УДК 519.767.6
DOI 10.33111/nfmte.2020.095

MACHINE LEARNING APPROACH OF ANALYSIS OF EMOTIONAL POLARITY OF ELECTRONIC SOCIAL MEDIA

Vasily Derbentsev

Kyiv National Economic University named after Vadym Hetman
54/1 Peremogy Ave., Kyiv, 03680, Ukraine
ORCID: 0000-0002-8988-2526, E-mail: derbv@kneu.edu.ua

Vitalii Bezkorovainyi

Kyiv National Economic University named after Vadym Hetman
54/1 Peremogy Ave., Kyiv, 03680, Ukraine
ORCID: 0000-0002-4998-8385, E-mail: retal.vs@kneu.edu.ua

Renat Akhmedov

Kyiv National Economic University named after Vadym Hetman
54/1 Peremogy Ave., Kyiv, 03680, Ukraine
ORCID: 0000-0002-3084-4672, E-mail: akhmedov.kneu@gmail.com

This paper proposes a new approach to evaluating the emotional polarity (or Sentiment Analysis) of electronic social media texts. For this purpose both conventional Machine Learning (Logistic Regression and Support Vector Machine), and Deep Neural Networks approaches (Fully Connected and Convolutional Neural Networks) were used. As vector representations of words, we used both the frequency-based and pre-trained words embeddings Word2vec and GloVe (with embedding dimensions of size 100 and 300).
For the selected English-language IMDb Movie Reviews dataset the classification accuracy using the Logistic Regression model was 87%, the Support Vector Machine – 87.5%, the Fully Connected Neural Network – 88%, and the Convolutional Network – 90%. The accuracy of the proposed models is a quite acceptable for practical use-cases and is not inferior to cutting-edge Natural Language Processing solutions in the field of Sentiment Analysis, which opens up good prospects for further research.

Keywords: *emotional polarity detection, sentiment analysis, electronic social media, machine learning, deep learning*

ПІДХІД НА ОСНОВІ МАШИННОГО НАВЧАННЯ ДО АНАЛІЗУ ЕМОЦІЙНОЇ ПОЛЯРНОСТІ ЕЛЕКТРОННИХ СОЦІАЛЬНИХ МЕДІА

В. Д. Дербенцев

Державний вищий навчальний заклад
«Київський національний економічний університет
імені Вадима Гетьмана»
просп. Перемоги, 54/1, м. Київ, 03680, Україна
ORCID: 0000-0002-8988-2526, E-mail: derbv@kneu.edu.ua

В. С. Безкоровайний

Державний вищий навчальний заклад
«Київський національний економічний університет
імені Вадима Гетьмана»
просп. Перемоги, 54/1, м. Київ, 03680, Україна
ORCID: 0000-0002-4998-8385, E-mail: retal.vs@kneu.edu.ua

Р. Р. Ахмедов

Державний вищий навчальний заклад
«Київський національний економічний університет
імені Вадима Гетьмана»
просп. Перемоги, 54/1, м. Київ, 03680, Україна
ORCID: 0000-0002-3084-4672, E-mail: akhmedov.kneu@gmail.com

У статті пропонується новий підхід до оцінки емоційної полярності (або аналізу настроїв) електронних текстів у соціальних мережах. Для цього використовувалися як класичні методи машинного навчання (логістична регресія та метод опорних векторів), так і інструментарій глибоких нейронних мереж (повнозв'язні та згорткові нейромережі). Векторне представлення ґрунтувалось на частотних та попередньо навчених вкладеннях слів Word2vec і GloVe (з розмірами вкладення 100 і 300).
Для вибраного англомовного набору даних IMDb Movie Reviews точність класифікації за допомогою моделі логістичної регресії становила 87%, машини опорних векторів – 87,5%, повнозв'язної нейронної мережі – 88% і згорткової мережі – 90%. Точність запропонованих моделей є цілком прийнятною для практичних ситуацій і не поступається передовим рішенням у сфері обробки природньої мови за напрямом аналізу настроїв, що відкриває обнадійливі перспективи для подальших досліджень.

Ключові слова: *виявлення емоційної полярності, аналіз настроїв, електронні соціальні медіа, машинне навчання, глибинне навчання*

JEL Classification: C45, C53, C89

1. Introduction

Recent two decades have been characterized by the rapid development of social media which caused a revolution in communication technologies in modern society. Various electronic platforms (such as social networks, blogs, forums, bulletin boards, podcasts, content communities, photo and video hosting) have created a fundamentally new ecosystem of communication and have become an integral part of almost all spheres of human activity.

According to the We Are Social agency, which recently has published an annual report on the state and prospects for the development of the world digitization [1], the number of active Internet users in the world at the beginning of 2020 was about 4.5 billion, and the number of active users of various social networks has almost doubled from 2 billion in 2015 and is approaching to 4 billion.

Therefore, a significant number of the world's leading companies began to rebuild their business models using the possibilities of modern communication tools through social networks and other electronic platforms.

In addition, due to the constant increase in the volume of information, traditional communication technologies began to lose their effectiveness. In today's business environment, the ability to quickly monitor and control the public opinion is becoming one of the key success factors.

Thus, companies are interested in interacting with their consumers through social media, as this significantly increases brand awareness and audience loyalty. They can also provide support and feedback to existing and potential customers, contractors, partners, etc.

On the other hand, consumers are increasingly using social media to make purchasing decisions. They look for reviews, analyse the opinions of famous bloggers. Moreover, consumers frequently make decisions based on how often they interact with a brand on social media.

Users leave in publications reviews about goods and services, form trends, share their preferences and interests. For this reason, large companies are showing growing interest in analysing huge amounts of social media data.

These trends, on the one hand, and the resent achievements in the field of Artificial Intelligence (AI), first of all Machine Learning (ML) and Deep Neural Networks (DNN), the accumulation of significant volumes of training data, the increase in the power of computing systems (in particular, the use of multi-core CPUs and GPUs for processing large amounts of information), on the other hand, gave a new impetus to the development of automated Natural Language Processing (NLP) systems [2, 3].

NLP is an interdisciplinary field, which combine Computer Science, AI, and Computational Linguistics. Its purpose is to develop approaches (methods, models and algorithms) for automated computer systems of processing natural languages.

Moreover, natural language is a weakly structured cognitive system. That's why, methods of the Theory of Complex Systems and Semantic Knowledge Networks recently have been actively used for the description and modelling different aspects of Natural Language Processing and Understanding [4, 5].

The range of applied tasks which solved using NLP is quite wide: creation of automated machine translation systems, chatbots and other dialog systems; information search; facts (knowledge) automatic extraction from electronic documents; automated clustering and classification of texts, and so on [2, 6, 7].

Text classification is one of the important directions in machine processing of natural language. Its purpose is to determine whether a text document belongs to one of the predefined classes. The task of understanding text written in natural language involves the identification of explicit or implicit features of individual elements, such as words, phrases, sentences and paragraphs, in order to highlight certain text properties.

Important tasks for many domains are: to assess trust of social network users; determining the presence of a person or bot as a conversation partner; fake news detection. These tasks in many cases can be reduced to the classification problems [8-10].

One of the key tasks of NLP is the analysis of emotion tonality or Sentiment Analysis (SA) of text documents. This is an attempt to "extract" subjective characteristics from the text: emotions, sarcasm, embarrassment, suspicion, etc.

SA is a class of NLP methods for automatic selection (extraction) of emotionally coloured vocabulary in the texts and polar assessment (usually negative, neutral, and positive) of authors (their opinions) in relation to objects mentioned in the text, which is a partial case of the classification problem [6, 7].

SA is widely used in such domain as [6, 7, 11, 12]:

– Marketing research: in particular, monitoring the loyalty of consumer audience to various brands, goods, services; competitor activity analysis; product analytics and customer support; searching for insights and trends in industries; studying the opinions of employees, etc.

– Sociology: evaluation of the polarities of opinions regarding socially important topics.

– Political science: assessment of the attitude of voters to candidates.

– Finance: financial time series forecasting based on the semantics (sentiment analysis) of website news feeds.

Therefore, the development of automated systems for analysing the tonality of texts in social media, in particular, with the use of modern technologies of artificial intelligence (machine and deep learning), becomes especially relevant.

The main objective of our research is to compare the effectiveness of conventional ML and DNNs approaches for Sentiment Analysis of messages in social media. For this purpose, we used four classifiers: Logistic Regression (LR), Support Vector Machine (SVM), Fully Connected Neural Network (FCN) and Convolutional Neural Network (CNN). All classifiers were examined on IMDb Large Movie Reviews labelled dataset with statistical (frequency) words vectors representations TF-IDF as well as pre-trained words embeddings Word2vec.

2. Literature review

Common way for conducting SA is based on a lexical or rule-based approach. This approach consists in using a dictionary composed of words from the studied corpus of text with their pre-determined emotional polarity.

Final polarity of the document (sentence, feedback, message, etc.) is determined on the basis of this dictionary by using certain rules (including ML tools). This approach to the analysis of social media was used, in particular, in [13-15].

Recently have been published several reviews devoted to SA of social media [16-18]. Thus, Kumar and Jaiswal [16] presented a systematic review of papers on Soft Computing techniques for SA on Twitter.

Drus and Khalid [17] published a review of SA usage in social media that explored the social media platforms, various SA techniques, and their application. This review contains studies published between 2014 to 2019 from the most popular databases (ACM, Emerald Insight, IEEE Xplore, Science Direct, and Scopus), and covers areas such as world events, healthcare, politics, finance and business. The result obtained shows that in most of the articles the opinion-lexicon method to SA in social media was used.

D'Andrea at al. [18] presented an overview of SA classification approaches (rule-based, ML, hybrid) with respect to feature extraction techniques, advantages and limitations. Authors noted that the main challenge in applying SA for social media is to overcome the ambiguity that is actually a particular problem since it is not easily make use of coreference information.

It should be noted that the main advantage of rule-based approach is that there is not necessary to use a labelled dataset and it becomes possible to use an unsupervised learning technique. Its disadvantage is relatively low classification accuracy. In addition, the creation of tonal dictionaries is quite time-consuming and expensive. It is also necessary to take into account that tonal dictionaries are not universal and depend on the research domain.

Authors survey [19] investigated various approaches and techniques used in opinion summarization (visualization, aspect based, query-focused, real time, update summarization), and discussed the current opinion summarization challenges for social media.

Therefore, in recent years the dominant approach to solving SA problems is the use of both classical ML tools (Naive Bayes (NB), Decision Trees (DT), LR, SVM) and models based on DNNs using pre-trained word embeddings [20-22].

Lately Jain and Pamula [20] published report on ML applications for consumer sentiment analysis in the domain of hospitality and tourism. This report based on 68 research papers, which were focused on sentiment classification, predictive recommendation decisions, and fake reviews detection. They have shown a systematic literature review to compare, analyse, explore, and understand ML possibilities to find research gaps and the future research directions.

The article [21] presents the results of a comparative analysis of the effectiveness of the applying several powerful ML algorithms (Random Forest (RF), SVM, and Hybrid Approach) for SA of Amazon customer's reviews. According to the results obtained, the Hybrid Approach, based on the combination of RF and SVM, produced better results in the basic metrics (Accuracy, Precision, Recall, F1-score).

Authors of the article [22] combined the Encoder-Decoder framework with the LSTM-CNN model for Twitter SA. In the proposed model, LSTM (Long Short-Term Memory recurrent neural network) used for remembering forward information of the text sequence, and CNN performed catching and learning of local information patterns. The reconstruction of input matrix by decoder made the features learning in CNN more efficient and, therefore, the higher accuracy rate was achieved.

Several papers present SA results of applying different ML and DL (Deep Learning) approaches on IMDb Movie Reviews dataset [23-27].

Thus, in paper [23] were shown the results of comparative study of various classification techniques and existing rating movie rating systems used across the web. Proposed approach based on structured N-grams demonstrated the best classification accuracy of 89%.

Yenter and Verma [24] investigated CNN, LSTM and LSTM-CNN architectures for sentiment classification on the IMDb movie reviews in order to find the best-suited architecture for this dataset. They experimented with numerous regularization techniques, network structures, and kernel sizes. According to their results the best classification performance was obtained by using CNN model with accuracy above 89%.

Authors of paper [25] used DL approach based on Multi-Layer Perceptron (MLP), CNN, LSTM, and a hybrid CNN-LSTM model. They also compared the results of these four models with those obtained by SVM and NB on IMDb dataset. Word2vec technique was used for words embedding. The authors note that all DNNs outperformed ML models, and the best result was shown by using hybrid CNN-LSTM model (accuracy 89%).

Paper [26] presented results of SA on IMDb dataset received by using DNNs with an embedding layer that has been provided by Keras Library. Authors used a vocabulary of 8 000 unique tokens which was embedded into a 100 dimension vector space. Instead of using a pre-trained word embedding model (such as GloVe, Word2vec), they have trained embedding layer by using the training samples from IMDb movie review dataset. Experimental results have shown that CNN has achieved an F1-Score of 91% (Accuracy 90%) which has outperformed LSTM, LSTM-CNN and other state-of-the-art approaches for sentiment classification on IMDb movie reviews.

Quraishi [27] used both DL and ML approaches for SA on IMDb dataset. As DL models were used LSTM and Gated Recurrent Unit (GRU), and as ML classifiers – SVM and NB. He found that among these four algorithms, GRU performed the best with an accuracy of 89%.

3. Materials and methods

3.1. NLP pipeline

Most NLP pipelines consist of the following stages [6, 28]:

Text preprocessing aims to present text in natural language in a convenient format. This stage depends on the purpose of the research, and may include: removing punctuation marks, numbers and spaces; removal of tags and other markup elements, which mostly do not contain useful information and only add noise to the data; converting all words to lower case, etc.

Segmentation or tokenization is process of dividing the text into sentences, and sentences into separate words, word forms or morphemes (tokens).

Definition of context-independent features which characterize each token (independent of adjacent elements).

Filtering "stop words". English (and, in particular, the vast majority of European languages) are characterized by the presence of many auxiliary parts of speech (prepositions, conjunctions, pronouns, articles, etc.). They distort the statistical significance of other words in the document due to their rather large absolute, as well as the relative frequency of appearance in texts, regardless of the application area.

Dependency parsing, or semantic (content) analysis of the text – selection of semantic relations. In general, the semantic representation is a graph, a network that reflects binary relations between two nodes (meaningful units of the text).

Vector representation of tokens, or the creation of embeddings, which allows highlighting words that are used in a similar context.

Model design, which depends on the research purpose and task (for example, machine translation, text generation, classification, etc.).

3.2. Vector representation of words

One of the most important NLP problems is representation the text information in a form suitable for the use of ML methods and models. The most common approach is to extract features from the text at the level of individual words or sentences by aggregating information.

ML models are focused on working with data in the form of numerical vectors. Therefore, it is necessary to present the text (sentences, words) in numerical form.

There are several approaches for construction of numerical vectors for words [6, 28]: statistical or term-frequency methods (One Hot Encoding, Bag of Words, TF-IDF), and context depending words embeddings (GloVe, Word2vec, Doc2vec, FastText).

BOW

One of the first and simplest approaches is the Bag of Words (BOW). This is a simplified representation of the text that shows which words are found in the text, but does not take into account their order. The result of the presentation is a dictionary of unique words and their number in sentences and throughout the text as a whole.

One of the options for formalizing the BOW model is the formation of a numerical table, where unique words are coded in columns, and collection documents (texts, or paragraphs, sentences, etc.) are coded in rows. Frequencies (the number of occurrences of a word in a certain document) are placed at their intersection in the cells of the table. Thus, each line is a numerical vector characterizing the composition of the document.

Such a vector representation is easy to implement in the form of software code. Despite its simplicity, it turns out to be quite useful and allows successfully solving such NLP tasks as text classification (labelling text with a certain group or category marks).

Among the main disadvantages is that the word order is not taken into account. Also, growing the volume of the analysed text leads to increasing dimensionality of the dictionary matrix, as each unique word adds a new column to the matrix of embeddings. Moreover, such vectors are sparse, that is, they contain a large number of zero elements.

TF-IDF

The main problem with word frequency estimation is that a document may be dominated by commonly used words, but they generally do not contain as much useful information for the model as rarer domain-specific ones.

One of the approaches to overcoming this drawback is to adjust the frequency of words depending on how often they appear in a certain collection of documents (text corpora of a given subject area). The *TF-IDF* metric (Term Frequency, Inverse Document Frequency) is based on this idea.

The *TF* estimates the frequency of a word (*w*) in the current document (*d*) is equal to:

$$TF(w,d)=\frac{n_w}{\sum_k n_k}, \tag{1}$$

where n_w is a number of occurrences of the word *w* in document *d*; $\sum_k n_k$ is the total number of words in the document.

The *IDF* is equal to the logarithm of the ratio of the number of documents in the collection to the number of documents in the collection in which the given word occurs:

$$IDF(w,D)=\ln\frac{N}{N_w}, \tag{2}$$

where *N* is a total number of documents in collection (corpus) *D*; N_w is the number of documents in collection *D* in which the word w occurs.

The *TF-IDF* metric is calculated as the product of these indicators:

$$TF\text{-}IDF = TF(w,d)\times IDF(w,\ D). \tag{3}$$

Typically, a *TF-IDF* is defined for each word. A higher value of this metric indicates a higher importance of the word for that category, document, and collection of documents.

But, despite the intuitive clarity and ease of implementation of the above approaches for text vectors representation, context depending words embeddings that take into account the semantic relationships between words are mainly used as input factors (features) for practical NLP tasks (for example, Word2vec, GloVe, FastText, etc).

Word2vec

Word2vec (Word to Vectors) is a technology that was developed at Google in 2013 by Mikolov et al. [29]. It contains a set of algorithms and models based on fully connected DNNs for computing vector representations of words based on the hypothesis that words used in close contexts mean similar concepts, i.e. are semantically close.

Therefore, words that are similar in content should be represented by vectors that are close in some metric (for example, cosine distance) in the *N*-dimensional space of embeddings: the smaller distance between the vectors, the semantically closer the corresponding words.

Word2vec technology is based on the use of two opposite approaches: Continuous Bag of Words (CBOW) and Skip-Gram.

CBOW is an algorithm where we try to predict the target word using the context or surrounding words. The context refers to the closest words formed depending on the size of the context window – a sequence of *C* words in the text. One of these words (target or central) is missed, and the Neural Network (NN) tries to predict it. Thus, words that occur frequently in a similar context will have similar vectors. CBOW usually works well on small datasets.

Skip-Gram is an algorithm where we try to predict context words using a target word, which is the opposite of CBOW. Skip-gram works better on large datasets.

Schematically simplified architectures of the CBOW and Skip-Gram models are shown in Fig. 1.

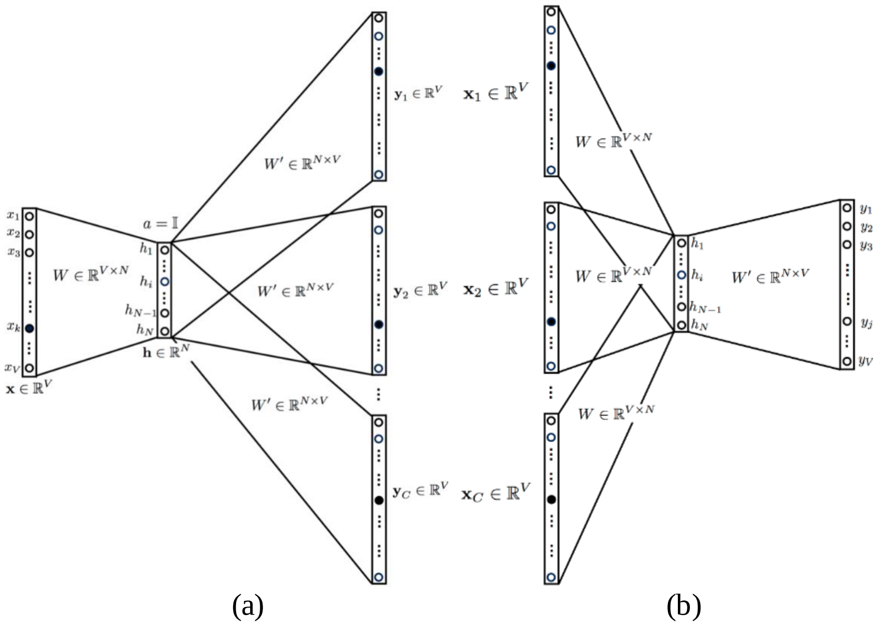

x_1
x_2
x_3
x_k
x_V
$\mathbf{x} \in \mathbb{R}^V$
$W \in \mathbb{R}^{V\times N}$
$a = \mathbb{I}$
h_1
h_i
h_{N-1}
h_N
$\mathbf{h} \in \mathbb{R}^N$
$W' \in \mathbb{R}^{N\times V}$
$\mathbf{y}_1 \in \mathbb{R}^V$
$\mathbf{y}_2 \in \mathbb{R}^V$
$\mathbf{y}_C \in \mathbb{R}^V$
$\mathbf{x}_1 \in \mathbb{R}^V$
$\mathbf{x}_2 \in \mathbb{R}^V$
$\mathbf{x}_C \in \mathbb{R}^V$
y_1
y_2
y_3
y_j
y_V
(a)
(b)

Fig. 1. Schematic representation of
Skip-Gram (a) and CBOW (b) models [30]

In both models, the input and output words are given in one-hot encoding vectors x, y $\in R^V$ (vector consists of 0s in all cells with the exception of a single 1 in a cell used uniquely to identify the word), where V is the size of dictionary D.

When they are multiplied by the matrix $\mathbf{W}_{V\times N}$ which is "connecting" the Input and Hidden layers, one row of $\mathbf{W}_{V\times N}$ is selected. As a result of this operation, we obtain the vector of the hidden state h $\in R^N$ of the word of a given dimension N which is a hyperparameter of the algorithm. This dimension is called embedding dimension.

The row matrix $\mathbf{W}_{V\times N}$ contains vector representations of words (embeddings) for the case when we are trying to predict a missing (target) word from its environment or context. The column matrix $\mathbf{W'}_{N\times V}$ contains vector representations of words (embeddings) for the case when we are trying to predict a context, or surrounding C words for selected target word.

Thus, based on the hidden state vector of the target word, Skip-Gram model should predict the words of its context, and CBOW model should predict the missing target word using the hidden state vectors of the context.

It should be noted that the Hidden layer neurons just copy the weighted sum of inputs to the next layer. There is no activation like Sigmoid, Tanh or ReLU. There is only non-linearity Softmax activation function (4) in the Output layer. Therefore, both models implement the Encoder-Decoder architecture.

Let us consider in more detail the algorithm of the CBOW (Fig. 1b). The input of the model is a sequence of C context words represented byone-hot vectors. This is a sequence with missing central or target word w_t (it is in the middle of the window of width C), which model should predict. For definiteness, we assume that a value of window width C is even (the number of context words before and after the target is the same).

The Hidden layer tries to predict the most probable target word. This operation is performed by multiplying the hidden state vector by the matrix $\mathbf{W'}_{N\times V}$.

The output layer contains V neurons with a Multivariate Logistic Activation Function (Softmax) that assigns a probability value p_j for all words $j=1,2,...,V$ from dictionary D:

$$\text{Softmax}\left(y_j\right)=\frac{\exp\left(y_j\right)}{\sum_{i=1}^{V}\exp\left(y_i\right)}=p_j\,, \tag{4}$$

where y_j is the one-hot vector which describes the probability distribution the word with number j from the dictionary ($j=\overline{1,V}$).

Training the CBOW model involves maximizing the objective function $L(\theta)$ (probability of predicting a word w_t based on the context

w_{t+j} (j = -C/2,…, -1, 1,…, C/2) with the width of the context window C, and θ is a vector of unknown parameters with $2N\times V$ dimensions):

$$L(\theta)=\prod_{t=1}^{T}\prod_{\substack{-\frac{C}{2}\le j\le\frac{C}{2}\\ j\ne0}} P\left(w_t\left|w_{t+j},\theta\right.\right), \tag{5}$$

where T is the number of training words.

In practice, the logarithm of the likelihood function is usually used, so it is necessary to minimize the following functional:

$$J(\theta)=-\frac{1}{T}\log L(\theta)=-\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-\frac{C}{2}\le j\le\frac{C}{2}\\ j\ne0}}\log\left(P\left(w_t\left|w_{t+j},\theta\right.\right)\right). \tag{6}$$

Taking into account (4), the functional (6) can be presented in the form:

$$\begin{aligned} J(\theta)&=-\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-\frac{C}{2}\le j\le\frac{C}{2}\\ j\ne0}}\log\frac{\exp\left(u_t^T v_{t+j}\right)}{\sum_{w\in D}\exp\left(u_t^T v_{t+j}\right)}=\\ &=-\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-\frac{C}{2}\le j\le\frac{C}{2}\\ j\ne0}}\left(u_t^T v_{t+j}-\log\sum_{w\in D}\exp\left(u_t^T v_{t+j}\right)\right), \end{aligned} \tag{7}$$

where $v_t\in\mathbf{W}_{V\times N}$ is the vector's representation of the word w_t when it is the target; $u_t^T\in\mathbf{W}'_{N\times V}$ is the vector's representation of the word w_t when it is contextual; $\sum_{w\in D}\exp\left(u_t^T v_{t+j}\right)$ is a normalizing factor designed to ensure that the value under the logarithm will be in the range from 0 to 1 (the sum is taken over all words *w* of the dictionary *D*, as seen in (4)).

After training, Word2vec allows to represent words as vectors from a space of a predetermined dimension. As a rule, for practical implementation, the dimensions of embeddings are chosen from 50 to 300.

Using this approach, any text document can be presented in the form of a $V \times N$ dimension matrix, where V is the number of words in the document (dictionary dimension), and N is the embedding vector dimension for a single word.

Thus, if the dictionary contains 100 000 words, and the dimension of the embedding is 300, the matrix of embeddings $\mathbf{W}_{V \times N}$ will contain 3×10^7 elements.

Word2vec was trained by Google on a huge corpus of unlabelled text, including Wikipedia, by learning words that occur in similar contexts.

GloVe

Based on similar ideas, experts from Stanford University proposed another model of embeddings – GloVe (Global Vectors) [31], which combines the features of Word2vec and the singular (or SVD) decomposition of the co-occurrence matrix of words.

Unlike Word2vec, the GloVe model tries to solve the problem of effective use of statistics of “co-occurrence” of words in the same context. The elements x_{ij} of the square matrix of co-occurrence of words $X = \left(x_{ij}\right)_{V \times V}$ (V is dictionary size) show the number of times when a word w_i occurred together (in a context with a certain window length) with a word w_j.

The objective function in the GloVe model minimizes the difference between the product of word vectors with indices i, j and the logarithm of the probability of their co-occurrence:

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{V}\sum_{j=1}^{V} f\left(P_{ij}\right)\left(u_i^T v_j-\log P_{ij}\right)^2, \tag{8}$$

where $P_{ij}=\dfrac{x_{ij}}{X_i}$ is the probability of the word with index *j* appearing in the context of the word with index *i*; X_i is the sum of the elements of the *i*-th row of the co-occurrence matrix; $f\left(P_{ij}\right)$ is discount factor for words (with indices *i*, *j*) that rarely occur together.

As for disadvantages of Word2vec and GloVe embeddings techniques, both of them can be used to obtain vectors for words from a dictionary, but these methods ignore that a word could have multiple forms and train vectors for each word's form independently.

In Word2vec, the frequency of co-occurrence of words does not matter much, it only helps to generate additional training samples. GloVe takes into account the frequency of co-occurrence of words, thereby improving the Word2vec model.

3.3. Description of classification models

Problem statement

Since the SA of the text in general is reduced to the binary classification problem (negative, positive), we have chosen Logistic Regression and Support Vector Machine as the basic classifier models (baseline). We will consider the supervised ML task.

Let's $\mathrm{X}=\left\{\mathrm{x}_1,\mathrm{x}_2,\ldots,\mathrm{x}_m\right\}$ is a training sample of texts (reviews, tweets), and $\mathrm{Y}=\left\{y_i\right\}$ is a vector of corresponding (target) binary class labels: $y_i\in\{0,1\}, i=1,2,\ldots,m$. Moreover, each element x_i is also a vector, which contains some vector representation of the words of the *i*-th text.

It is necessary to build a classification model:

$$b(\mathrm{X},\mathrm{w}):\mathrm{X}\rightarrow\mathrm{Y}, \tag{9}$$

where w is unknown weight vector.

The main idea of using a classifier is to divide the space of features (factors) by a hyperplane into two half-spaces, each of which predicts one of the two values of the target class.

LR classifier

Let the linear classifier model is given in the form:

$$b(\mathrm{X},\mathrm{w}) = \mathrm{sign}\left(\langle \mathrm{w},\mathrm{x}\rangle + w_0\right), \tag{10}$$

where $\langle\cdot\rangle$ denote scalar product.

Logistic Regression is a special case of the linear classifier and can be represented as:

$$b(\mathrm{X},\mathrm{w}) = f\left(\langle \mathrm{w},\mathrm{x}\rangle\right), \tag{11}$$

where $f(\cdot)$ is a function from the family of Sigmoid functions, in particular Hyperbolic tangent:

$$f(z) = \frac{2}{1+\exp(-kz)} - 1. \tag{12}$$

The task of training classifier is to adjust the vector of weights w for the training sample $\mathrm{X} = \{\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_m\}$ in such way as to minimize Loss function – the deviation between the labels of real classes and the labels predicted by the classifier.

Loss function for LR which optimizes the parameters of the model on the training sample and gives the correct estimates of the probability of belonging to a positive class for the case of binary class labels $y_i \in \{-1,1\}, i = 1,2,...,m$, can be expressed as follows [32]:

$$Q(\mathrm{w},\mathrm{X}) = \frac{1}{m}\sum_{i=1}^{m}\log\left(1+\exp\left(-y_i\langle \mathrm{w},\mathrm{x}_i\rangle\right)\right) + \frac{1}{2C}\|\mathrm{w}\|^2 \to \min_{w}. \tag{13}$$

The second term in (13) is a penalty for increasing the weight norm (L2 regularization), *C* is the regularization parameter.

LR has such an advantage that it can be used to predict the probability that an instance will belong to one of the classes.

SVM classifier

Support Vector Machine is one of the most popular algorithms for Supervised ML, used for both classification and regression tasks [32].

The main idea of the SVM is to map the source vectors into a higher-dimensional space and find the separating hyperplane with the maximum deviation in this space. Two parallel hyperplanes are built on both sides of the hyperplane, which is separating the classes. The separating hyperplane will be the one that maximizes the distance to these two parallel hyperplanes.

Note that if we simultaneously multiply the parameters w and w_0 of linear classifier (10) by the same positive constant, then the classification will not change. Therefore, we can accept that

$$\min_{\mathrm{x}\in\mathrm{X}} \left|\langle \mathrm{w}, \mathrm{x}\rangle + w_0\right| = 1. \tag{14}$$

The distance from an arbitrary point x_0 to the hyperplane determined by the classifier is equal to:

$$\rho(\mathrm{x}) = \frac{\left|\langle \mathrm{w}, \mathrm{x}\rangle + w_0\right|}{\|\mathrm{w}\|}, \tag{15}$$

where $\|\mathrm{w}\|$ is weight vector norm.

Then the minimum distance from the hyperplane to the nearest object of the training sample, taking into account (14), will be:

$$\min_{\mathrm{x}\in\mathrm{X}} \frac{\left|\langle \mathrm{w}, \mathrm{x}\rangle + w_0\right|}{\|\mathrm{w}\|} = \frac{1}{\|\mathrm{w}\|} \min_{\mathrm{x}\in\mathrm{X}} \left|\langle \mathrm{w}, \mathrm{x}\rangle + w_0\right| = \frac{1}{\|\mathrm{w}\|}. \tag{16}$$

This value is called a margin. The corresponding optimization problem, which determines the parameters of SVM for a samples that are linearly separable, can be written in the form:

$$\begin{cases} \frac{1}{2}\|\mathrm{w}\|^2 \to \min\limits_{\mathrm{w}, w_0}, \\ y_i\left(\left\langle \mathrm{w}, \mathrm{x}_i \right\rangle + w_0\right) \ge 1,\ i = 1, 2, \ldots, m, \end{cases} \tag{17}$$

where m is training sample size.

For the case when the classes are not linearly separable, the problem can be modified by introducing “soft” restrictions in (17) by introducing a “penalty” ξ_i for their violation:

$$y_i\left(\left\langle \mathrm{w}, \mathrm{x}_i \right\rangle + w_0\right) \ge 1 - \xi_i, i = 1, 2, \ldots, m. \tag{18}$$

Thus, if any object’s margin is between -1 and 1, it will be correctly classified, but have a non-zero penalty. So, the task is to finding a compromise between maximizing deviation and minimizing the total Loss:

$$\begin{cases} \frac{1}{2}\|\mathrm{w}\|^2 + C\sum\limits_{i=1}^{m}\xi_i \to \min\limits_{\mathrm{w}, w_0, \xi_i}, \\ y_i\left(\left\langle \mathrm{w}, \mathrm{x}_i \right\rangle + w_0\right) \ge 1 - \xi_i,\ i = 1, 2, \ldots, m, \\ \xi_i \ge 0. \end{cases} \tag{19}$$

FCN classifier

One of the simplest architectures of deep networks used for classification problems is Fully Connected Network (in particular, Multi-Layer Perceptron). Features of the classified object are fed to the input neurons of MLP, and the class label is formed at the output (Fig. 2).

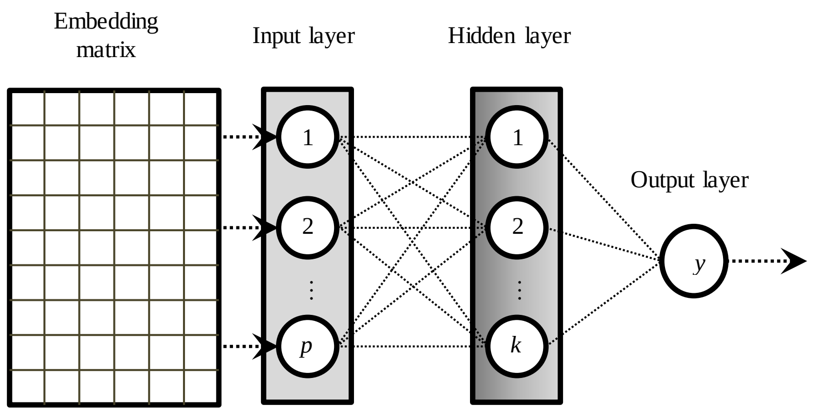


Fig. 2. A simplified architecture of a FCN with one hidden layer

Word vectors (embeddings) of the tokenized text are given to the input layer of the network (see Fig. 2). Then the signal is propagated from the input layer to the output one. Here, each neuron of the previous layer is connected to all neurons of the next layer without forming feedback connections.

The idea of the training process is similar to the training of other classifiers. The network adjusts its connection weights, which are involved in the calculations in such a way as to minimize the Loss function, or the deviation of the values at the output layer from the real values. In our case, these are class labels – binary values 0 or 1.

Activation functions are used to transform the signal in the neurons of the intermediate and output layers. One of the most common in DL is ReLu (Rectified Linear Unit) function [28]. This function returns the value of the argument for positive values, and 0 otherwise:

$$\mathrm{ReLu}(x) = \max(0, x). \tag{20}$$

The ReLu function has several advantages over the Sigmoid and Hyperbolic tangent. In particular, the derivative is quickly calculated for it, which is important for training the network with gradient methods.

CNN classifier

A Convolutional Neural Network is a special model of NNs that was proposed by LeCun and Bengio for image processing [33]. The CNN architecture is based on the sequential arrangement of Convolutional and Subsampling (Max Pooling) layers (see Fig. 3). As a rule, the maximum function is chosen as a subsampling operation.

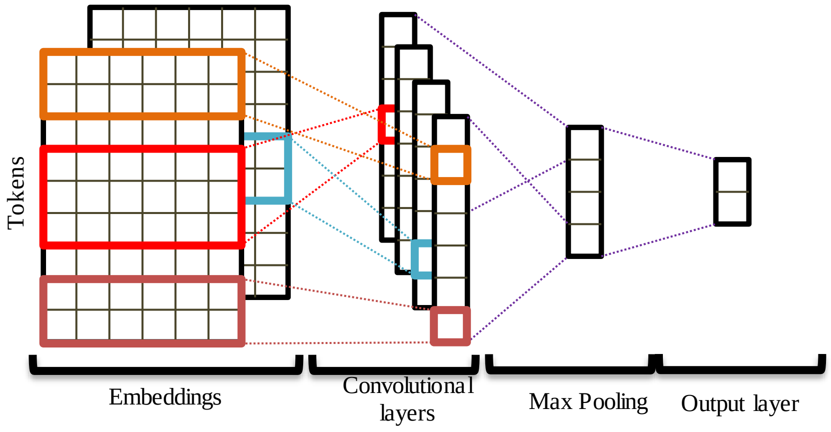


Fig. 3. A simplified architecture of CNN for processing text

Embedding layer generates a matrix of embedding tokens with the dimension of the number of tokens multiplied by the dimension of the embedding space.

Convolutional layers perform a convolutional operation to the outputs of a previous layer (Embedding), where the weights of the convolutional kernels are the learned parameters.

Unlike FCN, CNN uses only a limited weight matrix of a small size in the convolution operation, which is "moved" throughout the layer, forming after each shift an activation signal for the neuron of the next layer with a similar position. Thus, the same weight matrix, or convolution kernel, is used to generate an output signal from this layer.

The size of the kernel is determined by the number of features that will be combined to obtain a new feature at the output. Moreover, each

Convolutional layer could have not only one convolution kernel, but several. Each kernel is designed to extract certain specific feature. These sets of kernels are called filters.

As a result of the convolution operation, a matrix of weights is formed. This matrix (Feature Map) characterizes the presence of a certain feature in the input template. The passage of each set of weights (filter) forms its own instance of the Feature Map, making multi-channel NN (presence of many independent feature maps on one layer).

The Max-pooling layer is designed for the subsampling (usually maximum) operation, which reduces the dimensionality of the generated feature maps. The Max-pooling operation consists in moving the sifting window along the data. From the initial matrix of data (feature map) falling into its field of view, the maximum is selected and moved to the resulting matrix. This operation is performed in order to speed up the training process and reduce the consumption of computing resources.

The last (Output) layer of the CNN is fully-connected. It is actually a classifier: neurons in it are activated using Sigmoid function (or Softmax for multi-class classification), which returns values from 0 to 1.

These values can be interpreted as probabilities of belonging to a certain class. For the classification problem, the number of output neurons is equal to the number of categories. In the case of binary classification problem we can use just one neurone in Output layer.

Unlike pixels in image processing, the input data for NLP problems are words, sentences or documents formalized in the form of a numerical matrix. In our case, each row of the matrix corresponds to a token, which is a word.

As these vectors, we can use word frequencies (for example, obtained using the *TF-IDF* metric), or pre-trained embeddings of a given dimension (in particular, Word2vec, GloVe, FastText, etc.).

Since tokens are located in only one dimension, instead of 2D filters applied to two-dimensional input images data, text convolution is performed using one-dimensional filters (1D Convolution) on one-dimensional input data, for example, sentences, using convolution kernels of different widths (see, for example [34]). Kernels with width of 2, 3, 4, 5 are usually used for processing texts [6, 28, 34]. These sizes correspond to the use of not individual words, but words combination: bi-gram, tri-gram, etc.

4. Data and software implementation

For models design and their implementation, the Python 3 programming language and the Scikit-learn library [35] (for LR, SVM models and evaluation of accuracy scores), Keras [36] and Tensorflow [37] (for MLP and CNN models) were used.

To test sentiment analysis models, we used the English-language dataset IMDB (Large Movie Review Dataset) [38], intended for binary sentiment classification. This dataset contains a sufficient amount of data (50 000 labelled texts of movie reviews), has no gaps and is balanced (it has an equal number of positive and negative reviews – 25 000 for each class). This dataset is the de facto standard for testing various ML models for NLP.

For text pre-processing and tokenization, we used the package of Python libraries and programs for symbolic and statistical processing of natural language NLTK (Natural Language ToolKit) [39], as well as custom tools based on regular expressions. Pre-trained Word2vec and GloVe embeddings were imported using the Gensim Python library [40]. All models were implemented using the Google Colab cloud service [41].

5. Empirical results

5.1. Pre-processing and vectorization

Pre-processing of the text included removing punctuation marks, mark-up tags, html addresses, removing stop words and converting all words to lower case. Reviews length analysis shows (Fig. 4) that the vast majority of them have a length of no more than 500 words.

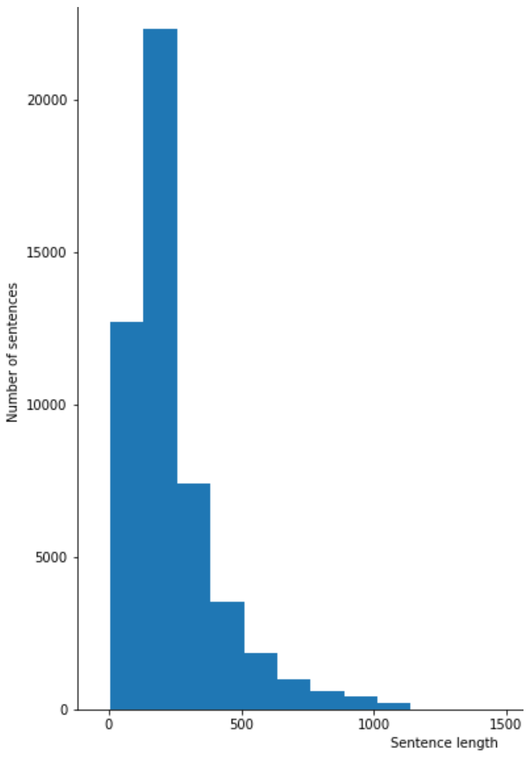


Fig. 4. Diagram of the distribution of the length of reviews

Since for the implementation of the models it is necessary that the input array (tensor) has a fixed dimension, 500 tokens were chosen for the maximum length of the statement: reviews that had a longer length were “cut off” from the end, and reviews that had a shorter length were supplemented from the beginning of expressions with 0-tokens (padding operation).

As embeddings, we used both classic TF-IDF and pre-trained Word2vec and GloVe embeddings. We tested two different embeddings sizes: with dimensions of 100 and 300.

After tokenization, about 1.2 million words remained in the dataset, which are contained in 50 000 marked texts of movie reviews. The volume of the dictionary (the number of unique words after pre-processing and tokenization) was $V = 92\ 393$.

Note that not all unique words from the dictionary have pre-learned embeddings: so, in particular, the "dictionary coverage" for the Word2vec-300 model was about 61%, GloVe-300 – 75.5%, and GloVe-100 – 66.4%.

5.2. Final models design and hyperparameters tuning

Hyperparameter values for LR and SVM models (regularization parameter *C*, kernel type, and optimization method) were selected by cross-validation using grid search. The final settings of the hyperparameters are given in Table 1.

Table 1

FINAL HYPERPARAMETERS SETTINGS FOR LR AND SVM MODELS

Parameter	LR	SVM
Regularization type	L2	L2
Regularization constant *C*	20	10
Kernel	–	Radial basis functions (rbf)
Numerical optimization method	Modified Newton method (newton-cg)	–
Validation subset size	20%	20%

Final settings of hyperparameters of NNs models are given in Table 2.

We used two of the simplest architectures of DNNs with a minimal number of layers: fully connected (MLP) and convolutional (CNN). For these models, both pre-trained Word2vec and GloVe embeddings were tested to initialize the weights of the first layer (Embedding layer). We have carried out computer simulations with embeddings of dimensions 100 and 300.

Table 2

FINAL HYPERPARAMETERS SETTINGS FOR MLP AND CNN MODELS

Parameter	MLP	CNN
Validation subset size	20%	20%
Number of setting parameters	220 402	230 913
Dropout – the number of layers (in brackets are the proportions of broken neural connections)	3 (0.3)	2 (0.2)
Activation function of Intermediate layers/Output layer	ReLu/Sigmoid	ReLu/Sigmoid
Numerical optimization method	Adam	Adam
Mini-batch	128	128
Loss function	Binary Cross Entropy	Binary Cross Entropy
Score metrics	Accuracy	Accuracy
Filter numbers	–	256
Kernel size	–	5

One of the problems of learning deep neural networks is overfitting – a phenomenon when the model explains well the data from the training sample, but performs relatively poorly on the data that did not participate in the training (test sample). Overfitting is the result of too “tight fitting” model parameters to the dependencies contained in the training data set. If overfitting occurs, model does not acquire the generalizing ability to extend hidden dependencies and regularities discovered on the training sample to new data.

One of the effective methods of overfitting prevention is the artificial breaking of part of the connections during the training process, or dropout. A predetermined percentage of neural connections in the NN are forcibly broken. Therefore, part of the information is sort of forgotten.

Thus, instead of perfectly matching the weights to only the training data set, the network learns to “find the answers” for similar data instances that were not found in the training sample. Therefore, dropout layers were used for both MLP and CNN models to prevent overfitting (see Table 2, Figs. 5, 6).

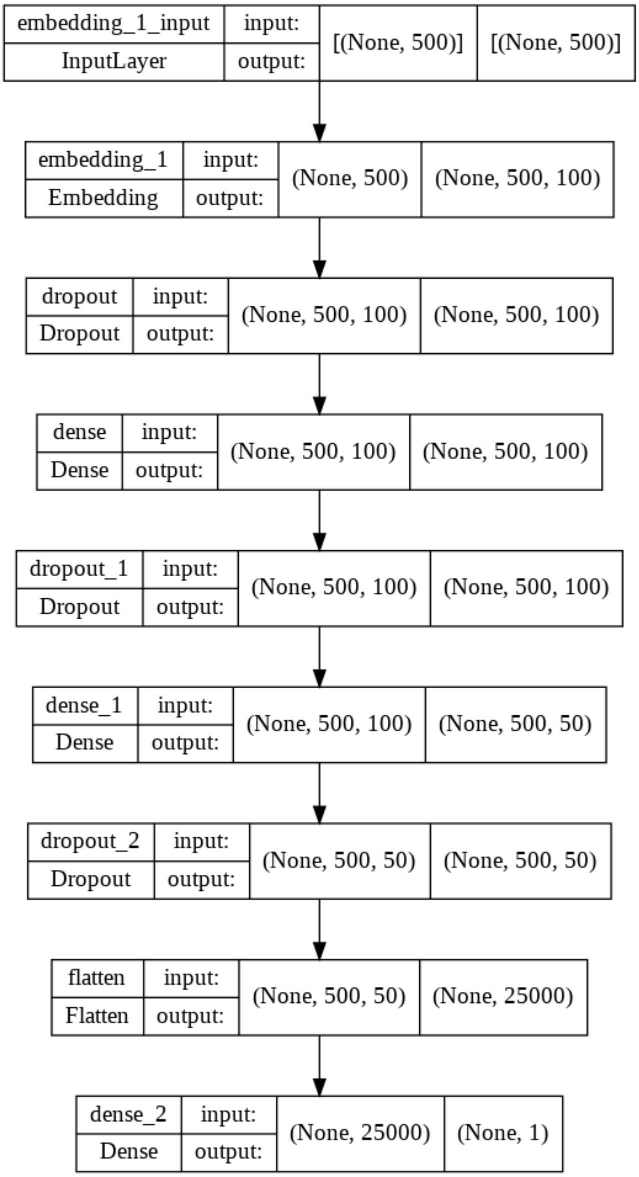


Fig. 5. MLP model graph for embeddings of length 100

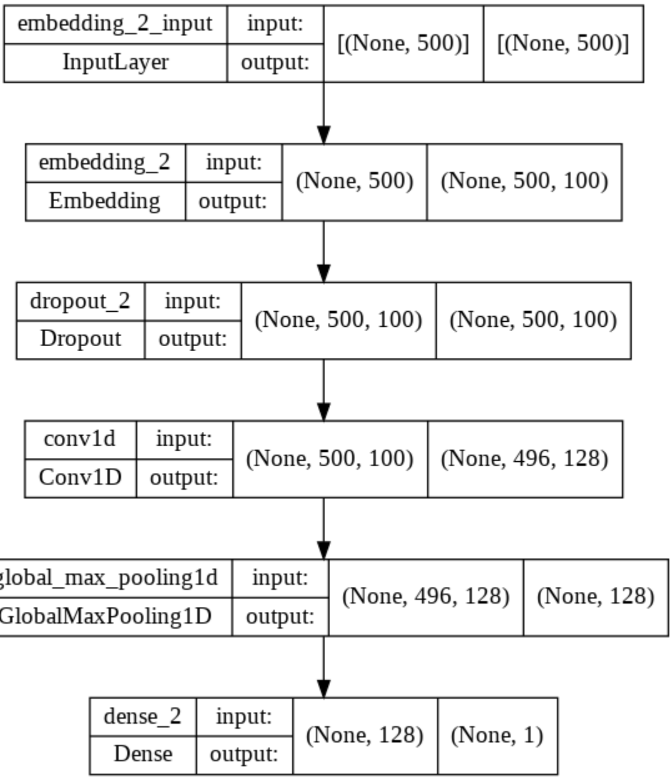


Fig. 6. CNN model graph for embeddings of length 100

Graphs in Figs. 5, 6 were formed using plot_model function of Keras Library for model visualization (plotting) [36]. These graphs conditionally show the sequence of the layers of the DNNs, as well as the dimensions (shape) of the input (the middle part of the boxes) and output data (right side of the boxes).

For example, Embedding layer gets a sequence of 500 words as input and returns a tensor with shape 500 by 100, where 100 is embeddings dimension of each word. For the reason that we take as input not images (2D arrays), but 1D vectors of words, the 3rd axis of our data shape is not defined (the value of this axis is displayed as "None" in the 1st position in the middle and right parts of the boxes in Figs. 5, 6).

In the MLP model (see Fig. 5), behind the Embedding layer there are two dense hidden layers that contain 100 and 50 neurons (300 and 100 for embeddings of length 300) that have the ReLu activation function, as well as a Flatten layer. Dense layers are ordinary fully connected layers.

Since we have a tensor (2D-array with shape 500x50) at the output of the last Dense layer (after Dropout), and the input to the Output (fully connected) layer must be a one-dimensional array, an alignment layer Flatten is used to transform the data, which decomposes the array into a dense vector of length 25 000.

For both MLP and CNN, the last (Output) layer with a single neuron and Sigmoid activation function is used to predict the most probable output: whether the review belongs to the positive or negative class.

As you can see in Fig. 6, in the CNN model after Embedding layer a Convolutional layer (1D Convolution) was used with 128 filters (256 for embeddings dimension 300). The best results were obtained with kernel size (width) equal to 5 what determines the processing of sequences of 5 words (tokens).

After that Global Max pooling layer is located which is Max pooling with pool size equals to the input's one. Its using has the advantage that the feature maps can be easily interpreted as categories' confidence maps. So we can see Global Max pooling as a structural regularizer that explicitly enforces feature maps to be confidence maps of predicted categories.

Finally, Dense layer with ReLu activation function was added to reduce the dimensionality of the raw data and perform classification.

Since the problem of SA is reduced to binary classification, Binary Cross-Entropy was chosen as the Loss function for MLP and CNN models. Binary cross-entropy can be calculated as the average cross-entropy over all data instances [28, 32]:

$$L_{CE} = -\frac{1}{N}\left[\sum_{i=1}^{N}\left(y_i \log\left(p_i\right) + \left(1-y_i\right)\log\left(1-p_i\right)\right)\right], \tag{21}$$

where N is a sample size; $y_i = \{0,1\}$ is a real (actual) class label for the i-th data instance; p_i is the probability of belonging to a positive class (to $y_i = 1$) calculated by the classifier for the i-th data instance.

5.3. Accuracy models score and empirical results

Accuracy scores

The most common function for assessing model's quality for classification problems is Mean Consequential Error (MCE), i.e. the share (percentage) of objects for which model gave correct answers [28, 32]:

$$MCE = \frac{1}{m}\sum_{i=1}^{m} I\left[y_i^{fact} = y_i^{pred}\right], \tag{22}$$

where y_i^{fact}, y_i^{pred} are the actual and predicted values of the target class, respectively; I is an indicator function; m is the sample size.

Another important tool for evaluation accuracy score is the Confusion Matrix [32] – a table with various combinations of predicted and actual values. Predicted values are described as “positive” and “negative”, and actual values are described as “true” and “false”.

For a binary classification problem, this is a 2×2 matrix, as seen in Table 3. In this matrix, at the intersection of the i-th row and the j-th column, the number of objects of the i-th class is given, to which the algorithm gave the label of the j-th class (i, j = 0, 1).

Here, objects that the algorithm classifies as belonging to the positive class (Positive, P or 1) and which actually belong to this class are True Positive (TP), otherwise – False Positive (FP). Similarly for the negative class (Negative, N or 0): if the model classifies object as “negative” and it really belongs to the negative class – True Negative (TN), otherwise – False Negative (FN).

Table 3

CONFUSION MATRIX STRUCTURE

		Actual output, y^{fact}	
		N (0)	P (1)
Predicted output, y^{pred}	N (0)	TN	FN
	P (1)	FP	TP

Then the *Accuracy* metric can be defined as follows:

$$Accuracy = \frac{TP + TN}{P + N}. \qquad (23)$$

Accuracy on the positive class (Precision, Positive Predictive Value) reflects what percentage of objects that were classified as positive are correctly classified:

$$Precision = \frac{TP}{TP + FP}. \qquad (24)$$

Recall (True Positive Rate) reflects what percentage of objects of the true positive class we correctly classified:

$$Recall = \frac{TP}{TP + FN}. \qquad (25)$$

F_1*score* is the harmonic mean between Precision and Recall:

$$F_1\,score = \frac{2TP}{2TP + FP + FN}. \qquad (26)$$

Training models

NNs models were trained for 5 epochs with a minibatch size of 128 (the number of sample objects processed by the model after which the network weights updated).

The dependencies of the accuracy and errors of the NNs models on the train and validation (test) subsets for Word2vec-300 embeddings on the number of training epochs are shown in Fig. 7.

Analysis of the graphs in Fig. 7 allows us to preliminarily assume that there is no overfitting effect: both on the training and test samples, the error curves progressively decrease (lower graphs), and the accuracy curves grow steadily (upper graphs). It can be argued that on the 5-th epoch the accuracy curve on the test set will go approximately to the asymptote, which indicates the inexpediency of the continuing the training process. For other embeddings we have got the similar dependencies.

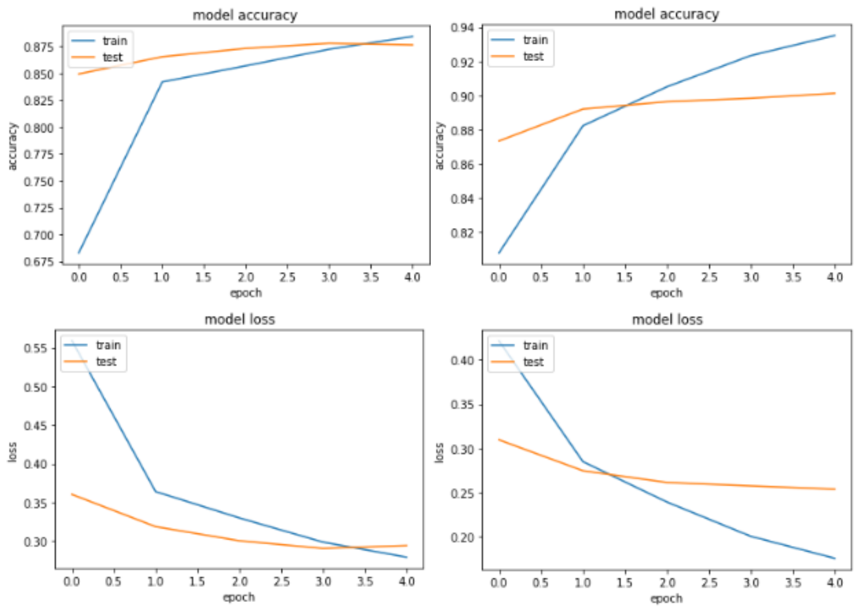
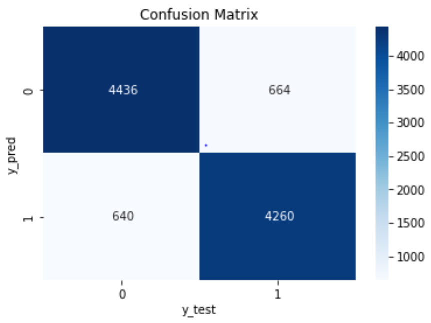
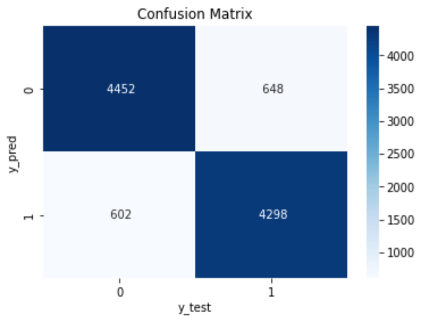


Fig. 7. Graphs of accuracy (upper figures) and error (lower figures) on the training and test sub-sets for the MLP (a) and CNN (b)

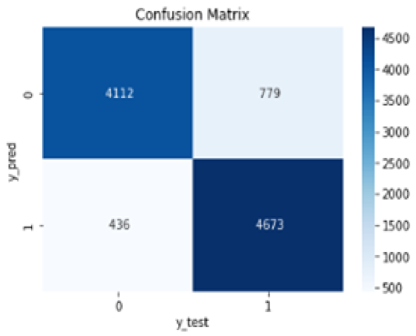
Confusion matrixes for all four classifiers are shown in Fig. 8. For LR and SVM models the results are given for *TF-IDF* embeddings (best results), for MLP and CNN – for Word2vec-300 embeddings (best results).



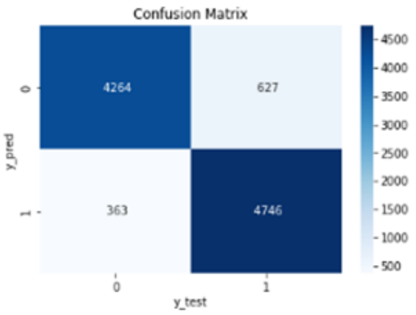
(a)



(b)



(c)



(d)

Fig. 8. Confusion matrices for LR (a), SVM (b), MLP (c), and CNN (d) models on the test sample

The values of classification metrics (23)-(26) (Classification Report) for all four classifiers on the test sub-set (size of 10 000), data from which were not used in the training process, are given in Tables 4-7.

Table 4

CLASSIFICATION REPORT FOR LR MODEL (TF-IDF EMBEDDINGS)

	Precision	***Recall***	***F_1score***	***Support***
Class 0*	0.87	0.87	0.87	4891
Class 1	0.87	0.87	0.87	5109
Accuracy			**0.87**	**10000**
macro avg**	0.87	0.87	0.87	10000
weighted avg**	0.87	0.87	0.87	10000

*Note, although metrics Precision and Recall are defined for the positive class, the function classification_report of Scikit-Learn Library also provides their values according to functions (24), (25) for the negative class, based on its prediction accuracy scores [35].
**The reported averages include macro average (averaging the unweighted mean per label), weighted average (averaging the support-weighted mean per label) [35].

Table 5

CLASSIFICATION REPORT FOR SVM MODEL (TF-IDF EMBEDDINGS)

	Precision	***Recall***	***F_1score***	***Support***
Class 0	0.87	0.88	0.87	4891
Class 1	0.87	0.88	0.87	5109
Accuracy			**0.875**	**10000**
macro avg	0.87	0.87	0.87	10000
weighted avg	0.87	0.87	0.87	10000

Table 6

CLASSIFICATION REPORT FOR MLP MODEL (WORD2VEC-300 EMBEDDINGS)

	Precision	***Recall***	***F_1score***	***Support***
Class 0	0.90	0.84	0.87	4891
Class 1	0.86	0.91	0.88	5109
Accuracy			**0.88**	**10000**
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000

Table 7

CLASSIFICATION REPORT FOR CNN MODEL (WORD2VEC-300 EMBEDDINGS)

	Precision	***Recall***	***F_1score***	***Support***
Class 0	0.92	0.87	0.90	4891
Class 1	0.88	0.93	0.91	5109
Accuracy			**0.90**	**10000**
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

An analysis of confusion matrices in Figs. 8a, 8b and classification reports in Tables 4, 5 shows that the LR and SVM models have sufficiently high classification accuracy of 87% and 87.5%, respectively (on *TF-IDF* embeddings). At the same time, these two models classify both positive and negative classes with almost the same accuracy (Figs. 8a, 8b). Classification errors are also balanced, with roughly equal numbers of misclassified objects for the negative and positive classes.

As to NNs models, despite their rather simple architecture, they showed overall somewhat better classification results: 88% and 90% for MLP and CNN, respectively (GloVe-300 and Word2vec-300 embeddings). At the same time, both models better classify the positive class (Figs. 8c, 8d, Tables 6, 7). The share of correctly classified objects of the positive class (*Recall* metric) is 91% and 93%, respectively.

Moreover, both models gave more False Negative predictions than False Positive: the *FN* value was 779 (7.8%) and 627 (6.3%) for MLP and CNN, respectively. Therefore, it can be concluded that the results are biased for predicting the positive class.

The final accuracy scores (*Accuracy* metric) for all models and embeddings are given in the Table 8.

Table 8

FINAL ACCURACY METRIC FOR ALL MODELS AND EMBEDDINGS

Model / embedding	***TF-IDF***	**GloVe-100**	**GloVe-300**	**Word2vec-300**
LR	0.870	0.815	0.840	0.859
SVM	0.875	0.814	0.843	0.870
MLP	0.827	0.845	0.879	0.877
CNN	0.834	**0.881**	**0.901**	**0.901**

As the simulation results show, using of pre-trained embeddings for classifiers based on conventional ML models (Logistic Regression and Support Vector Machine) in general turned out to be less efficient than using *TF-IDF* (Table 8). In our opinion, this may be caused by the fact that the pre-trained embeddings covered no more than 75% of the vocabulary of the studied dataset. But on *TF-IDF* embeddings, these classifiers showed sufficiently high accuracy: 87.5% and 87% for SVM and LR, respectively.

As for the DNNs models, CNN performed better on all embeddings than the Fully Connected Network (MLP). It should be noted that increasing the embedding dimension from 100 to 300 improved the classification accuracy for all models by an average of 2-3%.

For neural networks, the classification quality for Word2vec-300 and GloVe-300 embeddings was almost the same: about 88% for MLP and 90% for CNN. But GloVe-300 embedding has a much higher percentage of dictionary coverage than Word2vec (75% and 61%, respectively), which turned out to be somewhat unexpected.

6. Conclusion and further research directions

Our research results have shown that for Sentiment Analysis of social media, at least for binary classification, conventional ML models and DNNs of a relatively simple architecture with a small number of layers have, in general, a level of accuracy sufficient for practical use-cases.

For the selected English-language dataset the classification accuracy using the Logistic Regression model was 87%, the Support Vector Machine – 87.5%, the Fully Connected Neural Network – 88%, and the Convolutional Network – 90%.

It should be noted that our results outperformed (or was the same) the results of the approaches proposed in [23-27]. We believe this was achieved due to: (i) the use of pre-trained words embeddings, (ii) better text pre-processing, (iii) choosing an adequate network architecture, and (iv) finer hyperparameters tuning.

For our opinion, the accuracy of the classification can be increased by using lemmatization (or stemming) which is applied in order to convert the words to their normal form. Also, it may be appropriate to use word embeddings weighted by their *TF-IDF* metric.

In addition, taking into account the fact that the pre-trained embedding models cover, depending on the dataset, only a certain part of it, it is possible to try for the missing words to use the weighted average value of the embeddings of the neighbouring words with a certain window length, or to replace the missing words with normalized *TF-IDF* embeddings transformed by using the method of principal components (SVD decomposition of the sparse *TF-IDF* matrix to reduce its dimension).

We believe that a promising direction for carrying out Sentiment Analysis of texts in social media is designing models based on Deep Convolutional Networks, or the synthesis of Convolutional and Recurrent networks. Using of pre-trained embeddings (GloVe, Word2vec, FastText) as a vector representation of words is more appropriate then statistical (frequency based) ones.

At the same time, the use of pre-trained embeddings allows to start training Deep Networks not from randomly generated values of model parameters (weighs), but already to some extent adapted to the text classification task. Moreover, the learning process is accelerated and the generalization abilities of classifiers based on deep networks are improved.

For practical implementation of monitoring electronic mass media, it is advisable to conduct a more detailed Aspect-based Sentiment Analysis, since judgments of different polarities can be expressed in one text regarding individual characteristics (aspects) of the object under study [7, 41]. So, in particular, if we analyze consumer reviews about restaurants, aspects can be prices, service level, quality of dishes, etc.

Summarizing, we note that GloVe and FastText pre-trained embeddings are created for many languages, including Ukrainian. However, at the time of preparing the article, the authors could not find comprehensive Ukrainian-language datasets for Sentiment Analysis.

Since manual labelling of texts is too time-consuming and expensive, the use of unsupervised ML, Transfer Learning (see, for example, [42]), automated labelling of texts and so on, may be promising for the Sentiment Analysis of Ukrainian-language resources based on ML approaches.

References

1. Kemp, S. (2020, January 30). *Digital 2020: Global Digital Overview*. DataReportal. https://datareportal.com/reports/digital-2020-global-digital-overview

2. Li, H. (2017). Deep learning for natural language processing: advantages and challenges. *National Science Review*, *5*(1), 24-26. https://doi.org/10.1093/nsr/nwx110

3. Silberztein, M., Atigui, F., Kornyshova, E., Métais, E., & Meziane, F. (Eds.). (2018). Lecture Notes in Computer Science*: Vol. 10859. Natural Language Processing and Information Systems*. Springer. https://doi.org/10.1007/978-3-319-91947-8

4. Soloviev, V., Moiseienko, N., & Tarasova, O. (2020). Complexity Theory and Dynamic Characteristics of Cognitive Processes. In V. Ermolayev, F. Mallet, V. Yakovyna, H. Mayr, & A. Spivakovsky (Eds.), *Communications in Computer and Information Science: Vol. 1175. Information and Communication Technologies in Education, Research, and Industrial Applications* (pp. 231–253). Springer. https://doi.org/10.1007/978-3-030-39459-2_11

5. Kiv, A., Soloviev, V., Tarasova, E., Koycheva, T., & Kolesnykova, K. (2020). Semantic knowledge networks in education. *E3S Web of Conferences*, *166*, Article 10022. https://doi.org/10.1051/e3sconf/202016610022

6. Lane, H., Howard, C., & Hapke, H. (2019). *Natural Language Processing in Action: Understanding, Analyzing, and Generating Text with Python*. Manning Publications Co.

7. Pozzi, F., Fersini, E., Messina, E., & Liu, B. (2017). *Sentiment Analysis in Social Networks*. Elsevier. https://doi.org/10.1016/B978-0-12-804412-4.09994-0

8. Zhang, X., & Ghorbani, A. A. (2020). An overview of online fake news: Characterization, detection, and discussion. *Information Processing & Management*, *57*(2), Article 102025. https://doi.org/10.1016/j.ipm.2019.03.004

9. Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, *19*(1), 22-36. https://doi.org/10.1145/3137597.3137600

10. Kononova, K. (2017). Trust evaluation: theoretical models and results of empirical research on social networks. *Neiro-Nechitki Tekhnolohii Modelyuvannya v Ekonomitsi (Neuro-Fuzzy Modeling Techniques in Economics)*, *6*, 68-89. https://doi.org/10.33111/nfmte.2017.068 [In Russian]

11. Iglesias, C., & Moreno, A. (Eds.). (2020). *Sentiment Analysis for Social Media*. MDPI. https://doi.org/10.3390/books978-3-03928-573-0

12. Kononova, K., & Dek, A. (2018). Investigation of traders' behavioral characteristics: experimental economics methods and machine learning technologies. *Neiro-Nechitki Tekhnolohii Modelyuvannya v Ekonomitsi*

(Neuro-Fuzzy Modeling Techniques in Economics), *7*, 148-167. https://doi.org/10.33111/nfmte.2018.148 [In Ukrainian]

13. Khoo, C. S., & Johnkhan, S. B. (2018). Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons. *Journal of Information Science*, *44*(4), 491-511. https://doi.org/10.1177/0165551517703514

14. Karamollaoglu, H., Dogru, I. A., Dorterler, M., Utku, A., & Yıldız, O. (2018). Sentiment analysis on Turkish social media shares through lexicon based approach. In *Proceedings of the 2018 3rd International Conference on Computer Science and Engineering (UBMK)* (pp. 45-49). IEEE. https://doi.org/10.1109/UBMK.2018.8566481

15. Dhaoui, C., Webster, C. M., & Tan, L. P. (2017). Social media sentiment analysis: Lexicon versus Machine Learning. *Journal of Consumer Marketing*, *34*(6), 480-488. https://doi.org/10.1108/JCM-03-2017-2141

16. Kumar, A., & Jaiswal, A. (2020). Systematic literature review of sentiment analysis on Twitter using soft computing techniques. *Concurrency and Computation: Practice and Experience*, *32*(1), Article e5107. https://doi.org/10.1002/cpe.5107

17. Drus, Z., & Khalid, H. (2019). Sentiment Analysis in Social Media and Its Application: Systematic Literature Review. *Procedia Computer Science*, *161*, 707-714. https://doi.org/10.1016/j.procs.2019.11.174

18. D'Andrea, A., Ferri, F., Grifoni, P., & Guzzo, T. (2015). Approaches, Tools and Applications for Sentiment Analysis Implementation. *International Journal of Computer Applications*, *125*(3), 26-33. https://doi.org/10.5120/ijca2015905866

19. Moussa, M., Mohamed, E., & Haggag, M. (2018). A survey on opinion summarization techniques for social media. *Future Computing and Informatics Journal*, *3*(1), 82-109. https://doi.org/10.1016/j.fcij.2017.12.002

20. Jain, P. K., & Pamula, R. (2020). *A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews*. arXiv. https://doi.org/10.48550/arXiv.2008.10282

21. Amrani, Y., Lazaar, M., & Kadiri, K. (2018). Random Forest and Support Vector Machine based Hybrid Approach to Sentiment Analysis. *Procedia Computer Science*, *127*, 511-520. https://doi.org/10.1016/j.procs.2018.01.150

22. Chen, N, & Wang, P. (2018). Advanced combined LSTM-CNN model for Twitter sentiment analysis. In *Proceedings of the 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)* (pp. 684-687). IEEE. https://doi.org/10.1109/CCIS.2018.8691381

23. Sahu, T.P., & Ahuja, S. (2016). Sentiment analysis of movie reviews: A study on feature selection & classification algorithms. In *Proceedings of the 2016 International Conference on Microelectronics, Computing and Communications (MicroCom)* (pp. 1-6). IEEE. https://doi.org/10.1109/MICROCOM.2016.7522583

24. Yenter, A., & Verma, A. (2017). Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. In *Proceedings of the 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)* (pp. 540-546). IEEE. https://doi.org/10.1109/UEMCON.2017.8249013

25. Ali, N.M., Hamid, M.M., & Youssif, A. (2019). Sentiment Analysis For Movies Reviews Dataset Using Deep Learning Models. *International Journal of Data Mining & Knowledge Management Process*, *9*(2/3), 19-27. https://doi.org/10.5121/ijdkp.2019.9302

26. Haque, M. R., Lima, S. A., & Mishu, S. Z. (2019). Performance Analysis of Different Neural Networks for Sentiment Analysis on IMDb Movie Reviews. In *Proceedings of the 2019 3rd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)* (pp. 161-164). IEEE. https://doi.org/10.1109/ICECTE48615.2019.9303573

27. Quraishi, A. (2020). Performance Analysis of Machine Learning Algorithms for Movie Review. *International Journal of Computer Applications*, *177*(36), 7-10. https://doi.org/10.5120/ijca2020919839

28. Kamath, U., Liu, J., & Whitaker, J. (2019). *Deep Learning for NLP and Speech Recognition*. Springer Nature Switzerland AG. https://doi.org/10.1007/978-3-030-14596-5

29. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv. https://doi.org/10.48550/arXiv.1301.3781

30. Marginalia. (2018, January 6). *The backpropagation algorithm for Word2Vec*. http://www.claudiobellei.com/2018/01/06/backprop-word2vec/

31. Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532-1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

32. Mohri, M., Rostamizadeh, A., Talwalkar, A. (2018). *Foundations of Machine Learning* (2nd ed.). MIT Press.

33. LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 255-258). The MIT Press.

34. Kim, Y. (2014). *Convolutional neural networks for sentence classification*. arXiv. https://doi.org/10.48550/arXiv.1408.5882

35. Scikit-Learn. (n.d.). *Machine Learning in Python*. Retrieved January 25, 2020, from https://scikit-learn.org/stable/

36. Keras. (n.d.). *Keras: Simple. Flexible. Powerful*. Retrieved January 25, 2020, from https://keras.io

37. TensorFlow. (n.d.). *Create production-grade machine learning models with TensorFlow*. Retrieved January 25, 2020, from https://www.tensorflow.org

38. IMDb. (n.d.). *IMDb Datasets* [Data set]. Retrieved January 9, 2020, from https://www.imdb.com/interfaces/

39. NLTK. (n.d.). *Natural Language ToolKit*. Retrieved January 25, 2020, from https://www.nltk.org

40. Gensim. (n.d.). *Topic modelling for humans*. Retrieved January 25, 2020, from https://radimrehurek.com/gensim/

41. Noh, Y., Park, S., & Park, S.-B. (2019). Aspect-Based Sentiment Analysis Using Aspect Map. *Applied Sciences*, *9*(16), Article 3239. https://doi.org/10.3390/app9163239

42. Sarhan, I., & Spruit, M. (2020). Can We Survive without Labelled Data in NLP? Transfer Learning for Open Information Extraction. *Applied Sciences*, *10*(17), Article 5758. https://doi.org/10.3390/app10175758

Стаття надійшла до редакції 28.08.2020