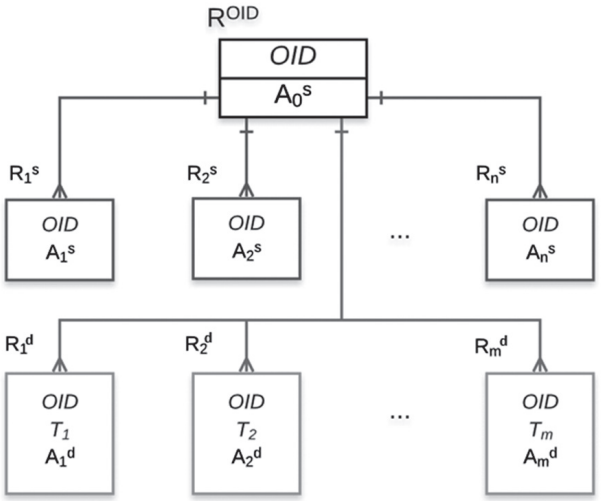


Рис. 3. ER-діаграма РТМ даних

Оскільки значення атрибутів у РБД повинні бути атомарними, то для представлення статичних та динамічних атрибутів одного реляційного відношення недостатньо. Подамо об'єкт O у вигляді сукупності взаємопов'язаних реляційних відношень:

$$O = \left(R^{OID},\ R_1^s, R_2^s, \ldots R_n^s, R_1^d, R_2^d, \ldots R_m^d\right),$$

де $R^{OID} = R^{OID}\left(OID, A_0^s\right)$ – батьківське відношення, що описує абстрактний ідентифікатор об'єкта OID та включає множину статичних атомарних атрибутів об'єкта A_0^s; отже, OID – ключ відношення R^{OID};

$R_1^s, R_2^s, \supset R_n^s$ – дочірні відношення, що описують статичні властивості A^s об'єкта;

$R_1^d, R_2^d, \supset R_m^d$ – дочірні відношення, що описують дискретно змінні у часі динамічні атрибути A^d об'єкта, та мають схеми виду:

$$R_i^d = R_i^d\left(OID, A_i^d, T_i\right),\ i = 1,2,\ldots,m,$$

де Ti – вектор атрибутів часу, що є в залежності від темпоральної форми або дійсним (VT, Valid Time), або транзакційним (TT, Transaction Time), або обома вимірами відразу, $T_i \subseteq T$, $T = \left\{t_{start}^{VT}, t_{start}^{VT}, t_{start}^{TT}, t_{end}^{TT}\right\}$ – множина атрибутів часових вимірів.

В подальшому підтримка темпоральності у моделі потребує визначення таких складових, як темпоральний ідентифікатор об'єкта OID, темпоральна алгебра OT та темпоральні обмеження цілісності CT.

Темпоральний ідентифікатор об'єкта OID може використовуватися в якості як первинного, так і зовнішнього ключів. В якості первинного ключа цей ідентифікатор однозначно може визначити стан будь-якого об'єкту у будь-який момент часу. У темпоральних моделях даних первинний та зовнішній ключі залежать від часу, адже час входить до їх складу. Додаткові обмеження цілісності CT, що накладає темпоральна модель даних, можуть бути реалізовані у РСУБД за допомогою вбудованих можливостей, таких як тригери, функції, збережені процедури.

Операції з темпоральними даними OT можуть бути реалізовані за допомогою реляційних операцій та забезпечуються у тій чи інший мірі вбудованою підтримкою РСУБД операцій з темпоральними даними (темпоральні предикати). Множина операцій має більше розбіжностей з точки зору реалізації у конкретній РСУБД, ніж, наприклад, структура даних.

Проектування запропонованої РТМ буде складатися з наступних етапів:

– концептуальне проектування з використанням моделі ER (часові аспекти ігноруються, основна задача – фіксація поточної реальності);

– логічне проектування з використанням реляційної моделі (нечасова схема ER відображається у нечасову реляційну схему, набір таблиць);

– фізичне проектування для забезпечення адекватних показників (застосовуються часові анотації, модифікуючи логічну схему). Алгоритм переходу до фізичної схеми – це вже тема окремої статті.

4. Планування експериментального дослідження

Для максимального занурення у принципи темпоральності прийнято рішення досліджувати саме бітемпоральну модель даних, щоб покрити і дійсний, і транзакційний час. Під час аналізу було виявлено, що в жодній РСУБД неможливо створити повноцінну бітемпоральну підтримку за допомогою лише вбудованої підтримки (але для окремих випадків достатньо і вбудованої), тому буде досліджуватися реалізація підтримки на рівні БД (шляхом додавання додаткових стовпців, тригерів і т.п.) (надалі, підхід «на рівні БД») та на рівні додатку (далі, підхід «на рівні додатку»).

З точки зору організації структур даних підходи не відрізняються, різниця лише у реалізації темпоральних операцій та обмежень. «На рівні БД» вони реалізуються на сервері у вигляді тригерів, функцій, збережених процедур. «На рівні додатку» – у вигляді класів, методів, тощо, залежно від обраної технології.

Тому за основу організації структур даних для обох підходів взята розроблена РТМ.

На базі РТМ для предметної області медицини була спроектована схема БД з підтримкою темпоральності (див. рис. 4). На рисунку відображений лише дійсний час (щоб забезпечити прозорість); для підтримки транзакційного часу використовуються таблиці-копії з постфіксом History усіх

темпоральних таблиць з двома додатковими стовпцями SysStartTime, SysEndTime (атрибути $t_{start}^{TT}, t_{end}^{TT}$ з математичної моделі).

Для експерименту БД була наповнена різними об'ємами даних. Дані для таблиць Drug, Disease, Procedure взяті із ресурсу https://www.cms.gov/ (вебсайт федерального уряду, керований американськими центрами Medicare & Medicaid Services), для інших таблиць дані згенеровано у різному обсязі за допомогою скриптів, що заповнюють таблиці випадковими даними.

Для обох підходів були реалізовані операції вставки, редагування, видалення та вибірки даних з урахуванням темпоральної моделі.

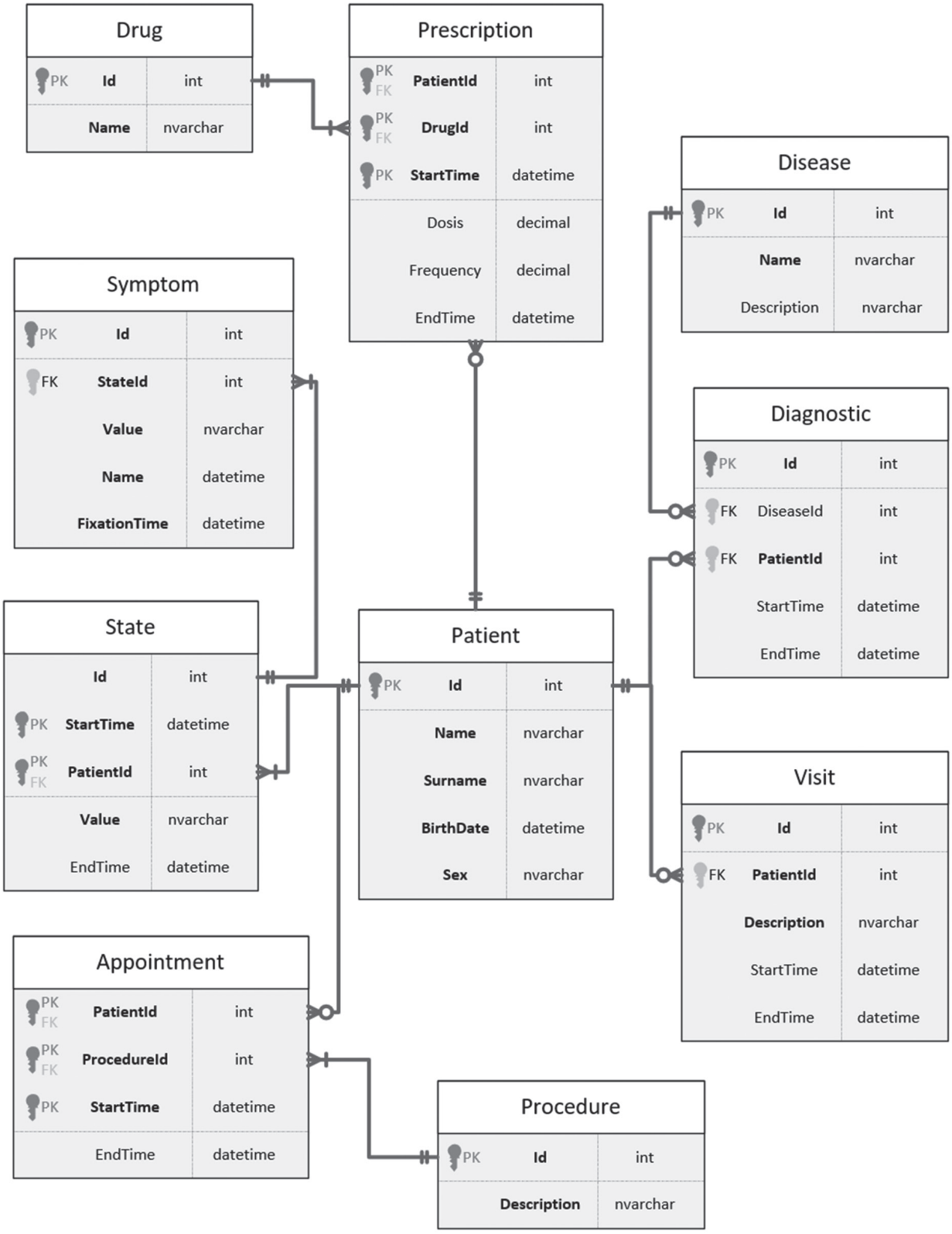


Рис. 4. Фізична схема БД з дійсним часом

Кожна вставка у бітемпоральну таблицю займає одну операцію з проставленням значень у двох колонках – дійсного часу, з якого факт починає бути істинним (атрибут t^{VT}_{start}), та транзакційного часу, у який цей факт був записаний у БД (атрибут t^{TT}_{start}). Факту, що змінюється з часом, відповідають атрибути $R^d_1, R^d_2, \supset R^d_m$ з моделі.

Кожне оновлення запису у бітемпоральній таблиці супроводжується поновленням попереднього рядка, який був істинним до операції UPDATE (проставлення дати закінчення істинності факту), та вставкою нового з проставленням дати початку істинності нового факту.

Кожне видалення з бітемпоральної таблиці полягає в проставленні дати закінчення істинності факту.

Для будь-якої операції, що змінює стан БД, перевіряються темпоральні обмеження цілісності.

Будь-яка вибірка з БД повертає дійсний стан бази даних на будь-який заданий момент часу.

Оскільки вбудованих функцій роботи з періодами немає, їх було реалізовано самостійно на рівні БД та додатку. Це такі часто використовувані функції у темпоральній логіці, як перетин періодів, включення моменту часу в період та інші. Вони є досить універсальними та не прив'язаними до конкретної предметної області, тому їх можна використовувати повторно.

Було прийнято рішення розширити експеримент та виконати його не тільки для двох обраних підходів, але й на двох СУБД, щоб отримати більш розширені висновки. Під час вибору РСУБД для проведення дослідження враховувались такі фактори, як рівень вбудованої підтримки (періоди, темпоральні предикати, операції), популярність СУБД, кількість існуючих досліджень щодо темпоральності. У результаті були обрані РСУБД Oracle та MS SQL Server.

Отже, для експериментів з MS SQL Server:

– для підходу «на рівні БД» бітемпоральну підтримку реалізовано самостійно з використанням тригерів INSTEAD OF INSERT / UPDATE / DELETE та перевіркою обмежень у них; рішення використання вбудованої в MS SQL Server підтримки транзакційного часу (відстеження зміни у таблиці, перенос недійсні записи у таблицю StateHistory) було відхилено, тому що для таких таблиць не дозволяється створювати тригери;

– для підходу «на рівні додатку» використано вбудовані можливості MS SQL Server щодо підтримки транзакційного часу, адже тригери вже непотрібні, усі перевірки цілісності йдуть на рівні додатку.

Для Oracle були прийняті наступні рішення:

– для підходу «на рівні БД» бітемпоральна підтримка реалізована самостійно з використанням тригерів: тригери BEFORE створено для перевірки обмежень, а тригери AFTER – для перенесення неактуальних даних в таблицю History; для підтримки періодів використано інструкцію *period for* при створенні таблиць з періодами (це вбудована підтримка обмежень цілісності щодо перетину періодів);

– для підходу «на рівні додатку» усі перевірки цілісності реалізовано на рівні додатку.

У ході експерименту замірялися такі характеристики, як час для SELECT/INSERT/UPDATE/DELETE операцій у темпоральних таблицях з різною кількістю записів та об'єм зайнятого дискового простору.

5. Результати проведених експериментів

Отже, під час дослідження були проведені наступні серії експериментів:

1) запити типу

SELECT/INSERT/UPDATE/DELETE

для MS SQL Server з підтримкою темпоральності на рівні БД;

2) запити типу

SELECT/INSERT/UPDATE/DELETE

для Oracle з підтримкою темпоральності на рівні БД;

3) запити типу

SELECT/INSERT/UPDATE/DELETE

для MS SQL Server з підтримкою темпоральності на рівні додатку;

4) запити типу

SELECT/INSERT/UPDATE/DELETE

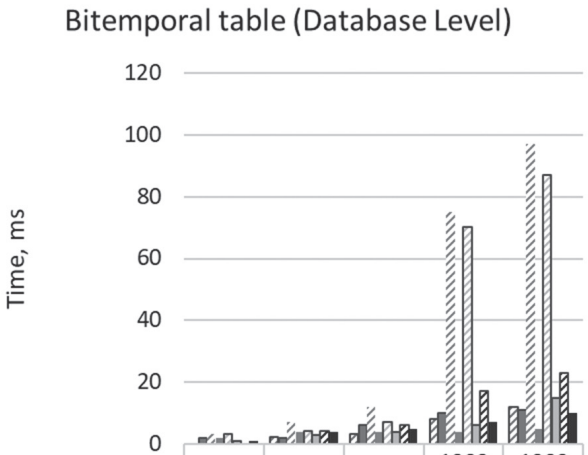
для Oracle з підтримкою темпоральності на рівні додатку.

Кожна серія експериментів проводилася на таблицях з 0, 100, 1000, 10000 та 100000 записами.

Результати експерименту перших двох серій з підтримкою на рівні БД зображені на рис. 5. Під графіком надано чисельний еквівалент у мілісекундах щодо кожного виду запиту та легенда кольорів.

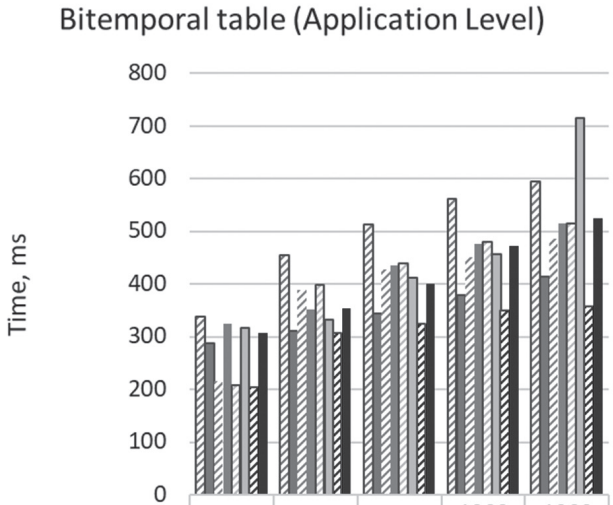
За графіками видно, що MS SQL суттєво програє Oracle на великих обсягах БД. Такі результати замовлено тим, що для перевірки цілісності періодів Oracle має оператор period for при створенні таблиць, тому у тригерах менша кількість перевірок, у MS SQL всі перевірки реалізуються самостійно за допомогою тригерів, тому INSERT / UPDATE / DELETE запити займають більше часу.

Результати серій експериментів з підтримкою на рівні додатку наведено на рис. 6. Як бачимо, є залежність від досліджуваних операцій та однозначного переможця немає. Звісно, що вся робота на рівні додатку збільшує час виконання запитів. Але для MS SQL Server ми використовуємо вбудовану підтримку транзакційного часу, завдяки чому INSERT/UPDATE/ DELETE операції виконуються швидше, ніж в Oracle. Бо в Oracle реалізуємо підтримку транзакційного часу на рівні додатку, що є повільнішим.



	0	100	1000	10000	100000
select ms sql	0	2	3	8	12
select oracle	2	2	6	10	11
insert ms sql	3	7	12	75	97
insert oracle	2	4	4	4	5
update ms sql	3	4	7	70	87
update oracle	1	3	4	6	15
delete ms sql	0	4	6	17	23
delete oracle	1	4	5	7	10

Рис. 5. Результати експериментів з Oracle та MS SQL Server для підходу «на рівні БД»



	0	100	1000	10000	100000
select ms sql	337	454	512	561	594
select oracle	287	312	345	379	414
insert ms sql	215	389	427	451	486
insert oracle	324	351	435	476	516
update ms sql	207	397	438	479	514
update oracle	316	332	412	457	715
delete ms sql	203	307	324	349	357
delete oracle	308	354	401	472	524

Рис. 6. Результати експериментів з Oracle та MS SQL Server для підходу «на рівні додатку»

Результати експериментів також було проаналізовано на базі параметру об'єму диску, що займає БД. І для MS SQL Server, і для Oracle кількість займаного місця з точки зору кількості збережених рядків однакова. Однак, таблиці з підтримкою темпоральності займають у рази більше місця, ніж звичайні, тому актуальним стає питання вакуумування. Вакуумування, що зменшує розмір History-таблиць шляхом видалення менш бажаних рядків, тим самим збільшуючи простір і ефективність роботи з таблицею, в темпоральних БД необхідно проводити обов'язково, Оскільки вбудованого функціоналу для цього немає ні в MS SQL Server, ні в Oracle, то воно реалізується самостійно.

6. Рекомендації щодо вибору підходів підтримки темпоральності у реляційних СУБД

Провівши дослідження, виконавши заміри та проаналізувавши результати, можна зазначити наступні рекомендації:

– якщо створюється система, для якої важлива підтримка лише транзакційного часу (ведення логжурналу операцій), то краще обрати у MS SQL Server з вбудованою підтримкою транзакційного; тобто використовувати підхід з підтримкою на рівні БД;

– якщо крім підтримки транзакційного часу необхідні темпоральні обмеження цілісності (не лише логування, але й елементи темпоральної логіки), то можна обрати один із трьох варіантів: реалізація перевірок обмежень цілісності на рівні додатку, або реалізація обмежень цілісності на рівні БД за допомогою збережених процедур або функцій, які треба викликати з додатка перед зміною БД, або реалізація підтримки транзакційного часу «на рівні БД»;

– якщо важлива підтримка лише дійсного часу, то краще обрати Oracle, адже Oracle має інструкцію *period for*, яка реалізує темпоральну перевірку цілісності;

– якщо треба реалізовувати підтримку і транзакційного, і дійсного часу у повній мірі, то її треба реалізовувати повністю самостійно на рівні БД що в Oracle, що в MS SQL Server, тому що:

а) у MS SQL Server неможливо створити тригери на таблицях з вбудованою підтримкою транзакційного часу, а це означає, що неможливо реалізувати перевірку темпоральної цілісності на рівні БД, що є дуже важливим при роботі з темпоральними моделями даних. Вбудованої підтримки дійсного часу зовсім немає;

б) в Oracle немає підтримки транзакційного часу, а підтримка дійсного часу теж не повноцінна;

– якщо прийнято рішення реалізувати повноцінну бітемпоральну підтримку самостійно у будь-якій СУБД, слід прийняти рішення залежно від архітектури створюваної системи, чи можлива сильна

прив'язка системи до певної СУБД, або потрібна можливість легкої зміни СУБД. Це треба вирішити на початковому етапі, тому що логіка на рівні БД повинна буде повністю переписана при використанні іншої СУБД. Якщо ж підтримка темпоральності буде реалізована на рівні додатку, то перейти на іншу СУБД буде набагато легше, але швидкість роботи буде меншою;

– у разі реалізації підтримки бітемпоральної моделі на рівні БД слід дотримуватися таких рекомендацій:

а) проектування БД здійснюється поетапно, як описується у пункті 4;

б) описується набір правил та обмежень, які повинні бути враховані у темпоральній логіці проектованої системи;

в) формується бібліотека функцій або збережених процедур, які реалізують загальну темпоральну логіку (працюють з періодами, перевіряють обмеження тощо). Надалі їх можна використовувати у тригерах для перевірки цілісності та навіть у інших системах;

г) поступово реалізується підтримка транзакційного і дійсного часу, потім додається перевірка обмежень темпоральної цілісності згідно з бізнес правилами та особливостями модельованої предметної області;

д) обирається та реалізується стратегія вакуумування на History-таблиці.

7. Висновки та перспективи

В роботі були досліджені основні аспекти підтримки темпоральності в РСУБД. Було запропоновано математичну модель представлення темпоральних даних з урахуванням дійсного та транзакційного часу.

Під час реалізації темпоральної підтримки були реалізовані загальні функції у СУБД Oracle та MS SQL Server, що можуть бути використані при роботі з будь-якими темпоральними моделями даних.

Результати експериментального дослідження дозволили сформулювати рекомендації щодо вибору підходів до роботи з темпоральними моделями даних у РСУБД. Отримані рекомендації можуть бути використані при проектуванні систем з підтримкою темпоральності.

Список літератури:

[1] Chalyi S., Leshchynskyi V. Temporal modeling of user preferences in recommender system // CEUR Workshop Proceedings. 2020. 2711. P. 518-528.

[2] Порай Д. С., Соловйов А. В., Корольков Г. В. Реализация концепции темпоральной базы данных средствами реляционной СУБД, – Едиториал, 2013. — с. 92-109.

[3] Григорович В.Г. Огляд технологій темпоральних баз даних / В.Г. Григорович, О.Ю. Косовська, О.М. Пігур-Пастернак, А.Ю. Шілінг // Вісник Національного університету «Львівська політехніка». – 2011.

[4] Krishna Kulkarni, Jan-Eike Michels. Temporal features in SQL:2011, — ACM SIGMOD Record, 2012, 34 p.

[5] Snodgrass R. Developing Time-Oriented Database Applications in SQL/Snodgrass R. — Morgan Kaufmann Publishers, 2000. — 504p.

[6] Date C.J., Hugh Darwen, Nikos A. Lorentzos. Temporal Data and the Relational Model: A Detailed Investigation into the Application of Interval and, Relation Theory to the Problem of Temporal Database Management/ Date C.J., Hugh Darwen, Nikos A. Lorentzos. – Morgan Kaufmann, December, 2002. – 480p.

Надійшла до редколегії 26.03.2021

УДК 004.93: 004.94 DOi 10.30837/ bi.2021.1(96).14

Ілона Ревенчук[1], Єгор Агарков[2]

[1]ХНУРЕ, м. Харків, Україна, ilona.revenchuk@nure.ua, ORCID: 0000-0002-5188-9538

[2]ХНУРЕ, м. Харків, Україна, yehor.aharkov@nure.ua

МОДЕЛЮВАННЯ ДОПОВНЕНОЇ РЕАЛЬНОСТІ НА ОСНОВІ МАРКЕРІВ

Доповнена реальність — це велика кількість можливостей для комерційного застосування, нові горизонти в освіті, промисловості, медицині, будівництві, торгівлі, навчанні і навіть туризмі. Комерційне зростання використання технологій доповненої реальності дуже стрімке. Їй, на відміну від віртуальної реальності, необов'язково спиратися на спеціалізоване залізо і громіздкі пристрої. Технологія працює на мобільних пристроях – смартфонах, планшетах. Саме взаємодія обчислювальних пристроїв з зображенням реального світу відрізняє доповнену реальність від віртуальної. Доповнена реальність має потенціал зробити відносини людини з інформацією більш ергономічними. Дані будуть автоматично доставлятися користувачам у необхідному контексті для різних ситуацій в повсякденному житті, таким чином, технологія підніме взаємодію людини з інформацією на принципово інший рівень. При моделюванні доповненої реальності особливу увагу приділяють методам та алгоритмам моделювання віртуального простору.

ДОПОВНЕННА РЕАЛЬНІСТЬ, МАРКЕРИ, АФІННІ ПЕРЕТВОРЕННЯ, ВІРТУАЛЬНА ЛАБОРАТОРІЯ.

Ревенчук И.А., Агарков Е.С. Моделирование дополненной реальности на основе маркеров. Дополненная реальность — это большое количество возможностей для коммерческого применения, новые горизонты в образовании, промышленности, медицине, строительстве, торговле, учебе и даже туризме. Коммерческий, стремительный рост использования технологий дополненной реальности позволяет, ей в отличие от виртуальной реальности, не использовать специализированное железо и громоздкие устройства. Технология работает на мобильных устройствах — смартфонах, планшетах. Именно взаимодействие вычислительных устройств с изображением реального мира отличает дополненную реальность от виртуальной. Дополненная реальность имеет потенциал сделать отношения человека с информацией более эргономичными. Данные будут автоматически доставляться пользователям в необходимом контексте для различных ситуаций в повседневной жизни, таким образом, технология поднимет взаимодействие человека с информацией на принципиально иной уровень. При моделировании дополненной реальности особое внимание уделяется методам и алгоритмам моделирования виртуального пространства.

ДОПОЛНЕНИЕ РЕАЛЬНОСТЬ, МАРКЕРЫ, АФФИННОЕ ПРЕОБРАЗОВАНИЕ, ВИРТУАЛЬНАЯ ЛАБОРАТОРИЯ.

Revenchuk I., Agarkov E. Marker-based Augmented Reality Modeling. Augmented reality is a lot of opportunities for commercial use, new implementation in education, industry, medicine, construction, trade, education and even tourism. The commercial, rapid growth in the use of augmented reality technologies allows it, unlike virtual reality, not to use specialized hardware and bulky devices. The technology works on mobile devices — smartphones, tablets. The differnce between augmented reality and virtual reality is interaction of computing devices with the image of the real world. Augmented reality has the potential to make human relationships with information more ergonomic. Data will be automatically delivered to users in the necessary context for various situations in everyday life, thus, the technology will raise the interaction of a person with information to a fundamentally different level. Methods and algorithms for modeling virtual space is an important in modeling augmented reality.

AUGMENTED REALITY, MARKERS, AFFINITY CONVERSION, VIRTUAL LABORATORY

Вступ

Однією з головних тенденцій, що виділяються експертами на IT-ринку, є розвиток технологій доповненої і віртуальної реальності, що стимулюють зростання популярності мобільних пристроїв, особливо в сегменті смартфонів. Технології доповненої і віртуальної реальності, які нещодавно фігурували тільки у фантастичній літературі, вже зараз представляють компаніям можливість для перетворення робочих процесів, зміни підходу до роботи зі своїми клієнтами. Доповнена реальність (Augmented reality, AR) — це технологія, яка дозволяє накладати інформацію в формі тексту, графіки, аудіо та інших віртуальних об'єктів на реальні об'єкти в режимі реального часу.

Визначення доповненої реальності було введено недавно. Спочатку цей термін був запропонований дослідником Томом Коделом в 1990 році, який в той час працював в компанії Boeing. У 1997 році, Рональд Т. Азума в своєму дослідженні, присвяченому різним способам використання доповненої реальності, дав їй досить просте визначення — це система, яка:

– поєднує віртуальне і реальне;
– взаємодіє в реальному часі;
– розташовується в тривимірному просторі.

Доповнена реальність, за Азумом, є різновидом віртуальної реальності, але з одним застереженням: доповнена реальність інтегрується і доповнює справжній світ замість того, щоб повністю його замінити, як це робить віртуальна реальність.

Багато аналітичних компаній прогнозують зростання популярності технологій доповненої і віртуальної реальності в сучасній культурі. Дані прогнози

цілком виправдані, тому що вже зараз лідери серед IT-компаній готові вкладати величезні кошти в розвиток доповненої і віртуальної реальності. Підвищення інтересу до технологій доповненої і віртуальної реальності спостерігається також серед споживачів. Даний факт можна простежити по динаміці пошукових запитів за допомогою сервісу Google Trends.

Зростання популярності даних технологій пов'язаний з ривком в їх розвитку, які досягли такого рівня, коли уявлення про данні технології стали відповідати очікуванням. Зокрема, Tom's Hardware, широко відоме інтернет-видання, присвячене комп'ютерним технологіям, опублікувало статтю, у котрій був зроблений огляд найбільш явних переваг доповненої і віртуальної реальності на даний момент, в порівнянні з минулими розробками [1].

1. Основні теоретичні відомості моделювання віртуального простору

Віртуальний простір моделюється з використанням афінних перетворень [1, 2], це коли вершина для відображення її на екрані перетворюється з тривимірних координат в двомірні. Вхідними даними є координати, а вихідними — зображення. При побудові доповненої реальності — накладання віртуальних об'єктів на зображення, що фіксується веб-камерою [3-5], — необхідно вирішити інше завдання, де вхідними даними буде зображення, а вихідними — координати. Подібні перетворення використовуються в фотограмметрії для визначення розмірів об'єктів. Основна проблема при побудові доповненої реальності полягає у визначенні відносних систем координат сцени (зображення) і відеокамери, тобто в локалізації веб-камери. Це завдання складається ще з декількох завдань: скласти нерухому систему координат зі сценою; вибрати еквівалент віртуальної камери і встановити її положення, при цьому положення камери визначається її обертанням і зміщенням відносно центру координат, так щоб проекції зображень, що фіксуються веб-камерою і віртуальною камерою, збігалися.

Як відомо [5], віртуальну камеру можна задати двома 4 × 4-матрицями:

$$A = \begin{bmatrix} a & 0 & f & 0 \\ 0 & b & g & 0 \\ 0 & 0 & c & d \\ 0 & 0 & e & 0 \end{bmatrix}. \qquad (2)$$

$$B = \begin{bmatrix} RY_x & RY_y & RY_z & T_x \\ RP_x & RP_y & RP_z & T_y \\ RR_x & RR_y & RP_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (2)$$

Коефіцієнти a, b, c, d, e, f, g матриці A (1) утворюють фрустум [1] — пірамідальну видиму область, обмежену шістьма площинами, які будуються виходячи зі значень кута зору камери, відносини висоти віртуального вікна до його ширини і значення глибини Z-буфера.

Матриця B (2) складається з 3 × 3-матриці R обертання навколо трьох осей (RY – yaw, вісь спрямована вгору; RP – pitch, вісь спрямована вправо; RR – roll, вісь напрямки) і вектора зміщення T уздовж осей x, y, z відповідно.

Проекція вершини на ближню площину екрану будується відповідно до формули

$$\begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} a & 0 & f & 0 \\ 0 & b & g & 0 \\ 0 & 0 & c & d \\ 0 & 0 & e & 0 \end{bmatrix} * \begin{bmatrix} RY_x & RY_y & RY_z & T_x \\ RP_x & RP_y & RP_z & T_y \\ RR_x & RR_y & RP_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} V_x \\ V_y \\ V_z \\ 1 \end{bmatrix} \qquad (3)$$

яка перетворює тривимірний вектор $\left[V_x, V_y, V_z, 1\right]^T$ іртуального простору в двовимірний вектор $\left[S_x, S_y\right]^T$ кранного простору. При вирішенні задачі локалізації камери 3D-вектори і 2D-вектори у формулі (3) відомі, а матриці A і B, що складаються із зовнішніх і внутрішніх параметрів веб-камери, невідомі, ці параметри і необхідно знайти.

Матриця B містить зовнішні параметри (Extrinsic Parameters) камери: кут її повороту й зміщення відносно центру сцени. Завдання знаходження матриці B зводиться до порівняння проекцій, що фіксуються реальною і віртуальною камерами [6, 7]. Як тільки матриця B стає відомою, завдання локалізації камери можна вважати вирішеним.

2. Проблеми реалізації моделі доповненої реальності

Технічні проблеми реалізації пов'язані не тільки з продуктивністю комп'ютера, на якому виконується програма, а й з технічними показниками використовуваної камери, такими як розмір одержуваного зображення і його якість. Мінімальні вимоги до обчислювальної системи невисокі, програма може бути реалізована на мобільних комп'ютерах або навіть на сучасних смартфонах після відповідної компіляції для обраної системи. Стосовно вимог до камери, слід враховувати, що чим більше і чіткіше область фіксованого зображення, тим вище точність розрахунків. Крім якості зображення, необхідно щоб швидкість його фіксування була не менше 30 кадрів / с, так як від цього залежить максимальна швидкість роботи програми в цілому.

Деякі недоліки камери можна компенсувати, на відміну від розміру отримуваного зображення і швидкості роботи використовуваної камери, які змінити неможливо. Наприклад, такий недолік, як дисторсія (рис. 1), виникає в оптичних системах, де використовуються опуклі лінзи, можна компенсувати перерахунком координат зображення.

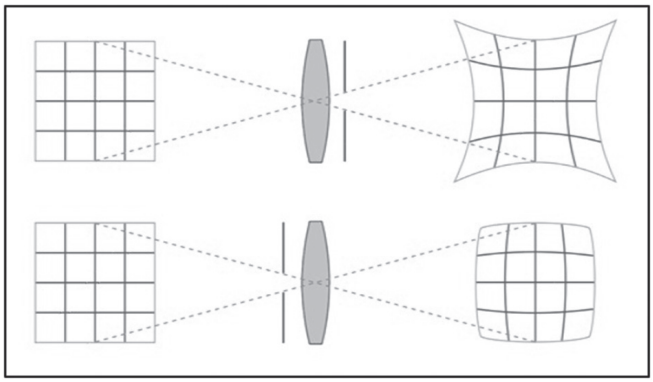


Рис. 1. Схематичне зображення дисторсії

Для цього необхідно знайти координати центру дисторсії і її значення; ці дані, називаються внутрішніми параметрами камери (Intrinsic Parameters), визначав ставлення ідеальних і реальних параметрів веб-камери і віртуальної камери:

$$\begin{cases} d^2 = (x_i - x_0)^2 + (y_i - y_0)^2; \\ p = (1 - fd^2); \\ x_d = p(x_i - x_0) + x_0, y_d = p(y_i - y_0) + y_0, \end{cases} \quad (4)$$

тут f значення дисторсії; (x_d, y_d) координати центру дисторсії; $(x_0, y_0), (x_i, x_i)$ оординати точок зображення, що фіксується веб-камерою. При перекладі зображення з реального в ідеальне необхідно врахувати, що внаслідок компенсації дисторсії зображення виходить не прямокутним (рис. 2), і по його краях з'являються незаповнені області (які можуть негативно впливати на знаходження маркера).

Щоб усунути ці області, необхідно змінити масштаб зображення, використовуючи коефіцієнт масштабу (s). Тоді система рівнянь (4) перетворюється до наступного вигляду:

$$\begin{cases} x = s(x_i - x_0),\ y = s(y_i - y_0); \\ d^2 = x^2 + y^2; \\ p = (1 - fd^2); \\ x_d = px + x_0,\ y_d = py + y_0. \end{cases} \quad (5)$$

Рис. 2. Схематичне зображення результату компенсації дисторсії

Внутрішні параметри камери (або калібрування) визначаються один раз і використовуються як константа, тоді як зовнішні параметри змінюються при кожному кадрі в залежності від положення і орієнтації камери.

3. Алгоритмічні проблеми

Головна проблема при моделюванні доповненої реальності полягає в отриманні невідомої інформації про простір. Проблему відсутності інформації можна вирішити різними способами, наприклад додаванням об'єкта, параметри якого заздалегідь відомі. Зіставлення фіксуємі веб-камерою і віртуальною камерою зображень найкраще робити за спеціальними областям, відповідність між якими відомо, такі області називаються «feature» — особливість, риса, особлива точка. Як правило, це області найбільшого перепаду освітленості, кути об'єктів або спеціально приготовлені маркери. Знаходження відповідностей за маркерами дає деякі переваги, наприклад: відомо кількість особливих точок і орієнтація маркера, також відомо, що всі особливі точки компланарні (в ідеалі). Маркер зазвичай складається з чотирьох особливих точок, які розташовані в яскраво виражених кутах (наприклад, кути білого квадрата, зображеного на чорному квадраті більшого розміру).

Процес зображення для автоматичного виявлення маркера зображено на рис. 3 [8], де:

– (а) — початкове зображення,

– (б) — результат застосування локального порогового значення,

– (в) — виявлення контуру,

– (г) — полігональна апроксимація і видалення непотрібних контурів,

– (д) — приклад маркера після перетворення перспективи,

– (є) — призначення бітів для кожного осередку.

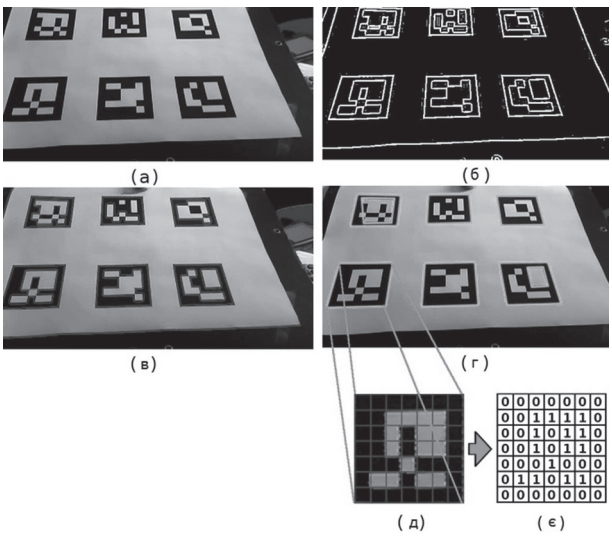


Рис. 3. Процес зображення для автоматичного виявлення маркера

Для розрахунку системи просторових координат сцени по довільному набору точок слід розрахунки проводити в динамічній системі (а не в статиці) з використанням алгоритму розпізнавання особливих точок, де аналіз ряду зображень і характеру руху вебкамери дасть відсутню інформацію про сцену.

4. Реалізація доповненої реальності на основі маркерів

Ключовою особливістю AR у порівнянні з іншими інструментами обробки зображень є те, що віртуальні об'єкти переміщуються та обертаються у 3D-координатах замість 2D-координат зображення.

Основними цілями AR є аналіз змін у захоплених кадрах камери та правильне вирівнювання віртуальних даних у сцені камери на основі результатів відстеження. У свою чергу, підхід, заснований на маркерах, забезпечує точне відстеження за допомогою візуальних маркерів, наприклад, двійкових маркерів (ARUCO, METAIO тощо) або з фотографією реальних площинних об'єктів у кадрі камери. Спрощена схема AR системи представлена на рис. 4.

Давайте детально розглянемо блок-схему AR-системи.

Спочатку нам потрібно мати маркерне зображення та витягти послідовні кадри камери. Модуль відстеження на блок-схемі (рис. 4) є ядром системи доповненої реальності. Він обчислює відносну позицію (pose) камери на основі правильно виявленого та розпізнаного маркера в сцені.

Рис. 4. Спрощена схема AR системи

Термін «pose» означає позицію шести ступенів свободи (DOF), тобто 3D-розташування та 3D-орієнтацію об'єкта. Модуль відстеження дозволяє системі додавати віртуальні компоненти як частину реальної сцени. Оскільки ми маємо справу з кадрами камери в 2D-системі координат, необхідно використовувати проективну геометрію для віртуального збільшення об'єктів 3D.

Виявлення та розпізнавання. У разі відстеження за допомогою двійкового маркера першим необхідним є надрукувати потрібний маркер і поставити його перед камерою. Ця вимога є очевидним недоліком алгоритму відстеження.

Алгоритм виявлення дуже простий і базується на природі маркера:

– застосування адаптивного порогового значення для вилучення країв;

– вилучення замкнених контурів з двійкового зображення;

– фільтрація контурів;

– наближення контурів та виявлення контурів чотиригранної форми.

Після вищевказаних кроків кандидати на маркери зберігаються для подальшого розпізнавання маркера.

Кожен кандидат перекошений у фронтальний вид і розділений на блоки. Завдання алгоритму розпізнавання — витягти двійковий код у кандидата на маркер та порівняти його з кодом справжнього маркера. Найбільш схожий кандидат розглядається як відповідний маркер.

На рис. 5 представлений приклад сцени та те, як здійснюється виявлення та розпізнавання двійкового маркера.

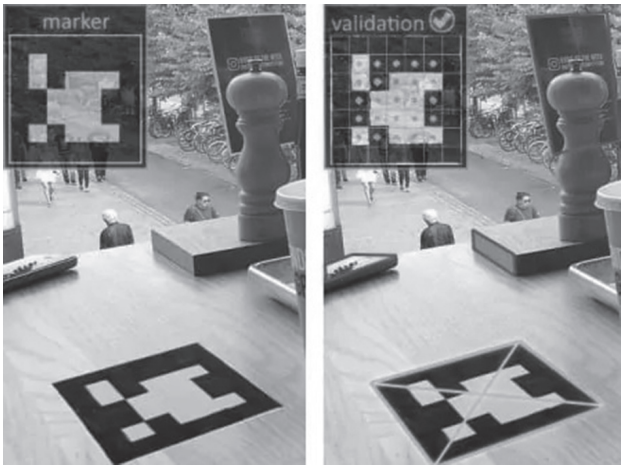


Рис. 5. Сцена з двійковим маркером (ліворуч) та виявленим маркером (праворуч)

Досконаліший алгоритм відстеження за допомогою фотомаркера дозволяє позбутися розміщення синтетичних двійкових маркерів на сцені. Досить просто сфотографувати площинний об'єкт у реальній сцені та використовувати його як маркер.

Методи, засновані на локальних особливостях, є найбільш загальними для цього завдання. Хорошими кандидатами для таких завдань є надійні дескриптори SURF [9] або один із двійкових дескрипторів: ORB [10], FREAK [11], BRIEF [12], BRISK [13] або

LATCH [14]. Зіставлення локальних дескрипторів, як правило, здійснюється за допомогою звичайного збігу Brute Force або з більш ефективним алгоритмом FLANN. Як результат, після узгодження даних може бути здійснено збільшення даних. Схема такого рівня наведена на рис. 6.

Рис. 6. Алгоритм відстеження на основі зображень

Цей спосіб також має деякі недоліки. Це дійсно ресурсомістка велика кількість обчислень на етапах виявлення ознак, обчислення дескрипторів та відповідності ознак.

Приклад збільшення віртуального об'єкта за допомогою реального плоского маркера в сцені представлений на рис. 7

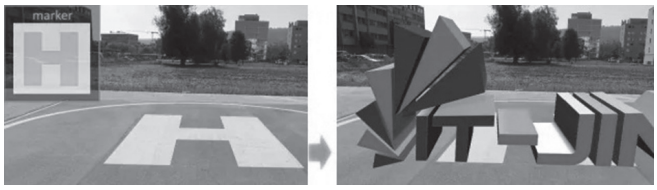


Рис.7. Приклад збільшення віртуального об'єкта за допомогою реального плоского маркера

Практична реалізація доповненої реальності при проектуванні віртуальних лабораторій

Віртуальні та цифрові лабораторії є дуже цінним середовищем для залучення учнів до науки. Це особливо важливо для отримання лабораторного досвіду майбутніми інженерами в галузі науки, техніки, техніки та математики [15]. Віртуальна лабораторія містить віртуальні лабораторні роботи (VLW). У більшості випадків VLW — це спеціальне програмне забезпечення, яке дозволяє користувачам експериментувати з віртуальними пристроями.

Інтерактивний дизайн віртуальних пристроїв був описаний у VLW, що передбачає математичне моделювання, можна вважати віртуальним симулятором. Машинне забезпечення педагогічного процесу дає можливості для самостійного досвіду тих, хто навчається, та їх виконання в класах та лабораторіях [16, 17].

Віртуальна лабораторія електронної мікроскопії для вимірювань представлена на рис. 8.

Рис. 8. Вид віртуальної лабораторії

Студент також має можливість не лише взаємодіяти з пристроями, а й пересуватися по віртуальній лабораторії. Панелі інструментів та кнопки виділяються, коли учні, які наводять курсор на них, наводять курсор, що забезпечує інтерактивну поведінку програми. Основним двигуном Unity 3D був обраний для розвитку віртуальної лабораторії [18]. Він досить доступний для того, щоб стати чудовим місцем для початку розробки вмісту VR, але це професійний механізм, що використовується великою частиною студій VR, який може досягти багатьох найвищих графічних ефектів. Веб-сайт Unity дозволяє завантажувати єдність (особисте видання безкоштовне). У цій лабораторній роботі студент має справу з установкою віртуального електронного мікроскопа. Всі прилади, що працюють у віртуальній лабораторії, за дизайном схожі на реальне обладнання. Перед виконанням лабораторних робіт необхідно передати контрольний список на віртуальному планшеті. Панель управління відеооператора показана на рис. 9 [19-20].

Зображення поверхні, отримане електронним мікроскопом, відображається на відеоекрані. Подібно до реального пристрою, віртуальний відеооператор може працювати із зображенням.

Керуючись завданням, студент робить необхідні процедури із зображенням. Здійснено інтерактивну зміну масштабу, яскравості та контрастності зразкового зображення. У VLW використовуються зображення реальних поверхонь зразків. Процес віртуальних вимірювань довжини піраміди на поверхні зразка кремнієвих сонячних елементів представлений на рис. 9.

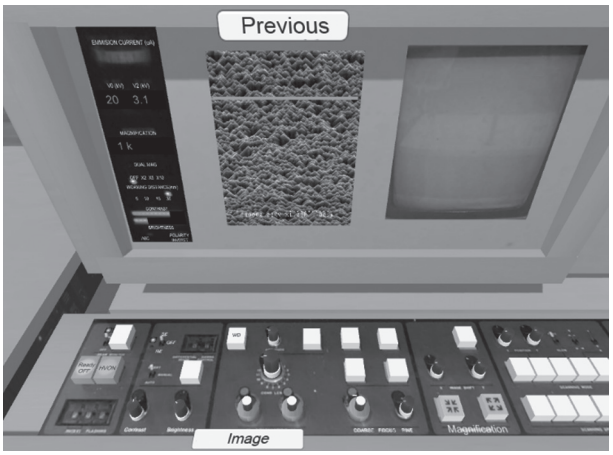


Рис. 9. Панель управління відеооператора

Оскільки масштаб зображення та кут нахилу зразка пластини добре відомі, поверхню можна точно дослідити. Ці операції дуже важливі, оскільки вони в основному використовуються в промисловості для контролю якості зразків.

Відповідно, за допомогою цього програмного забезпечення користувач може модифікувати та вдосконалити продукт, або навіть використовувати існуючі елементи, спеціально розроблені для цієї роботи, та створити нові лабораторні роботи.

Висновки

Зростання популярності технологій доповненої реальності пов'язане з ривком в їх розвитку. Завдяки перевагам сучасних технологій доповненої реальності можна по-новому отримувати візуальну інформацію, що може вирішити безліч пов'язаних з візуалізацією інформації проблем в різноманітних індустріях.

За останні 2 роки ми перебуваємо в розпалі глобальної пандемії. В умовах карантину університети повинні тимчасово закрити навчальні лабораторії та відкласти практичні роботи зі складним обладнанням. Особливо важливо для студентів природничих дисциплін продовжувати проводити експериментальні дослідження, але безпека студентів та інженерів-лабораторії під пріоритетом. У звичайних умовах студенти проводять кілька годин на тиждень у лабораторіях, відповідно, навчання поза лабораторією не дає студентам можливості набути практичних навичок, що призводить до зниження якості освіти. Візуалізація лабораторних робіт знімає низку проблем університетів, які забезпечують якісну освіту та забезпечують віддалену практичну лабораторну роботу.

Список літератури:

[1] Augmented Reality Tool Kit URL: http://www.hitl.washington.edu/ artoolkit/.

[2] Доповнена реальність або AR-технології. URL: http://thefuture.news/lessons/ua/ar

[3] *Goldman S.* Global Investment Research. URL: http://www.goldmansachs.com/careers/divisions/global-investment-research

[4] *Ларин М. С.* Работа с пакетом программ Open Computer Vision // Науч.-техн. вестн. СПбГУ ИТМО. 2008. Вып. 48. С. 95.

[5] *Roger Y. Tsai.* An efficient and accurate camera calibration technique for 3D machine vision // Proc. of IEEE Conf. on Computer Vision and Pattern Recognition. Miami Beach, FL, 1986. P. 364—37.

[6] OpenGL Projection Matrix URL: http://www.songho.ca/opengl/gl_ projectionmatrix.html.

[7] *Roger Y. Tsai.* A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off- the-shelf TV cameras and lenses // IEEE J. of Robotics and Automation. 1987. Vol. RA-3, N 4. P. 323—344.

[8] Automatic generation and detection of highly reliable fiducial markers under occlusion URL: https://www.researchgate.net/publication/260251570_Automatic_generation_and_detection_of_highly_reliable_fiducial_markers_under_occlusion

[9] *Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool*. "SURF: Speeded Up Robust Features." Computer Vision and Image Understanding (CVIU).Vol. 110, No. 3, pp. 346–359, 2008.

[10] *Rublee, Ethan, et al.* "ORB: an efficient alternative to SIFT or SURF." Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

[11] *Alahi, Alexandre, Raphael Ortiz, and Pierre Vandergheynst.* "Freak: Fast retina keypoint." Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012.

[12] *Calonder, Michael, et al.* "Brief: Binary robust independent elementary features." Computer Vision–ECCV 2010. Springer Berlin Heidelberg, 2010. 778-792.

[13] *Leutenegger, Stefan, Margarita Chli, and Roland Y. Siegwart.* "BRISK: Binary robust invariant scalable keypoints." Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

[14] *Gil Levi and Tal Hassner*, LATCH: Learned Arrangements of Three Patch Codes, IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Placid, NY, USA, March, 2016.

[15] Sus, B., *Tmienova, N., Revenchuk, I., Vialkova, V.* Development of virtual laboratory works for technical and computer sciences. -Communications in Computer and Information Science, 2019, 1078 CCIS, pp. 383–394.

[16] Julie Jebeile. Explaining with Simulations: Why Visual Representations Matter,2018-Perspectives on Science 26 (2): 213-238.

[17] *Chodos, D., Stroulia, E., & King, S.* (2011). Developing a virtual-world simulation. In Proceedings of the 3rd Workshop on Software Engineering in Health Care (pp. 71–78). New York, NY, USA: ACM. doi:10.1145/1987993.1988007.

[18] U. Technologies, "Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations." https://unity.com/, last accessed 2020/01/28

[19] Sus, B., *Revenchuk, I., Tmienova, N., Bauzha, O., Chaikivskyi, T.* Software System for Virtual Laboratory Works.- 2020 IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2020 — Proceedings, 2020, 1, pp. 396–399.

[20] Sus, B., *Revenchuk, I., Bauzha, O., Zagorodnyuk, S.*Virtual laboratory as custom e-learning implementation and design solution.- CEUR Workshop Proceedings, 2021, 2833, pp. 177–187.

Надійшла до редколегії 02.04.2021

УДК 004.8 DOi 10.30837/ bi.2021.1(96).15

И.И. Евтушенко[1],А.В. Вечур[2]

[1]ХНУРЭ, г. Харьков, Украина, ivan.yevtushenko@nure.ua,
ORCID iD: 0000-0001-6085-4874

[2]ХНУРЭ, г. Харьков, Украина, alexander.vechur@nure.ua,
ORCID iD: 0000-0001-9605-1475

РАСПОЗНАВАНИЕ ХУДОЖНИКА ПО КАРТИНЕ

Разработан алгоритм определения автора картины с изображением. Задача является сложной, так как сфера искусства традиционно плохо поддаётся формализации. К тому же, при обработке нужно минимально видоизменять картины, иначе информация о стиле может быть потеряна. Алгоритм основан на современных подходах, а именно на свёрточной нейронной сети с архитектурой RegNet. Эта архитектура известна не только своей мощностью, но и интересным механизмом проверки гипотез, который обсуждается в работе. Для обучения используется набор данных «Best Artworks of All Time». В рамках экспериментов проводится обучение с использованием нескольких функций потерь случайных изменений данных, предварительного обучения без учителя, угасающей скорости обучения.

АРХИТЕКТУРА, ГЛУБОКОЕ ОБУЧЕНИЕ, СВЁРТОЧНАЯ НЕЙРОННАЯ СЕТЬ, ЭПОХА, КАРТИНА, КЛАССИФИКАЦИЯ, НАБОР ДАННЫХ, ОБУЧЕНИЕ, ПЕРЕОБУЧЕНИЕ, ПРОСТРАНСТВО, СТИЛЬ, ФУНКЦИЯ ПОТЕРЬ, ХУДОЖНИК

Євтушенко І.І., Вечур О.В. Розпізнавання художника за картиною. Розроблено алгоритм визначення автора картини із зображення. Завдання є складним, так як сфера мистецтва традиційно погано піддається формалізації. До того ж, при обробці потрібно мінімально видозмінювати картини, інакше інформація про стилі може бути втрачена. Алгоритм заснований на сучасних підходах, а саме на згортковій нейронної мережі з архітектурою RegNet. Ця архітектура відома не тільки своєю потужністю, але і цікавим механізмом перевірки гіпотез, який обговорюється в роботі. Для навчання використовується набір даних "Best Artworks of All Time". У рамках експериментів проводиться навчання з використанням кількох функцій втрат випадкових змін даних, попереднього навчання без вчителя, згасаючої швидкості навчання.

АРХІТЕКТУРА, ГЛИБОКЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ЕПОХА, КАРТИНА, КЛАСИФІКАЦІЯ, НАБІР ДАНИХ, НАВЧАННЯ, ПЕРЕНАВЧАННЯ, ПРОСТІР, СТИЛЬ, ФУНКЦІЯ ВТРАТ, ХУДОЖНИК

Yevtushenko I.I., Vechur O.V. Artist recognition by painting. An algorithm for determining the author of a painting with an image has been developed. The task is difficult, since the field of art has traditionally been difficult to formalize. In addition, during processing, you need to modify the paintings to a minimum, otherwise information about the style may be lost. An algorithm is based on modern approaches, namely a convolutional neural network with the RegNet architecture. This architecture is known not only for its power, but also for its interesting hypothesis testing mechanism, which is discussed in the work. The data set "Best Artworks of All Time" is used for training. Within the framework of the experiments, training is carried out using several functions of loss of random data changes, preliminary training without a teacher, and a dying learning rate.

ARCHITECTURE, DEEP LEARNING, CONVENTIONAL NEURAL NETWORK, ERA, PICTURE, CLASSIFICATION, DATA SET, LEARNING, RETAINING, SPACE, STYLE, LOSS FUNCTION, ARTIST

Вступление

Сегодня информационные технологии играют огромную роль в мире не только потому что они хорошо изучены, но и потому что они все чаще и чаще автоматизируют рутинную работу людей на предприятиях. Эффективность бизнеса повышается, ведь компьютеры способны работать быстро, точно, дешево и без перерыва, в то время как человек имеет возможность сконцентрироваться на других.

Но с некоторых пор компьютеры умеют решать даже сложные задачи, которые, как казалось ранее, способен решать только мозг человека. Такие случаи принято называть задачами искусственного интеллекта, и их список очень велик: это может быть чтение рукописного текста, управление транспортом, выявление климатических аномалий, общение с людьми и тому подобное. Они решаются большим количеством математических методов еще с прошлого века, но за последние десять лет особый успех и популярность получили алгоритмы, способные со временем улучшать качество своей работы, анализируя статистику данных, с которыми они работают — алгоритмы машинного обучения. За это время они с научных исследований переросли в реально работающих программные системы, и сегодня почти каждое известное приложение или компания определенным образом использует их для улучшения качества работы.

Особым прорывом в сфере машинного обучения стали нейронные сети, ведь хоть они и были придуманы в середине XX века, только недавно стало понятно, насколько хорошо они способны анализировать

тексты, изображения, звуки и тому подобное. Это дало возможность попробовать использовать их даже в, наверное, наиболее гуманном направлении - искусстве. Нейронные сети обучали генерировать стихи, писать музыку, рисовать картины. Конечно, как и в жизни, большинство их произведений не была оценено, но некоторые из них стали известными в обществе. Безусловно, такие работы имеют, в первую очередь, академический и развлекательный интерес, но иногда они могут позволить более глубоко понять процесс создания шедевров искусства, и на основе этого делать новые интересные выводы, которые не очевидны искусствоведам при визуальном анализе.

В данной работе речь пойдёт о задаче определения автора картины с помощью модели машинного обучения.

1. Связанные работы

С точки зрения анализа данных это является случаем классификации, когда имеется некоторое конечное множество классов (художников), и для наблюдений (картин) прогнозируется, к какому из классов и с какой вероятностью оно относится.

Уже на протяжении нескольких лет при решении задач классификации изображений наилучшие результаты демонстрируют нейронные сети. Их архитектура практически ничем не ограничена, за счёт чего вариаций таких алгоритмов существует огромное количество. С одной стороны, это дает возможность добиваться впечатляющих результатов на разных задачах, но с другой стороны, поиски той самой оптимальной архитектуры отнимают большое количество интеллектуальных, вычислительных, материальных и временных ресурсов. Команда из Facebook AI Research весной 2020-ого года попробовала решить эту проблему в своей работе “Designing Network Design Spaces” [1], попытавшись выяснить “рецепт” успешной архитектуры. Наиболее важными в их работе можно считать даже не сами правила проектирования, которые авторам удалось выработать, а подход, которым им удалось их получить, так как он хорошо обобщается на методы проведения других научных исследований. В итоге для сообщества это стало не статьёй с очередной архитектурой, а целым пособием по проведению быстрых и эффективных экспериментов. Поэтому в научном сообществе эта работа ценится по сей день.

Изначально авторы берут некоторую простую базовую архитектуру, похожую на известный ResNet [2], которая имеет большое количество параметров, таких как глубина, ширина, количество блоков и так далее, каким-то образом влияющие на ее качество, и назвали её AnyNet. Они трактуют множество этих параметров как некоторое пространство, из которого мы случайным образом выбираем архитектуру для тренировки. Задачей работы является сузить это пространство параметров до такого, в котором вероятность получить хорошую архитектуру будет максимальной.

Для того, чтобы сравнивать два пространства, был принят следующий критерий: из каждого выбирается по 500 случайных архитектур, после чего каждая из них обучается в течении 10 эпох на решении задачи ImageNet [3]; затем для обоих пространств составляется функция распределения успешных архитектур, скорость роста которой демонстрирует качество пространства; на основе сравнений этих функций принимается статистически значимое решение о том, лучше ли новое пространство. На рис. 1 слева продемонстрирован процесс последовательных переходов к более узким пространствам, а справа их функции распределения.

Таким образом, большое количество выбранных архитектур позволяет быть уверенным, что улучшение не есть случайным, при этом их запуски можно делать параллельно, а тот факт, что они короткие и быстрые, не мешает им обобщаться на полномасштабные тренировки. Это подтверждается тем, что таким методом авторам удалось сузить пространство архитектур до достаточно хорошего для того, чтобы затем обучить полноценную архитектуру, названную RegNet, и получить лучшие результаты для задачи

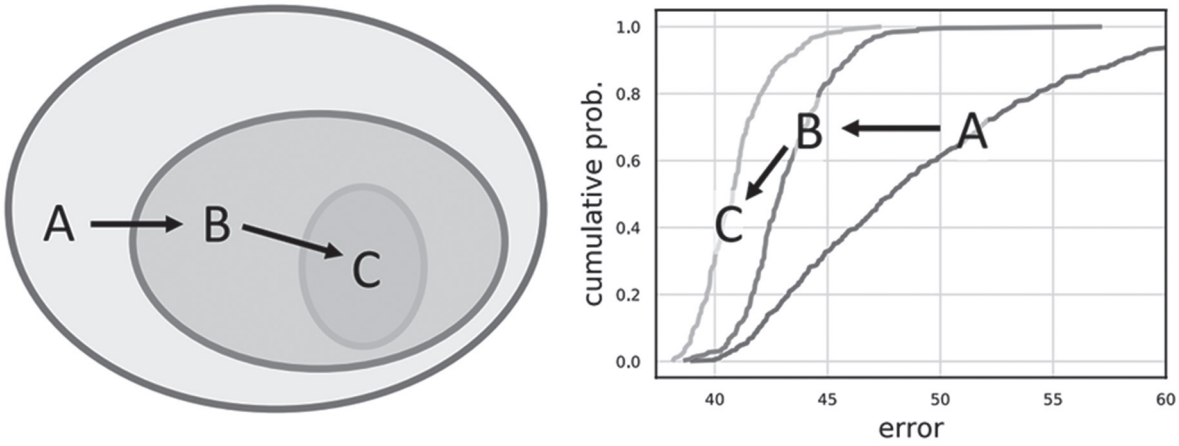


Рис. 1. Процесс перехода между пространствами архитектур

ImageNet на момент публикации статьи. Вдобавок к этому были сформулированы и интерпретированы правила, которые позволили прийти к успеху. Через короткое время вышли новые архитектуры, которым удалось улучшить этот результат, но подобный принцип проведения экспериментов стал повсеместно использоваться исследователями.

Важным преимуществом нейронных сетей является тот факт, что в ходе тренировки они обучаются формировать полезные сжатые представления наблюдений. Сжатие происходит в архитектурах неизбежно с целью экономии вычислительных ресурсов, а полезная информация собирается для решения финальной задачи. Помимо того, что именно возможность выучивать такие представления делает данный вид алгоритмов столь выразительным, часто всё эти же представления могут быть использованы для решения каких-то других задач. Однако, на практике сеть не всегда обучается формировать хорошие представления, так как она может иметь не совсем удачную функцию потерь, страдать проблемой переобучения, несбалансированности классов и тому подобное. Одним из самых эффективных методов навязывания нейронной сети полезных представлений является использования многозадачного обучения, хорошо описанного в статье “An Overview of Multi-Task Learning in Deep Neural Networks” [4]. Суть подхода заключается в том, что, как ни странно, если мы обучаем алгоритм решать сразу несколько разных задач, а не одну, то в итоге он способен обучиться решать каждую из них более эффективно. Происходит это главным образом по той причине, что как раз в ходе решения большого количества задач сеть вынуждена сохранить как можно больше полезной информации в своём внутреннем представлении данных, и может оказаться, что как раз этой информации не хватало для успеха в отдельной задаче.

Отдельно стоит отметить, что задача классификации изображений является методом обучения с учителем, в которых для тренировки алгоритма нужны не только наблюдения, но и правильно указанные классы для них. Плюс этого в том, что такие методы очень хорошо изучены и широко используются, но не для каждой задачи есть возможность найти достаточно большие объемы данных. Эта проблема актуальна и в нашем исследовании, так как не у всех художников физически есть много картин, доживших до сегодняшних дней хоть в каком-то виде. Поэтому всё больше исследований проводится в области самообучения, когда задача обучения подбирается таким образом, что наблюдения сами для себя становятся размеченными данными. Одним из наиболее успешных примеров таких работ является “Unsupervised Representation Learning by Predicting Feature Decoupling” [5], в рамках которой входное изображений поворачивается на определенный угол, а алгоритм должен спрогнозировать угол наклона. Это заставляет нейронную сеть выучить хорошие представления наблюдений, и при этом им совсем не нужна разметка. После предварительного обучения сети таким подходом можно на уже небольшом наборе размеченных данных продолжить обучать её под финальную задачу. Это может помочь получить более качественные результаты за счет того, что в ходе предварительного обучения алгоритм смог сформировать полезные представления для значительно большего объема данных, и всё, что ему остаётся, это адаптировать их для классификации художников.

2. План работы

В рамках данной работы предлагается обучить собственную нейронную сеть для решения задачи распознавания художника по картине. Её архитектура унаследует RegNet, предварительно обученный для решения задачи ImageNet. Но сам процесс обучения будет заметно отличаться и потребует большого количества экспериментов, так как картины художников потребуют особого вида обработки.

В частности, планируется проверить гипотезы по оптимальному разрешению входного изображения (112x112/224x224/448x448), попробовать различные методы оптимизации (SGD, Adam [6], LAMB [7]), различные методы изменения скорости обучения (константный, затухающий, косинусный [8]), методы регуляризации для борьбы с переобучением (Dropout [9], Регуляризация Тихонова, Модификация данных). Также для борьбы с несбалансированностью классов стоит попробовать использовать в качестве функции потерь не только стандартную для классификации кросс-энтропию, но и взвешенную, где веса классов обратно пропорциональны их частоте, и DiceLoss [10], где больший вес имеют сложные для модели примеры.

Важными экспериментами будут попытки использовать многозадачное обучение и предварительное самообучение для получения лучших представлений и, соответственно, качества. Гипотезы предлагается проверять последовательно, а для их проверки обучать каждую модель в течение 20 эпох и сравнивать их качество по метрике macro f1-score. В отличие от точности, она хорошо учитывает несбалансированность классов, поэтому если модель будет совсем плохо работать для какого-то художника, то это не останется незамеченным. При этом точность также будет озвучиваться для лучшей интерпретации экспериментов.

В работе используется язык Python, а наиболее важной библиотекой — PyTorch [11], что является одним из наиболее популярных решений сегодня. Обучение алгоритмов будет проходить на компьютере с GPU для ускорения вычислений.

3. Данные

Преимуществом поставленной задачи является тот факт, что данные, необходимые для тренировки модели распознавания — это картины и их авторы. Эти данные можно найти в интернете в открытом доступе, для их сбора не придется проводить или генерировать наблюдения, что является частой проблемой анализа данных в других отраслях, например, в медицине.

С другой стороны, есть проблема, связанная с тем, что для качественной работы нейронных сетей тренировочных данных нужно много. При этом мы никак не сможем собрать больше картин художника, чем дожило до сегодняшних дней, поэтому данный вопрос решается не на уровне сбора данных, а на уровне проектирования и обучения модели.

Одним из популярных способов сбора данных из интернета выглядит следующим образом: находится веб-источник, на котором есть множество необходимых нам данных, например, интернет-галерея, после чего создаётся скрипт, который рекурсивно посещает разные страницы веб-сайта и скачивает изображения из него, попутно вылавливая имя указанного там же автора. Естественно, такие данные требуют последующего осмотра, так как в работе программы могли быть допущены ошибки, но в ситуациях, когда другого выхода нет, этот метод очень полезен.

К счастью, сообщество исследователей в области анализа данных очень активно делится множеством полезных ресурсов, от чего выигрывают все. Одной из самых известных площадок для их коммуникации является Kaggle [12]. Главной целью этого сайта всегда было и остаётся проведение онлайн-соревнований по машинному обучению, но со временем здесь также появились форум, курсы и возможность публиковать свои наборы данных. По запросу удалось найти набор данных “Best Artworks of All Time” [13]. В нём содержится 8446 картин 50 разных художников. На рис. 2 видна проблема несбалансированности классов, так как несмотря на то, что большая часть художников имеет от 50 до 200 картин, у Джексона Поллока их всего 24, что почти в 40 раз меньше, чем у Ван Гога, имеющего 877 картин в этом наборе. Это может вызвать проблемой, так как алгоритм выучит одних лучше других.

Преимуществом этого набора данных является тот факт, что помимо разметки авторов для картин здесь присутствует дополнительная информация об авторах: их национальность, годы жизни и жанр. То есть мы можем для картины помимо самого автора прогнозировать эти переменные, реализуя концепцию многозадачного обучения. Благодаря этому у алгоритма будет возможность понимать, что хоть картины Клода Моне, Эдуарда Мане и Микеланджело относятся к разным авторам, первые два имеют что-то общее — национальность, жанр и приблизительные годы жизни, а третий отличается от них. Ровно как знание этого делает искусствоведа более профессиональным, так и нейронная сеть от него становится более эффективной.

Итого, распределения в наборе данных соответствуют реальности, данных в нём достаточно, и у них хорошая разметка, так что как основной источник информации он подходит для исследования. Но дополнительно нам хотелось бы иметь ещё больший набор неразмеченных данных для проверки эффективности методов предварительного самообучения. Эти данные не обязательно должны быть из того же распределения изображений, но при этом максимально приближены к предметной области. Всё на том же Kaggle удалось найти данные “Wiki-Art: Visual Art Encyclopedia” [14], в котором содержится 96000 изображений. Для них, конечно, не известны авторы, ведь не все из них являются живописью, но при

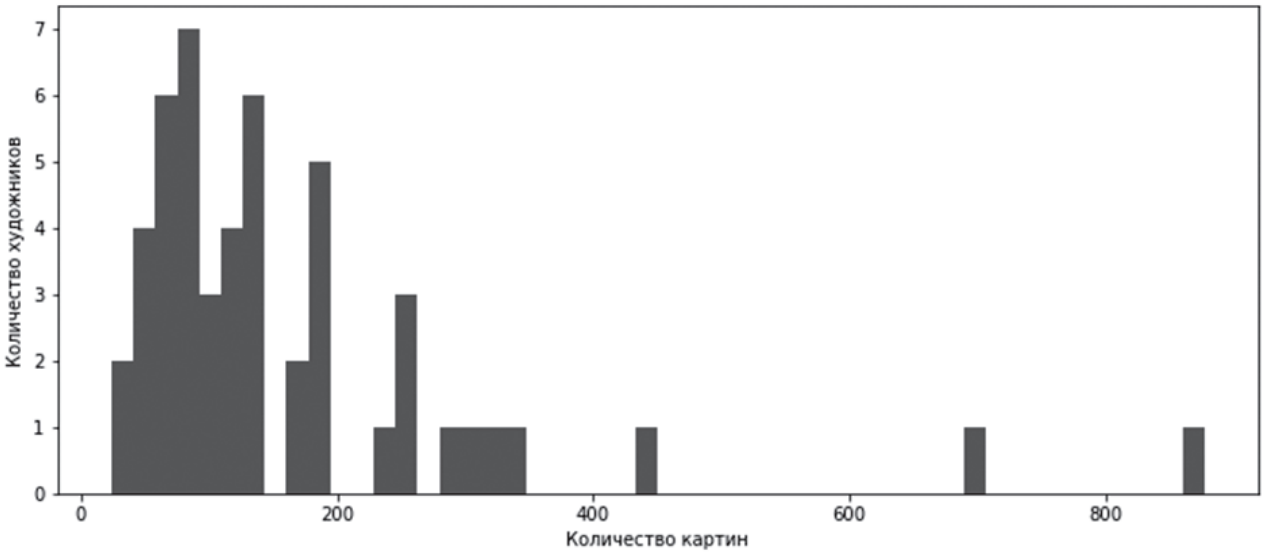


Рис. 2. Распределение количества картин художников

этом они разбиты на 14 категорий — ландшафты, портреты, животные, морские пейзажи и так далее. Эти метки также интересно попробовать использовать при предварительном самообучении в рамках концепции многозадачного обучения.

4. Эксперименты

Очевидно, что в моделях машинного обучения главным являются данные и то, какие преобразования мы делаем над ними перед тем, как подать на вход алгоритму. Часто полезно делать эти преобразования случайными для того, чтобы повысить разнообразие данных. Среди таких преобразований мы предлагаем делать поворот картины по вертикальной оси, так как очень маловероятно, что это мешает узнать художника, и при этом повышает разнообразие изображений в 2 раза. Также все картины должны быть одного размера, чтобы нейронная сеть, во-первых, могла адаптироваться к конкретным масштабам шаблонов в картинах, а во-вторых, мы имели возможность обучить её не по одной картинке в один момент времени, а пакетами, которые эффективно складываются в одну многомерную матрицу. Для этого предлагается сначала масштабировать все картины таким образом, чтобы по своей меньшей стороне они имели размер 256 пикселей, но при этом они сохраняли свое исходное соотношение сторон. Затем из полученной картины вырезается случайный квадрат 224x224 (стандартный размер изображений для классификационных архитектур), и эта случайность также повышает разнообразие наших данных. Естественно, такие преобразования нужны только во время обучения, в режиме тестирования достаточно отмасштабировать картину по тому же принципу и вырезать квадрат 224x224 из ее центра.

Как уже упоминалось ранее, архитектурой нейронной сети в ходе экспериментов являлась RegNet. Для неё есть предварительно обученные на данных ImageNet веса, начинать тренировку с которых часто является более быстрым решением, чем со случайной инициализации. В рамках первого эксперимента в качестве алгоритма оптимизации мы использовали Adam с параметрами по умолчанию, часто это дает хорошие результаты благодаря адаптивности этого алгоритма под ход обучения. Функция потерь для классификации стандартная — кросс-энтропия. Длительность обучения — порядка 15 минут на 20 эпох. Как было упомянуто в разделе данных, метрикой для проверки гипотез является macro f1-score.

В итоге такая первичная конфигурация тренировки сразу получает достойные результаты — f1-score равен 0.655, точность — 72.5%. Стоит отметить, что модель достаточно сильно переобучена, так как на тренировочной выборке ее точность превышает 99%.

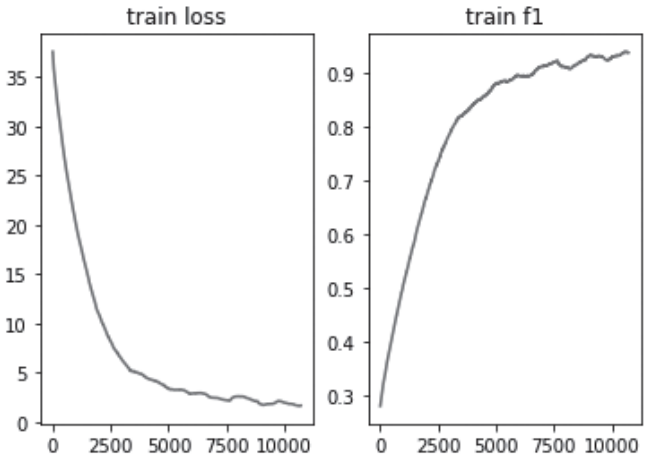


Рис. 3. Визуализация процесса обучения: пошаговые изменения значения функции потерь (слева) и метрики f1-score (справа)

Для борьбы с этим эффектом хорошей идеей часто является усилить случайные преобразования в данных, чтобы повысить их разнообразие и усложнить алгоритму задачу. В качестве таких преобразований были попытки использовать добавление нормального шума, перевороты по горизонтали, преобразование цвета, перемасштабирования и так далее. Оказалось, что они не дают прирост, а даже ухудшают метрики на несколько процентов. Скорее всего, это связано с тем, что в картинах очень важны мелкие детали, которые могут быть испорчены подобными преобразованиями, и в итоге художник идентифицируется ещё хуже. Была попытка увеличить разрешение входного изображения с 224 до 448 для того, чтобы терять меньше деталей, но в итоге это привело только к замедлению тренировки. При разрешении 112 наблюдается сильное падение качества. В результате изначальная конфигурация преобразований данных оказалась самой эффективной.

Также причиной переобучения может быть тот факт, что у некоторых художников настолько мало картин, что нейронная сеть просто их запоминает, не обобщаясь на те, которые она не увидела. Попыткой устранить эту проблему было использование больших весов для таких классов, чем для популярных, либо же использование DiceLoss вместо кросс-энтропии, которая сильнее штрафует примеры, на которых алгоритм ошибается. Оба эти метода вызвали нестабильность в процессе обучения, из-за чего при их использовании значение метрики только ухудшалось. Такой же неудачный эффект приносили попытки использовать Dropout и регуляризацию Тихонова.

В рамках экспериментов с процессом оптимизации выяснилось, что менять алгоритм Adam смысла нет, так как SGD работает быстро, но сходится гораздо хуже, а LAMB работает дольше, но при этом финальные результаты дает приблизительно те же. А вот усложнение способа изменения скорости обучения

оказалось эффективным — уменьшая его в 0.7 раз каждую эпоху удалось добиться f1-score 0.695, а точности — 75.5%.

Следующий шаг для улучшения — многозадачное обучение. Мы воспользовались доступными метками о жанре, национальности и годах жизни. Жанры принимают 31 различных значений, национальность — 17, предсказания обоих этих переменных являются решением все той же задачи классификации, функцией потерь для которой традиционно взята кросс-энтропия. Для лет жизни сначала было вычислено среднее между годом рождения и годом смерти, чтобы прийти к одному численному показателю, затем они были масштабированы в промежуток от 0 до 1, и наконец нейронной сети была поставлена задача регрессии, решаемая путем минимизации среднеквадратичного отклонения между предсказаниями и метками. В результате алгоритм научился предсказывать жанр с точностью 76.5%, национальность с точностью 80%, год со средней ошибкой в 100 лет, и, что самое интересное, стал ещё лучше решать задачу идентификации художника. F1-score вырос до 0.722, а точность стала 77.75%. Примечательно, что при таком подходе алгоритм получает хорошее качество на более ранних эпохах. Скорее всего, это связано с тем, что модель получает больше градиентов с разных задач, и каждый шаг обучения становится более информативным.

Финальный эксперимент — предварительное самообучение. Для него было использовано больше преобразований над данными, чем в задачи распознавания автора картины, так как в данном случае они менее критично могут повлияют на целевую метку, но архитектура и параметры тренировки сохранились теми же. Алгоритм за 8 эпох, которые из-за большого количества данных продлились 6 часов, был обучен распознавать один из четырёх возможных углов поворота с точностью 67%, а категорию — с точностью 70%. При этом последующее обучение этой модели для решения финальной задачи привело к f1-score 0.65, а точности 71.5%. Это неплохие результаты, но они заметно хуже, чем предыдущие. На самом деле, это связано с тем, что в предыдущих экспериментах мы также неявно использовали подход предварительного обучения, когда взяли параметры модели, которые были тщательно сформированы для качественного решения задачи ImageNet. В то время как в данном эксперименте мы самостоятельно обучили модель на собственных данных в режиме самообучения, что, конечно, лучше, чем начинать обучение со случайной инициализации, но всё же хуже, чем веса для ImageNet, благодаря чему этот набор данных остается самым популярным в компьютерном зрении по сей день. Однако, перспективы подходов самообучения гораздо шире, так как они способны использовать необъемлемый набор данных.

Выводы

По итогу данной работы удалось разработать модель машинного обучения, основанную на сверточных нейронных сетей, которая распознаёт, какому из 50 художников принадлежит данная картина. Ключевой метрикой для выбора оптимальной модели была f1-score. Финальная модель использует архитектуру RegNet, оптимизатор Adam с затухающей скоростью обучения, многозадачное обучение и модификацию данных в виде случайных вырезаний и вертикальных поворотов. Несмотря на попытки устранить переобучение, добавление различных методов регуляризации делают обучение нестабильным и лишь ухудшают метрики, поэтому финальная модель всё равно страдает от этой проблемы. В последующих исследованиях можно попробовать решить эту проблему, например, повышением разнообразия данных, или использованием ансамблей алгоритмов. Эксперимент с предварительный обучением через методы самообучения показал, что они способны помочь модели получать неплохие результаты, но пока что методы обучения с учителем остаются более предпочтительным решением, так как их качественно обученные параметры есть в открытом доступе для большинства современных архитектур.

Итоговое качество модели — метрика F1-score 0.722, точность — 77.75%. С большинством картин она справляется без проблем, например, на рис. 4 изображена картина Клода Моне, которая была правильно идентифицирована с вероятностью 99%.

Рис. 4. Легко идентифицированная картина Клода Моне

Если считать точность не по наиболее вероятному художнику, а по тройке, предложенной моделью, то точность возрастает до 91.5%. Так уже правильно идентифицируются и более сложные изображения, например, одна из картин Питера Пауля Рубенса (см. рис. 5). Она была отнесена к её автору с вероятностью 26%, в то время как к Эжену Делакруа — с вероятностью 41%. Интересно, что если посмотреть на работы Делакруа, то они действительно похожи на данную по стилистике, но при этом в тренировочном наборе данных их всего 45, поэтому модель плохо изучила этого художника и ошибочно отнесла картину к нему.

Рис. 5. Правильно идентифицированная картина Питера Пауля Рубенса по топ-3 предсказаниям

Тем не менее, итоговый алгоритм всё же имеет проблемы с некоторыми картинами, на которых так и не удалось получить качественный прогноз. К примеру, такой стала картина всё того же Питера Пауля Рубенса, изображенная на рис. 6.

Рис. 6. Неправильно идентифицированная картина Питера Пауля Рубенса

Согласно предсказанию, картина принадлежит ему с вероятностью менее 1%, в то время как Альбрехту Дюреру — с вероятностью 47%, Сандро Боттичелли — 16%, Иерониму Босху — 10%. Эти три художника жили и творили в конце XIV — начале XV века в направлении Ренессанс, в то время Рубенс вел деятельность примерно на 100 лет позднее, и был уже представителем Барокко. Каждый из этих художников имеет около 150 картин в используемом наборе данных, что не так уж и мало. Это демонстрирует несовершенство модели, ведь несмотря на высокую точность в среднем иногда ее работу сложно интерпретировать.

Дополнительно модель способна прогнозировать жанр с точностью 76.5% (90.4% на топ-3), а также национальность с точностью 80% (94.4% на топ-3).

Список литературы:

[1] I. Radosavovic, R. P. Kosaraju, R. Girshick и др. Designing Network Design Spaces, 2020. URL: https://arxiv.org/pdf/2003.13678.pdf.

[2] K.He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition, 2015. URL: https://arxiv.org/pdf/1512.03385.pdf.

[3] O. Russakovsky J. Deng H. Su и др. ImageNet Large Scale Visual Recognition Challenge, 2015. URL: https://arxiv.org/pdf/1409.0575.pdf.

[4] Sebastian Ruder An Overview of Multi-Task Learning in Deep Neural Networks, 2020. URL: https://arxiv.org/pdf/1706.05098.pdf.

[5] S. Gidaris, P. Singh, N. Komodakis Unsupervised Representation Learning by Predicting Image Rotations, 2018. URL: https://arxiv.org/pdf/1803.07728.pdf.

[6] D. P. Kingma и J. L. Ba Adam: A Method for Stochastic Optimization. URL: https://arxiv.org/pdf/1412.6980.pdf.

[7] Y. You J. Li S. Reddi и др. LARGE BATCH OPTIMIZATION FOR DEEP LEARNING: TRAINING BERT IN 76 MINUTES. URL: https://arxiv.org/pdf/1904.00962.pdf.

[8] I. Loshchilov и F. Hutter SGDR: STOCHASTIC GRADIENT DESCENT WITH WARM RESTARTS. URL: https://arxiv.org/pdf/1608.03983.pdf.

[9] N. Srivastava G. Hinton A. Krizhevsky и др. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. URL: https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf.

[10] X. Li X. Sun Y. Meng и др. Dice Loss for Data-imbalanced NLP Tasks. URL: https://arxiv.org/pdf/1911.02855.pdf.

[11] Документация PyTorch: https://pytorch.org/docs/stable/index.html.

[12] Сайт Kaggle: https://www.kaggle.com.

[13] Набор данных Best Artworks of All Time: https://www.kaggle.com/ikarus777/best-artworks-of-all-time.

[14] Набор данных WikiART: https://www.kaggle.com/ipythonx/wikiart-gangogh-creating-art-gan.

Поступила в редколлегию 16.04.2021

УДК 004.89 DOi 10.30837/ bi.2021.1(96).16

І.Ю. Шубін[1], Г.Г. Четвериков[2], В.А Ляшик[3], Н.О. Шанідзе[4]

[1]Кандидат технічних наук, професор кафедри програмної інженерії,
Харківський національний університет радіоелектроніки,
igor.shubin@nure.ua, ORCID iD: 0000-0002-1073-023X

[2]Доктор технічних наук, професор кафедри програмної інженерії,
Харківський національний університет радіоелектроніки,
grirorij.chetverykov@nure.ua, ORCID iD: 0000-0001-5293-5842

[3]Аспірант кафедри програмної інженерії,
Харківський національний університет радіоелектроніки,
volodymyr.liashyk@nure.ua, ORCID iD: 0000-0001-7326-0813

[4]Кандидат соціологічних наук, доцент кафедри соціології та політології ,
Національний технічний університет «Харківський політехнічний інститут»,
nashanidze@ukr.net ORCID iD 0000-0002-9613-186X

МЕТОДИ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ АДАПТИВНОГО ТЕСТУВАННЯ ЗНАНЬ

Під адаптивним тестовим контролем розуміють комп'ютеризовану систему науково обґрунтованої перевірки й оцінювання результатів навчання, що має високу ефективність за рахунок оптимізації процедур генерації, пред'явлення й оцінки результатів виконання адаптивних тестів, що заснована на методах побудови та оптимізації логічних мереж. Алгоритми підбору й пред'явлення завдань будуються за принципом зворотного зв'язку, коли при правильній відповіді суб'єкта навчання чергове завдання вибирається більш важким, а невірна відповідь спричиняє пред'явлення наступного більш легкого завдання, ніж те, на яке суб'єктом навчання була дана невірна відповідь. Також є можливість завдання додаткових питань по темах, які суб'єкт навчання знає не дуже добре для більш точного з'ясування рівня знань у даних областях. Вибір алгоритмів тестування наразі фактично обмежений формами представлення тестових завдань і алгоритмами оцінювання результатів тестування. Досягнення більш високих результатів і підвищення мотивації навчання в остаточному підсумку є основною метою тестування знань. Для визначення базового алгоритму, необхідно навести сценарій роботи системи. У його основі лежить модель приймання іспиту викладачем у студента, як модель адаптивного тестування. Такий вибір сценарію роботи системи обумовлений тим, що, по-перше, дана процедура історично добре формалізована, по-друге, при проектуванні тестів, їх розробнику необхідно спиратися на загальноприйняті, відомі й використовувані їм методи з мінімальною модифікацією.

ДИСТАНЦІЙНА ОСВІТА, ЛОГІЧНА МЕРЕЖА, АЛГЕБРА СКІНЧЕННИХ ПРЕДИКАТІВ, ДИСТАНІЙНЕ ТЕСТУВАННЯ ЗНАНЬ, МОДЕЛЬ СУБ'ЄКТА НАВЧАННЯ

Шубин И.Ю., Четвериков Г.Г., Ляшик В.А., Шанидзе Н.А. Методы искусственных нейронных сетей для адаптивного тестирования знаний. Под адаптивным тестовым контролем понимают компьютеризированную систему научно обоснованной проверки и оценки результатов обучения, имеет высокую эффективность за счет оптимизации процедур генерации, предъявления и оценки результатов выполнения адаптивных тестов, основанная на методах построения и оптимизации логических сетей. Алгоритмы подбора и предъявления задач строятся по принципу обратной связи, когда при правильном ответе субъекта обучения очередное задание выбирается более тяжелым, а неверный ответ вызывает предъявления следующего более легкого задания, чем то, на которое субъектом обучения была дана неверная ответ. Также имеется возможность задания дополнительных вопросов по темам, которые субъект обучения знает не очень хорошо для более точного выяснения уровня знаний в данных областях. Выбор алгоритмов тестирования пока фактически ограничен формами представления тестовых заданий и алгоритмами оценки результатов тестирования. Достижения более высоких результатов и повышения мотивации обучения в конечном итоге является основной целью тестирования знаний. Для определения базового алгоритма, необходимо привести сценарий работы системы. В его основе лежит модель принятия экзамена преподавателем у студента, как модель адаптивного тестирования. Такой выбор сценария работы системы обусловлен тем, что, во-первых, данная процедура исторически хорошо формализована, во-вторых, при проектировании тестов, их разработчику необходимо опираться на общепринятые, известные и используемые им методы с минимальной модификацией

ДИСТАНЦІОННОЕ ОБРАЗОВАНИЕ, ЛОГИЧЕСКАЯ СЕТЬ, АЛГЕБРА КОНЕЧНЫХ ПРЕДИКАТОВ, ДИСТАНЦИОННОЕ ТЕСТИРОВАНИЕ ЗНАНИЙ, МОДЕЛЬ ОБУЧАЕМОГО

Shubin I.Yu., Chetverikov G.G., Liashyk V.A., Shanidze N.A. Methods Of Artificial Neural Networks for Adaptive Knowledge Testing. Adaptive test control is a computerized system of scientifically based verification and evaluation of learning outcomes, which is highly effective by optimizing the procedures for generating, presenting and evaluating the results of adaptive tests, based on methods of building and optimizing logical networks. Algorithms for selection and presentation of tasks are based on the principle of feedback, when the correct answer of the subject of training is the next difficult task, and the wrong answer causes the presentation of the next easier task than that to which the subject

of training the wrong answer was given. It is also possible to ask additional questions on topics that the subject does not know very well to clarify the level of knowledge in these areas. The choice of testing algorithms is currently actually limited by the forms of presentation of test tasks and algorithms for evaluating test results. Achieving higher results and increasing the motivation to learn is ultimately the main goal of testing knowledge. To determine the basic algorithm, it is necessary to provide a scenario of the system. It is based on the model of taking the exam by a teacher as a model of adaptive testing. This choice of the scenario of the system is due to the fact that, firstly, this procedure is historically well formalized, and secondly, when designing tests, their developer must rely on common, known and used methods with minimal modification.

DISTANCE EDUCATION, LOGICAL NETWORK, ALGEBRA OF FINITE PREDICATES, DISTANCE TESTING OF KNOWLEDGE, MODEL OF SUBJECT OF LEARNING

Вступ

Аналіз методів педагогічних вимірювань дозволяє зробити висновок про те, що одним з об'єктивних і ефективних методів контролю якості знань є тестовий метод, заснований на використанні педагогічних тестових матеріалів. У цей час важко назвати дисципліну, процес вивчення якої так чи інакше не містить в собі контроль знань в тестовій формі.

Дослідження показали, що, незважаючи на велику кількість розроблених тестів, вони мають певні недоліки, серед яких необ'єктивність вагових коефіцієнтів тестових завдань, неоптимальна кількість тестових завдань або одноваріантність тесту, наявність зв'язку між послідовними завданнями.

На практиці дуже часто виникає ситуація, коли одна й та сама група студентів без особливих зусиль справляється з усіма тестовими завданнями або, навпаки, не може впоратися із більшістю тестових завдань.

Таким чином, існує проблема коректності підбору складності тестових завдань з метою найбільш адекватної оцінки рівня знань студентів. В зв'язку зі збільшенням кількості й недостатньою якістю тестів, що застосовуються при навчанні у вузах, не завжди є можливим якісно визначити рівень навчальних досягнень студента, грунтуючись тільки на тестах, в яких кількість завдань є фіксованою (так звані тести фіксованої довжини). Цей недолік можливо усунути за допомогою застосування такого виду тестування, яке здатне «підлаштовуватися» під рівень знань студентів, варіювати складність і кількість завдань в залежності від правильності відповідей на них. Таким чином, процес тестування адаптується до рівня знань студента, що проходить тест. Цо дозволяє отримати більш достовірні результати, скоротити час, що потрібен для проходження тесту, не знижує мотивацію студентів до навчання й тестування. Подібні підходи прийнято називати адаптивним тестуванням.

При комп'ютерному адаптивному тестуванні тестові завдання формуються індивідуально для кожного студента, що екзаменується, з урахуванням результатів виконання попередніх завдань. Типи завдань, їх кількість та порядок проходження індивідуальні. Таким чином, адаптивне тестування не тільки дає більш об'єктивну оцінку знанням, умінням і навичкам студентів, але й дозволяє виявляти, які знання є помилковими або неповними, а також дозволяє формувати подальшу траєкторію навчання.

Проблеми вимірів і діагностики є найбільш складними як у теоретичному, так і у практичному відношенні. Під «виміром» зазвичай розуміють виявлення кількісних характеристик досліджуваних явищ або особливу процедуру, за допомогою якої числа або, принаймні, порядкові величини приписуються об'єктам за певними правилами. Сюди слід включати й математичну обробку результатів вимірів. При цьому в якості результату виміру виступає значення величини, що характеризує якість, знайдену шляхом її виміру.

Основна проблема полягає в тому, що часто неможливо безпосередньо виміряти потрібну величину, наприклад рівень знань студентів з певної дисципліни. Такі величини називають латентними. Для визначення латентної змінної використовують вимір інших змінних, на основі яких обчислюють потрібну величину.

Педагогічний вимір має свої рівні (оцінка результатів педагогічних вимірів; аналіз і узагальнення педагогічних явищ; ефективність засобів і методів навчання; оцінка досягнення цілей на рівні глобальних цілей, на рівні навчального предмета, на рівні навчального матеріалу).

До основних проблем педагогічних вимірів слід віднести можливу необ'єктивність, неоднозначність, латентність, забезпечення точності, стійкості та валідності [1].

Основним предметом педагогічних вимірів є розробка якісних тестів для виміру рівня підготовленості студентів, як найбільш точного засобу вимірювань. Ефективність організації та проведення педагогічних тестів підвищується за рахунок використання комп'ютерних технологій.

1. Постановка задачі

Проблеми вимірів і діагностики є найбільш складними як у теоретичному, так і у практичному відношенні. Під «виміром» зазвичай розуміють виявлення кількісних характеристик досліджуваних явищ або особливу процедуру, за допомогою якої числа або, принаймні, порядкові величини приписуються об'єктам за певними правилами. Сюди слід включати й математичну обробку результатів вимірів. При

цьому в якості результату виміру виступає значення величини, що характеризує якість, знайдену шляхом її виміру.

Основною ідеєю класичної теорії тестів є припущення про існування істинного бала (true score). Нерідко в одномірних вимірах дійсний бал називають параметром студента. При цьому передбачається, що кожному студентові можна поставити у відповідність єдине на момент виміру значення параметра, що не залежить від застосовуваного тесту. У цілому істинний бал – це ідеалізована константа випробуваного в гіпотетичній генеральній сукупності завдань нескінченного тесту.

Крім припущення про існування істинного балу в класичній теорії тестів виділяють кілька постулатів, що дозволяють побудувати математико-статистичний апарат для розробки науково обґрунтованих тестів і оцінки якості результатів педагогічних вимірів [2; 3]:

– емпірично отриманий результат виміру (X) є сумою дійсного результату виміру (T) і помилки виміру (E). Величини T і E зазвичай невідомі.

– істинний результат виміру можна виразити як математичне очікування E(X);

– кореляція істинних та помилкових компонентів по групі випробуваних дорівнює нулю;

– помилкові компоненти двох будь-яких тестів не корелюють;

– помилкові компоненти одного тесту не корелюють із дійсними компонентами будь-якого іншого тесту.

Крім цього, основу класичної теорії тестів становлять два визначення – паралельних та еквівалентних тестів [4].

Однак класична теорія має ряд недоліків, що обмежують її використання у сфері педагогічних вимірів.

Оцінка рівня підготовленості студента залежить від складності тесту. Так, якщо тест складається з важких тестових завдань, то частка правильних відповідей у студента буде низкою. Якщо ж тест складається з легких тестових завдань, то частка правильних відповідей у того ж самого студента буде високою.

Оцінка складності тестового завдання залежить від рівня підготовленості студентів. Так, якщо вибірка складається з добре підготовлених студентів, то складність тестових завдань буде невисокою. Якщо ж вибірка складається з погано підготовлених студентів, то труднощі тестових завдань буде високою.

Шкала виміру рівня підготовленості є нелінійною. Наприклад, та сама різниця в балах (5 балів) на краях і в середині шкали відповідає різному збільшенню в рівні підготовленості (100 балів – 95 балів) ≠ (50 балів – 45 балів).

Дисперсія оцінки рівня підготовленості найбільша в середині діапазону виміру й найменша по краях, що суперечить логіці побудови довірчих інтервалів.

Тестовий бал студента нелінійно залежить від рівня його підготовленості.

Аналіз предметної галузі дозволяє зробити декілька висновків:

– задача розробки та використання тестів як засобів реалізації педагогічних вимірів є актуальною та повинна розглядатись у комплексі із іншими задачами підвищення рівня підготовки студентів та оптимізації навчального процесу;

– на даний час розроблено велика кількість математичних моделей та методів тестування та обробки його результатів;

– розглянуті методи обробки результатів тестування, включаючи класичні статистичні методи та методи сучасної торії тестування, дозволяють здійснювати аналіз результатів тестування, але не здатні надавати рекомендації з оптимізації навчального процесу та адаптації змісту тестів, складності завдань та розподілу часу на вивчення різних тем;

– адаптивні методи комп'ютерного тестування найчастіше використовуються для розв'язання проблеми адаптації процедури пред'явлення тестових завдань;

– у сучасних дослідженнях недостатньо уваги приділяється питанню адаптації навчальних курсів у залежності від результатів тестування, хоча деякі засоби (наприклад ШНМ) дозволяють її реалізувати.

Цей аналіз дозволяє сформулювати основні задачі дослідження:

– визначити специфіку функціонування штучних нейронних мереж та можливість їх використання при розробці педагогічних тестів зі змінною складністю завдань;

– розробити контрольно-вимірювальні матеріали для адаптивного тестування з певної дисципліни;

– удосконалити розроблені контрольно-вимірювальні матеріали за рахунок використання штучних нейронних мереж;

– розробити методику застосування отриманих результатів для адаптації системи навчання студентів.

2. Опис проведених теоретичних досліджень

Одним з шляхів підвищення ефективності комп'ютерних тестів є розробка та реалізація методів адаптивного тестування (АТ). Під АТ зазвичай розуміють комп'ютерну систему науково обґрунтованої перевірки та оцінки результатів тестування, яка має високу ефективність за рахунок реалізації можливості оптимізації процедур генерації, пред'явлення тв. оцінки результатів виконання адаптивних тестів [4].

Набагато частіше зустрічаються завдання, у яких можливі проміжні варіанти відповідей, або ж взагалі варіантів немає, питання є відкритим, і відповідь оцінюється в якій-небудь шкалі (наприклад,

п'ятибальна). Прикладом моделі, що враховує градацію правильних відповідей, є Partial Credit Model (PCM).

Безсумнівним плюсом застосування IRT моделей є можливість одержувати одночасно з оцінками компетенцій студентів обґрунтовані статистичні оцінки завдань, що може бути основою для поліпшення освітніх програм ВНЗ. Оцінка рівня підготовленості студентів не залежить від набору завдань, а неповнота даних (пропуск деяких комбінацій «випробуваний – завдання») не є критичною. Однак слід відзначити, що досліджувані завдання повинні бути гомогенними («одномірними»), тобто формувати та оцінювати тільки одну компетенцію. Основний шлях збільшення об'єктивності, результатів, а також розширення сфери застосування є використання в якості «сирих» балів не екзаменаційних оцінок, а оцінок, що отримуються під час проміжній атестації за виконання різних завдань. Цей шлях вимагає перегляду й значного розширення банку контрольних завдань по кожному предмету – кожне окремо взяте завдання повинне діагностувати тільки одну компетенцію. Так само це повинне знайти відображення в інформаційній системі ВНЗ – облік і зберігання цих оцінок.

Цю модель доцільно використовувати, коли:

– передбачається оцінювати частково вірні відповіді (наприклад тестові завдання з множинним вибором);

– завдання потребує послідовності кроків в його виконанні наприклад розв'язання задачі з математики). При цьому складність кожного кроку може бути різною.

АТ дозволяє підвищити ефективність педагогічних вимірів, скоротити кількість питань у тесті, знизити затрати часу та вартості тестування та підвищити точність тестування.

Для реалізації технології АТ необхідно [5]:

– визначити цілі реалізації АТ (навіщо потрібна адаптація у конкретному тесті?);

– визначити фактори, яки будуть враховуватись у якості вхідної інформації під час прийняття адаптаційних рішень (до чого буде адаптуватися тест?);

– які аспекти будуть змінюватися у процесі адаптації (що буде адаптуватися?);

– які механізми адаптації будуть використовуватися та як вони будуть реалізовані (як буде здійснюватися адаптація?).

Як правило, АТ базується на процедурі оптимізації складності завдань у залежності від припущень щодо рівня підготовки студентів. У найпростішому варіанті загальна процедура виглядає наступним чином. Студент отримує перше завдання тесту на основі початкових припущень. У разі його виконання рівень складності наступного завдання підвищується. У протилежному випадку рівень наступного питання знижується.

Процес тестування може бути закінчено, якщо, наприклад, студент не зміг виконати три завдання поспіль. У загальному випадку можуть бути використані більш складні процедури зупинення.

Тобто АТ навіть у простішої реалізації дозволяє динамічно змінювати кількість та складність тестових завдань конкретного студента. Більш складні алгоритми АТ дозволяють обирати наступне завдання тесту, приймаючи до уваги не тільки складність завдань, але також їх приналежність до певної теми дисципліни, форму представлення та інші фактори.

Таким чином будується індивідуальний тест для кожного студента. Різні студенти отримують різні тести, які розрізняються за складністю та складом завдань, проходячи простір тестів за різними траєкторіями (рис. 1).

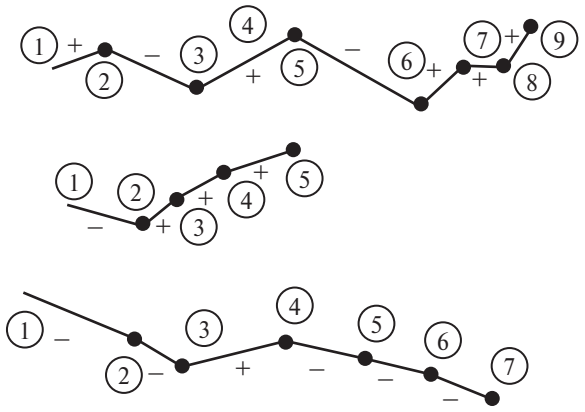


Рис. 1. Візуалізація індивідуальних траєкторій

На цьому рисунку показано три приклада траєкторій трьох студентів. Цифри позначають номера питань, знаки + та – відповідають вірним та невірним відповідям. Для припинення процедури тестування використовується просте правило: тестування закінчується, якщо студент вірно виконав три завдання поспіль, або зробив три помилки поспіль.

Різні алгоритми АТ можуть розрізнятися також стратегією тестування. Розрізняють двокрокові та багатокрокові стратегії. У рамках двокрокової стратегії на першому кроці всі студенти отримують однакові тести. За їх результатами вони розподіляються на осі змінній виміру. На другому кроці вмикається адаптивний режим, та здійснюється підсумкове адаптивне тестування.

Адаптивна навчаюча система (АНС) – це система, яка застосовує у процесі взаємодії зі студентом алгоритми адаптації до певних його характеристик [12]. Найчастіше це рівень знань, цілі, рівень підготовки, досвід роботи з засобами комп'ютерного представлення інформації (гіпермедіа, мультимедіа та ін.).

Використання нечіткої логіки – один з поширених напрямів інтелектуалізації систем контролю знань. Засоби нечіткої логіки використовуються

окремо, або у комбінації з іншими (наприклад з мережами Байєса).

Найбільш поширеним є перехід від завдання складності питань або відповідей у категоріях бінарної логіки до більш загальної та універсальної схеми оцінювання відповідей на основі функцій належності, яки визначаються в категоріях нечіткої логіки.

Використання нечіткої оцінки завдань та відповідей передбачає і використання нечітких правил виводу для формування гіпотези про рівень складності наступного завдання [6].

Тестування починається з припущення, що студент має певний рівень знань (S), який має нечіткий характер. Обчислюється група G=G(S). Студент отримує завдання відповідного рівня складності. При вірної відповіді збільшується кількість вірних відповідей, відсоток вірних відповідей та змінюється складність. На базі цього визначається новий рівень підготовки (S1) як функція поточного рівня підготовки S, відсотка вірних відповідей p, рівня складності завдання T, та часом на відповідь t :

Якщо кількість виданих завдань менше певної кількості, яка є обов'язковою для відповіді у межах одного рівня складності, то G=G1 та студент отримує нове завдання.

Якщо кількість критичних завдань перевищена, то рівень знань вважається рівним S1, та студент отримує відповідну оцінку R = R(G(S1)). Якщо змінилася група, то встановлюється S=S1, G=G1, обнуляється кількість завдань, та здійснюється повернення до видачі завдань.

В цієї процедурі АТ складність завдань змінюється у залежності від вірності відповідей студента, а поняття рівня підготовки, вірності відповіді, складності завдань та ін. є нечіткими та задаються за допомогою нечітких оцінок.

Марківські процеси дуже активно використовуються для реалізації алгоритмів АТ. Розглянемо функціонування АТ на основі моделі, яка використовує марковські процеси з дискретними станами та дискретним часом (ланцюг Маркова). Особливістю цього підходу є те, що складність завдань визначається на основі граничних розподілів ймовірностей перебування у станах, отриманих за допомогою матриць ймовірностей переходу. Тут оцінка враховує всю історію виконання тестових завдань, яка містить розподіл успішних та неуспішних виконань завдань та їх порядок, а також час виконання тестових завдань. Ймовірності переходу між станами є параметрами моделі.

Мережі Петрі широко використовуються не тільки для реалізації АТ, але і для моделювання прогресу студентів під час вивчення деякого курсу у адаптивних навчаючих системах [7]. Моделювання здійснюється на рівні подій. Визначається, яки дії відбуваються у системі, які стани передували цім діям та як і у яких станах опиниться система після виконання дії. Виконання моделі подій описує поведінку системи. Аналіз результатів виконання може розповісти про те, у яких станах перебувала, або не перебувала система, яки стани не є досяжними. Під час тестування алгоритм обирає переходи до завдань, які мають більш високий або більш низький рівень складності.

Рівень знань є найбільш важливою характеристикою студента (користувача) та найбільше часто використовуваним джерелом інформації для адаптації. Рівень знань є змінною величиною для кожного конкретного студента. Тому будь-яка система, що адаптується до знань користувача, повинна мати його модель, фіксувати зміни рівня знань і відповідним чином корегувати цю модель.

Найпоширенішою в даний час є оверлейна модель студента, яка заснована на структурній моделі дисципліни, що вивчається. Структурна модель зазвичай представляється у вигляді семантичної мережі, що відбиває взаємозв'язки між окремими елементами знань із цієї дисципліни (поняттями, діями, завданнями й ін.). Для кожного елемента знань, представленого в мережі модель зберігає деяке значення, що представляє собою оцінку знання студентом даного елемента. Іншим підходом до побудови моделі користувача є поділ студентів на певні категорії й наступне використання для адаптації інформації не про окремого студента, а узагальненої інформації про категорію, до якої він віднесений. У цьому випадку модель виходить трохи більш спрощеної й надає менше можливостей для адаптації дій системи.

На основі аналізу зазначених параметрів моделі користувача адаптивний навчальний курс повинен дозволяти змінювати зміст матеріалу, що пропонується для вивчення, його форму представлення, послідовність проходження за курсом і можливості навігації на сторінках курсу (якщо це дисципліна, що представлена в гіпермедійному форматі). Адаптація змісту може реалізовуватися шляхом представлення навчального матеріалу з різним ступенем деталізації, з урахуванням рівня знань про суміжні теми та поняття, а також моделі користувача.

Багатокрокові стратегії поділяються на фіксовано-розгалужені та змінно-розгалужені. У фіксовано-розгалужених стратегіях для всіх студентів використовується однакові тестові набори завдань з фіксованим розташуванням на осі складності. Але шлях кожного студента є різним. Як правило, всі завдання на цієї осі розташовані на однієї відстані, або шаг зменшується відповідно складності. Це дає можливість змінювати швидкість тестування у залежності від стану втомленості студентів.

У змінно-розгалужених тестах завдання обираються з бази завдань на основі певних алгоритмів,

яки прогнозують оптимальну складність наступного завдання. З цих окремих завдань складається адаптивний тест. Ця стратегія реалізує покрокову переоцінку рівня знань студентів, яка здійснюється після кожного виконання завдання тесту.

Вибір початкових оцінок для входу в процедуру АТ може основуватись на результатах попереднього тестування, на результатах тестів, які проводилися протягом певного періоду, або на основі моделі суб'єкта навчання, якщо вона використовується у системі.

Необхідно відзначити, що коректність роботи ШНМ прямо залежить від правильності вибору моделі ШНМ і алгоритму навчання для поставленого завдання, що важливо. Аналіз можливостей та властивостей різних ШНМ дозволяє зробити висновок, що з усіх моделей ШНМ найкращою для вирішення задач адаптивного тестування є перцептрон Розенблатта [8].

Для подальшого дослідження необхідно розглянути класифікацію способів навчання ШНМ і визначитися з алгоритмом навчання ШНМ. Якщо алгоритм буде обраний неправильно, то ШНМ не зможе коректно навчитися на результатах тестування, а відповідно, не можна буде говорити ні про яке поліпшення адаптивного тесту.

Алгоритм передбачає виконання наступної послідовності дій:

– підготовляється навчальна вибірка вхідних та вихідних значень;

– на початку роботи ШНМ усім зв'язкам мережі присвоюються невеликі випадкові значення;

– з навчальної вибірки береться поточний приклад і його вхідні параметри, (що у сукупності складають вектор вхідних сигналів) подаються на вхідні синапси начальної ШНМ. Зазвичай кожний вхідний параметр прикладу подається на один певний вхідний синапс;

– ШНМ робить задану кількість епох функціонування, вектор вхідних сигналів поширюється по зв'язках між нейронами (пряме функціонування);

– визначення дійсного виходу мережі, виконання її перерахування;

– обчислення різниці між дійсним і бажаним виходом мережі. Чим менше ця різниця, тем краще розпізнаний приклад, і тем ближче відповідь до необхідного. Якщо різниця рівна 0, то ніяких дій не вживається;

– зміна ваги мережі для мінімізації помилки (у цьому і є зміст навчання);

– повторення пунктів доти, поки загальна помилка мережі не буде мінімальною. Прохід по всіх прикладах навчальної вибірки з першого до останнього вважається одним циклом навчання.

Як можна бачити з опису алгоритму навчання ШНМ, цей варіант найкраще підходить для аналізу комп'ютерних інструктивних матеріалів на предмет їх відповідності заявленій тематиці, тобто іноді тестові завдання по певній темі не завжди підходять для визначення рівня знань саме для цієї теми. Для рішення цієї проблеми необхідно:

– описати банк термінів, які будуть розподілятися по класах, тобто потрібно виділити ознаки, по яких терміни будуть поєднуватися в класи. Той самий термін може належати різним класам. Для цього необхідна строга ієрархія термінів;

– після того, як будуть виділені необхідні ознаки термінів для кожного конкретного класу з наступним їхнім розподілом, необхідно навчити мережу на готових прикладах, у яких заздалегідь відомо до якого класу буде віднесений кожен термін;

– після навчання ШНМ (добору коефіцієнтів) можна пред'являти їй завдання. У процесі кластеризації термінів по класах ШНМ буде виносити рішення про відповідність або невідповідність даного завдання заявленій темі.

При навчанні без вчителя ШНМ не має потреби в цільовому векторі для виходів і, отже, не потребує порівнянь із визначеними ідеальними відповідями. Навчальна множина складається лише із вхідних векторів. Навчальний алгоритм налаштовує ваги мережі так, щоб виходили погоджені вихідні вектори, тобто щоб пред'явлення достатнє близьких вхідних векторів давало однакові виходи. Процес навчання виділяє статистичні властивості навчальної множини та групує подібні вектори в класи. Пред'явлення на вхід вектора з даного класу дасть певний вихідний вектор, але до навчання неможливо передбачити, який вихід буде проводитися даним класом вхідних векторів. Отже, виходи подібної мережі повинні трансформуватися в деяку зрозумілу форму, обумовлену процесом навчання.

При змішанім навчанні частина ваг визначається за допомогою навчання із учителем, а інша частина отримується за допомогою самонавчання.

3. Опис методу адаптивного тестування із застосуванням логічних мереж

3.1. Розробка контрольно-вимірювальних матеріалів

Для якісного та вірогідного виміру знань студентів необхідно діяти згідно такої процедури:відібрати зміст навчання, для цього розглядається програма обраного (необхідного) курсу;розробити систему тестів по обраній дисципліні (створити базу завдань); розробити необхідне програмне забезпечення для здійснення тестування; побудувати ШНМ та навчити її на результатах попередніх тестувань; провести тестування; виконати обробку та інтерпретацію отриманих результатів.

Згідно цієї процедури, на першому етапі для проведення дослідження було обрано курс «Інформа-

ційні забезпечення професійної діяльності» (ІТПД), який курс викладається студентам спеціальностей «Правознавство» та «Правоохоронна діяльність» Харківського національного університету внутрішніх справ (ХНУВС) та містить одинадцять тем. Загальний обсяг бази результатів тестування складає біля 300 студентів, оскільки в умовах дистанційного навчання тестування проходить кожен студент. Це достатньо велика кількість для проведення нашого дослідження. Для навчання та тестування студентів використовується система Moodle.

Загальний обсяг бази – 100 тестових завдань. Для реалізації тестування питання проходили попередню обробку – розподіл по темах та оцінку складності.

Для попередньої обробки тестових завдань було розраховано кілька величин: ємність відповіді на завдання, імовірність правильної відповіді, імовірність «сліпого» угадування правильної відповіді, складність завдання.

Ємність відповіді на завдання *I*. Дана характеристика показує кількість інформації, що міститься у відповіді на завдання. Будемо вважати, що для студента всі відповіді рівнозначні. Тоді для того, щоб оцінити величину *I,* необхідно буде скористатися формулою для рівноймовірнісних подій:

$$I = Log_2 M, \qquad (1)$$

де M – кількість можливих відповідей на завдання.

Імовірність правильної відповіді на завдання *P*. Ця характеристика відображає знання групи студентів з розділу, якому відповідає завдання визначається як відношення кількості вірних відповідей до загальної кількості відповідей на це завдання.

Ймовірність «сліпого» вгадування правильної відповіді P_y . Цей показник характеризує подію тільки в тому випадку, коли студент не читає текст завдання або не розуміє його. У цьому випадку вибір правильної відповіді носить випадковий характер (вгадування). У даної характеристики немає єдиної формули, тому для кожного виду завдання вона буде розраховуватися окремо.

Складність завдання *Q* враховує статистику відповідей студентів та їх знання завдання. Для визначення рівня складності завдання будемо застосовувати формулу, що відбиває невизначеність стану об'єкту:

$$Q = -\sum_{i=1}^{M} P\left(X_j\right) Log_2 P\left(X_j\right), \qquad (2)$$

де M — кількість можливих відповідей на завдання; $P(X_j)$ — ймовірність j-ї відповіді.

Початкові умови тестування зазвичай завдаються таким чином, що спочатку кожен з варіантів відповідей на завдання рівноймовірнісний. Для того, щоб більш точно визначити рівень складності завдань, необхідно щоб усі вони пройшли експериментальну перевірку. По завершенню цієї перевірки буде отримана інформація про необхідні статистичні характеристики завдань.

Складність *Q* також може визначатися на основі експертної оцінки. Зазвичай у ролі експерта виступає викладач, що часто є виправданим у випадку невеликої кількості завдань, хоча може привнести певний суб'єктивізм.

3.2. Процедура адаптивного тестування знань

Для реалізації адаптивного тестування було запропоновано процедуру, яка реалізує багатокрокову варіюючу стратегію. Ця стратегія під час тестування здійснює вибір наступного питання з банку питань на базі результату відповіді на два попередні з урахуванням його рівня складності.

Тестові завдання адаптивного тесту передбачаються закритої форми з вибором одного з 4-х запропонованих варіантів. На кожному кроці тестування по кожному рівню складності студенту дається два завдання, і за результатами відповідей на них визначається рівень складності для наступних завдань. Така кількість завдань (два) дозволяють більш адекватно оцінювати рівень знань, ніж одне завдання, і в той же час не дає великої кількості комбінацій варіантів відповідей, як у випадку трьох та більшої кількості завдань.

Алгоритм можна описати наступним чином. У тесті є m рівнів складності завдань (у тестовому випадку їх 3). Вводиться коефіцієнт $K_i = 100/m$. На другому кроці позначимо поточний рівень знань студента як t, t_n – нижній рівень знань, t_v – верхній рівень знань. Усі рівні знань будуть вимірятися від 0 до 100 (0 — немає знань, 100 — абсолютні знання).

Спочатку будемо вважати, що студент має середній рівень підготовки. Тому встановимо $t = 50$, $t_n = 0$, $t_v = 100$. Обчислюємо поточний рівень складності: $tt = t/K_i$.

На наступному кроці пред'явимо студенту два завдання складності *tt*, при цьому стежимо за кількістю правильних відповідей k_{pr}.

Перераховуємо рівень знань із урахуванням відповідей на два завдання:

Якщо $k_{pr} = 2$, то $t_n = t$; $t_v = t_v + 0{,}5t$. Якщо $t_v > 100$, то $t_v = 100$.

Якщо $k_{pr} = 1$, то $t_n = t_n/4$; $t_v = t_v + 0{,}1t$. Якщо $t_v > 100$, то t_v =v100.

Якщо $k_{pr} = 0$, то $t_n = t_n/2$; $t_v = t$.

Обчислюємо $t_v = (t_n + t_v)/2$.

Якщо $|t - t_v| > 0$, то $t = t_1$.

Якщо досягнуто критичний рівень кількості завдань або балів за завдання, то рівень знань рівний t_1. Вихід.

У противному випадку перейти до другого кроку.

Отриманий результат перераховується згідно обраної шкали оцінок.

Результати тестування у бінарному вигляді представляються у вигляді таблиці. Дані таблиці є вхідною інформацією для другого кроку адаптації – оцінки складності тестових запитань.

Систем вважає необхідним підвищити рівень складності завдань з номерами 2, 4, 14 та 20, а також знизити рівень складності для завдання № 25. Для решти завдань рівень змінювати не потрібно.

Ця інформація повинна бути розглянута експертом (викладачем) для аналізу та прийняття рішення.

Наступний етап обробки результатів тестування – надання рекомендації щодо зміни часу на окремі теми дисципліни.

У результаті роботи ШНМ педагогам-розроблювачам комп'ютерних навчальних матеріалів надається інформація про номер завдання, його початковий рівень складності та рекомендація щодо можливої зміни рівня складності (підвищити, залишити без змін, знизити). Результати обробки вхідних даних представлено на рис. 2.

Дійсно, якщо в ході адаптивного тестування виявлене загальне нерозуміння студентами однієї з тем навчальної дисципліни, то при наступному навчанні педагог повинен затратити більше навчального часу й приділити більшої увагу вивченню цієї теми. Практичне впровадження системи коректувань змісту й методики освіти залежно від рівня підготовленості студентів, виявленого в ході адаптивного тестування, стикається із проблемами обробки й обліку більших інформаційних потоків, рішення яких також можливо на основі використання ШНМ.

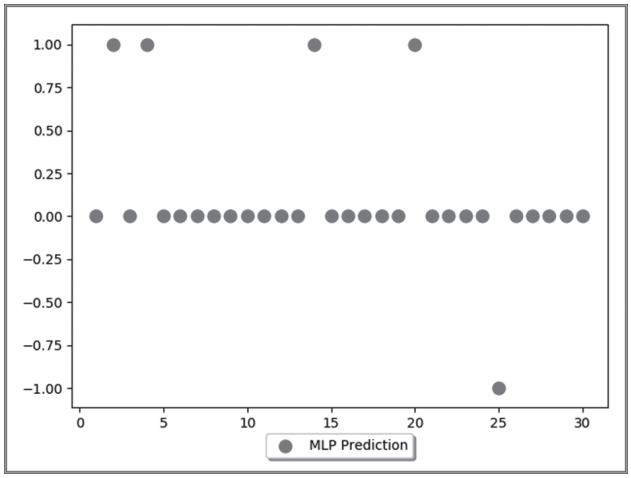


Рис. 2. Результати роботи ШНМ

На підставі проведення тестів можна стверджувати, що створений прототип виконує всі функції, що зазначені у вихідних даних на розробку й відповідає всім вимогам, що пред'явлені до нього.

Визначено ряд обмежень логічного характеру, що визначають як перехід зі стану в стан, видалення станів, так і обмеження кількості переходів (тестування не може проводитися нескінченно й питання не повинні повторюватися). Однак, при невеликій різниці між зазначеними передбачуваними й дійсними властивостями групи випробуваних застосування зазначеної методики дає позитивний ефект, із чого можна зробити висновок, що застосування запропонованого методу є ефективним для будь-яких обсягів і інших властивостей, що визначають тестування як процес.

Висновки

Ідея адаптивного тестування, заснована на блоках питань, має безпосереднє відношення до одного із самих розповсюджених форматів багатоступінчастого тестування, при якому суб'єкт навчання проходить через послідовність тестів, рухаючись убік більш складних питань при успішних відповідях або до більш простих, якщо його відповіді невірні. При цьому перехід від питання до питання відбувається за певними правилами. Автоматизація у практиці тестування надає можливість статистичної оцінки знань на кожному кроці тестування.

Очікуваний результат по використанню розроблених моделей стосовно до 3-х параметричної моделі тестування має бути досягнутий при апроксимації функції успіху, що припускає визначення уточнених значень дискримінаторів завдань шляхом рішення завдання оптимізації при використанні відомих алгоритмів. Це пов'язане насамперед з тим, що застосовані алгоритми оптимізації є універсальними, а рішення виниклої проблеми вимагає додаткового дослідження метою якого є часткове заміщення цих алгоритмів операційними методами, що використовують як властивість поставленого завдання тестування так і іншої апріорної інформації.

Різниця в рівнях складності основних і додаткових питань і запропонований зв'язок між основними питаннями і гілками додаткових питань дозволяє в процесі тестування як мінімізувати кількість необхідних відповідей суб'єкта навчання для визначення рівня його знань, так і суттєво поліпшити адаптаційні властивості тестування.

Для розв'язання цієї проблеми було запропоновано використовувати ШНМ з тією ж структурою. Входом мережі є вектор оцінок студентів з певної теми. Вихідна реакція – рекомендація щодо можливих змін у обсягу часу на вивчення цієї теми (збільшити, не змінювати, знизити).

Після попереднього навчання на вхід системи було подано вхідні сигнали та отримана рекомендація «не змінювати» обсяг часу на вивчення конкретних те за досліджуваними наборами тестів з дисциплін.

Для обробки результатів тесту та врахування додаткових чинників, яки можуть впливати на результати тестування, можуть використовуватися також

статистичні методи. Ці методи дозволяють отримати оцінки додаткових показників тестів:

– відсоток вірних або невірних відповідей;

– відносну успішність групи студентів по одній темі стосовно всіх інших тем курсу;

– розподіл знань студентів по всім темам;

– середня навченість групи та ін.

Список літератури:

[1] Baker R. “Educational data mining and learning analytics.” The Cambridge handbook of the learning sciences, 2019, p. 274.

[2] Rzheutska S. “Experience of using clustering methods for analyzing the results of distance learning.” Informatization of engineering education: materials of international scientific-practical conf., No. 56, 2016, pp. 617 – 620.

[3] I.Shubin, V.Skovorodnikova, A.Kozyriev, Mining Methods for Adaptation Metricsin E-Learning, in: Proceedings of the 3rd International Conference Computational Linguistics and Intelligent Systems CoLInS 2019,NTU “KhPI”, Kharkiv, (2019) 288–300.

[4] Chetverikov G.G., Vechirska I.D., Tanyanskiy S.S. The methods of algebra finite predicates in the intellectual system of complex calculations of telecommunication companies // International Conference Proceedings Crimean Microwave and Telecommunication Technology (CriMiCo).-2014, 6959425. - pp. 346-347.

[5] Sharonova, N., Kyrychenko, I., Tereshchenko, G. Application of big data methods in E-learning systems CEUR Workshop Proceedingsthis link is disabled, 2021, 2870, pp. 1302–1311

[6] Gruzdo, I., Kyrychenko, I., Tereshchenko, G., Cherednichenko, O. Applıcatıon of paragraphs vectors model for semantıc text analysıs CEUR Workshop Proceedingsthis 2020, 2604, pp. 283–293

[7] G. Bezhanishvili “Locally finite varieties” Algebra Universalis, Vol. 46, no. 4, 2001, pp. 531–548.

[8] Chetverikov G., Puzik O., Vechirska I. Multiple-valued structures of intellectual systems //Proceedings of the with Internations Computer Sciences and Information Technologies (CSIT). 2016, 7589907. -pp. 204-207.

[9] A Guide to the SCRUM BODY OF KNOWLEDGE (SBOK™ GUIDE) URL: https://www.scrumstudy.com/SBOK/SCRUMstudy-SBOK-Guide-3rd.pdf.

Надійшла до редколегії 19.05.2021

ПРАВИЛА
оформлення рукописів для авторів науково-технічного журналу «БІОНІКА ІНТЕЛЕКТУ»

Науково-технічний журнал «Біоніка інтелекту» приймає до друку написані спеціально для нього оригінальні рукописи, які раніше ніде не друкувались. Структура рукопису повинна бути такою: індекс УДК, відомості про авторів, заголовок, анотації (на трьох мовах), ключові слова, вступ, основний текст статті, висновки, список використаної літератури, резюме.

Відповідно до Постанови ВАК України від 15.01.2003 №7-05/1 (Бюлетень ВАК, №1, 2003, с. 2), стаття повинна мати такі необхідні елементи: постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями; аналіз останніх досліджень і публікацій і виділення не вирішених раніше частин загальної проблеми в даній області; формулювання цілей та завдань дослідження; виклад основного матеріалу досліджень з повним обґрунтуванням отриманих наукових результатів; висновки з даного дослідження та перспективи подальших досліджень у даному напрямку.

Статті мають бути виконані в редакторі Microsoft Word. Формат сторінки – А4 (210×297 мм), поля: верхнє – 25 мм, нижнє – 20 мм, ліве, праве – 17 мм. Кількість колонок – 2, з інтервалом між ними 5 мм, основний шрифт Times New Roman, кегль основного тексту – 10 пунктів, міжрядковий інтервал – множник (1,1), абзацний відступ – 6 мм. Обсяг рукопису – від 6 до 12 сторінок (мови: українська, англійська, російська та мовою орігінала).

УДК друкується з першого рядка, без відступів, вирівнювання по лівому краю.

ПІБ автора (-ів), назва статті, назва та адреса учбового закладу необхідно надати повністю російською, українською та англійською мовами.

Назва статті друкується прописними літерами; шрифт прямий, напівжирний, кегль 12.

Назви розділів нумерують арабськими цифрами, виділяють жирним шрифтом. Відступи для назви статті, ініціалів та прізвищ авторів, відомостей про авторів, назв розділів, вступу та висновків, списку літератури: зверху – 6 пт, знизу – 3 пт.

Анотації (мовою статті, абзац 6–12 рядків, кегль 9) розміщують на початку статті, в ній має бути розміщена інформація про очікувані результати описаних досліджень (на трьох мовах).

Ключові слова (4–10 слів з тексту статті, які з точки зору інформаційного пошуку несуть змістовне навантаження) наводять мовою рукопису, через кому в називному відмінку, кегль 9.

Рисунки та *таблиці* (чорно-білі, контрастні) розміщуються у тексті після першого посилання у вигляді окремих об’єктів і нумерують арабськими цифрами наскрізною нумерацією за наявності більше ніж одного об’єкта. Невеликі схеми, що складаються з 3–4 елементів виконують, використовуючи вставку об’єкта Рисунок Microsoft Word. Більш складні виконують у графічних редакторах у вигляді чорно-білих графічних файлів форматів .tif, .jpg, .wmf, .cdr із розділенням 300 dpi. Рисунки мають міститися у текстовому файлі й обов’язково подаватися окремими файлами з відповідними назвами (наприклад, рис1.jpg).

Усі елементи рисунка, включаючи написи, повинні бути згруповані. Усі написи в рисунках і таблицях мають бути виконані шрифтом Times New Roman, кегль у рисунках – 10, у таблицях – 9.

Рисунок повинен мати центрований підпис (поза рисунком), шрифт 9, відступи зверху і знизу по 6 пт. Ширина рисунка має відповідати ширині колонки (або ширині сторінки).

Формули, символи, змінні повинні бути набрані в **редакторі формул MathType**. Формули розміщують посередині рядка й нумерують за наявності посилань на них у рукописі. Шрифт – Times New Roman. Висота змінної – 10 пунктів, великих і малих індексів – 8 пт, основний математичний символ – 12 (10) пт. Змінні, позначені латинськими літерами, набирають курсивом, грецькі літери, скорочення російських слів і цифри – прямим написанням. Змінні, які є в тексті, також набирають у редакторі формул.

Список літератури вміщує опубліковані джерела, на які є посилання в тексті, укладені у квадратні дужки, друкують без абзацного відступу, кегль 9 пт, відступ зверху – 6 пт.

Після списку літератури з відступом зверху 6 пт зазначають *дату подання статті до редколегії*. Число та місяць задають двозначними числами через крапку. Розмір шрифта – 9 пт, курсив, вирівнювання по правому краю.

Резюме (Times New Roman, кегль — 10 пунктів,) подають англійською мовою: обсяг резюме до 2000 знаків (бажаний переклад). *Структура резюме:* **Background, Materials and methods, Results, Conclusion.**

Разом із рукописом (на аркушах білого паперу формату А4 щільністю 80-90 г/м^2, надрукований на лазерному принтері) необхідно подати такі документи:

1. Заяву, яку повинні підписати всі автори.
2. Акт експертизи про можливість опублікування матеріалів у відкритому друці (якщо потрібно).
3. Рецензію, підписану доктором чи кандидатом наук.
4. Відомості про авторів.
5. Електронний варіант рукопису, резюме та відомостей про авторів.
6. Зробити оплату публікації.

Необхідно також зазначити один з наступних тематичних розділів, якому відповідає рукопис:

1. Теоретичні основи інформатики та кібернетики. Теорія інтелекту.
2. Математичне моделювання. Системний аналіз. Прийняття рішень.
3. Інтелектуальна обробка інформації. Розпізнавання образів.
4. Інформаційні технології та програмно-технічні комплекси.
5. Структурна, прикладна та математична лінгвістика.
6. Дискусійні повідомлення.

СОДЕРЖАНИЕ

Наукове видання

БІОНІКА ІНТЕЛЕКТУ

інформація, мова, інтелект

Науково-технічний журнал

№ 1 (96)

2021

Головний редактор — *Г. Г. Четвериков*

Відповідальний редактор — *І.В. Кириченко*

Комп'ютерна верстка — *О. Б. Ісаєва*

Рекомендовано Вченою Радою
Харківського національного університету радіоелектроніки
(протокол № 6 от 02.07.2021)

Адреса редакції:

Україна, 61166, Харків-166, просп. Науки, 14,
Харківський національний університет радіоелектроніки, к. 127
тел. 702-14-77, факс 702-10-13,
e-mail: bionics@nure.ua

Підписано до друку 02.07.2021. Формат $60 \times 84\ ^1/_8$. Друк ризографічний.
Папір офсетний. Гарнітура Newton. Умов. друк. арк. 15,4. Обл.-вид. арк. 15,0.
Тираж 100 прим.

Віддруковано в редакційно-видавничому відділі ХНУРЕ
61166, Харків, просп. Науки, 14.