DOI: https://doi.org/10.15276/aait.05.2022.24
UDC 004.912

Three language political leaning text classification using natural language processing methods

Yurii A. Kosiv[1)]
ORCID: https://orcid.org/0000-0001-7412-2025; yurii.kosiv.mknssh.2021@lpnu.ua
Vitaliy S. Yakovyna[1)]
ORCID: https://orcid.org/0000-0003-0133-8591; vitaliy.s.yakovyna@lpnu.ua. Scopus Author ID: 6602569305
[1)] Lviv Polytechnic National University, 12, Bandery Str. Lviv, 79013, Ukraine

ABSTRACT

In this article, the problem of political leaning classification of the text resource is solved. First, a detailed analysis of ten studies on the work's topic was performed in the form of comparative characteristics of the used methodologies. Literary sources were compared according to the problem-solving methods, the learning that was carried out, the evaluation metrics, and according to the vectorizations. Thus, it was determined that machine learning algorithms and neural networks, as well as vectorization methods TF-IDF and Word2Vec, were most often used to solve the problem. Next, various classification models of whether textual information is pro-Ukrainian or pro-Russian were built based on a dataset containing messages from social media users about the events of the large-scale Russian invasion of Ukraine from February 24, 2022. The problem was solved with the help of Support Vector Machines, Decision Tree, Random Forest, Naïve Bayes classifier, eXtreme Gradient Boosting and Logistic Regression machine learning algorithms, Convolutional Neural Networks, Long short-term memory and BERT neural networks, techniques for working with unbalanced data Random Oversampling, Random Undersampling , SMOTE and SMOTETomek, as well as stacking ensembles of models. Among the machine learning algorithms, LR performed best, showing a macro F1-score value of 0.7966 when features were transformed by TF-IDF vectorization and 0.7933 when BoW. Among neural networks, the best macro F1-score value of 0.76 was obtained using CNN and LSTM. Applying data balancing techniques failed to improve the results of machine learning algorithms. Next, ensembles of models from machine learning algorithms were determined. Two of the constructed ensembles achieved the same macro F1-score value of 0.7966 as with LR. Ensembles that was able to do so consisted of the TF-IDF vectorization, the B-NBC meta-model, and the SVC, NuSVC LR, and SVC, LR base models, respectively. Thus, three classifiers, the LR machine learning algorithm and two ensembles of models, which were defined as a combination of existing methods of solving the problem, demonstrated the largest macro F1-score value of 0.7966. The obtained models can be used for a detailed review of various news publications according to the political leaning characteristic, information about which can help people identify being isolated by a filter bubble.

Keywords: Text classification; political leaning; machine learning algorithms; neural networks; ensembles of models; natural language processing.

***For citation*: Kosiv Y. A., Yakovyna V. S. "Three language political leaning text classification using natural language processing methods". *Applied Aspects of Information Technology.* 2022; Vol.5 No.4: 359–370. DOI: https://doi.org/10.15276/aait.05.2022.24**

INTRODUCTION. FORMULATION OF THE PROBLEM

The Internet, social networks and messengers have become an integral part of human life. There are thousands of news resources, thousands of pages in social networks that publish various news. It is physically impossible to monitor all the available sources, so a small part of the resources is selected, from which the world is known.

When a person follows certain news resources, recommendation systems suggest similar ones. In this way, the number of sources that a person follows increases, and the risk of falling into a filter bubble also increases. A filter bubble is when a person consumes information one-sidedly, because the resources they follow are similar, and they share the same point of view, political trend or position. In this way, a person is isolated from other points of view, opinions, etc.

It is worth noting the role of news resources in the modern world as a whole. Now is an interesting and at the same time difficult time, there are pandemics, wars, crises, Russia still exists, which has always been and always remains a threat to the civilized world. Thus, it is clear that it is not physically possible to search for all the primary sources on your own due to the large number of events. Because of this people begin to follow news resources in which the dedicated team searches, prepares, and publishes the material.

News resources shape people's worldview, but they can also manipulate their readers, covertly carry out campaigns to tune the audience against one or another political trend. Since the political leaning of the resource is difficult to determine

independently sometimes, it is easy to fall into the filter bubble by it, which, in this period of time, is especially dangerous for life, because it is the people from the bubbles who are easily manipulated and can be called to various actions and protests. So, there is a need for a tool that will allow a thorough analysis of all publications of the resource and draw a conclusion about its political leaning. This, in turn, can help identify being isolated by a filter bubble. The basis of such a tool is the classification model of political leaning based on textual information.

Thus, **the purpose of this study** is to solve the problem of classification of the political leaning of the text resource.

The task of the study is to build a new classifier to improve the quality of political leaning classification of the social media texts in three languages, viz., Ukrainian, English and Russian, using natural language processing approach.

The object of this study is political leaning classification.

The subject of the study is methods of implementation of political leaning classification.

1. LITERATURE REVIEW

The scientometric database Scopus was used to search for scientific sources on the research topic. Scopus allows to form a query, which consists in applying various search functions, combining them with logical operators and other criteria. Thus, ten studies on the political leaning classification were selected.

After that, a detailed review of each selected literary source was carried out. The main points that were paid attention to during the analysis were: the type of algorithm, whether there was a supervised or unsupervised learning, the algorithms that were used to solve the problem, vectorization methods, algorithm comparison metrics and obtained results of the developed models.

The summary of the methodologies used in the ten related works are given in Table 1.

According to the table, it can be seen that the supervised methods were most often used for the classification of political leaning, they were used in all ten reviewed articles. Unsupervised methods were tested only in a study [10], but they failed to improve results.

Regarding the methods themselves, it can be seen from the table that neural networks were applied in seven out of ten cases, of which five times showed the best results, which confirms the suitability of using neural networks for the classification of political inclination. It should also be noted the popularity of testing the SVM method, it was used in six out of ten articles, but showed the best value of indicators in only one, namely in [3]. It is also worth noting the performance of the pre-trained BERT model, it was applied in two studies, namely [4] and [8], and in both it showed the best metrics values.

Table 1. **Methodologies overview**

Work	Methods	Learning	Metrics	Vectorizations
[1]	LSTM, SVM, DTC, LR, RFC	Supervised	Precision, recall, F1-score	TF-IDF
[2]	MLR, K-NN, DTC, RFC, SVM, MLP	Supervised	Accuracy, precision, recall, F1-score	BoW, HV, TF, TF-IDF
[3]	SVM, K-NN, DTC, XGBoost, LR, NBC, RFC	Supervised	Accuracy, precision, recall, F1-score	TF, TF-IDF
[4]	SVM, NN, CNN, BERT	Supervised	Accuracy, precision, recall	TF-IDF, Word2Vec One-Hot,
[5]	FastText, TextCNN, HAN, R-CNN	Supervised	Accuracy, precision, recall, F1-score	-
[6]	FastText, TextCNN, HAN R-CNN	Supervised	Precision, recall, F1-score	-
[7]	RFC, SVM, Vanilla Deep Multi-Task, Attention Deep Multi-Task, Hierarchical Attention Deep Multi-Task	Supervised	F1-score	-
[8]	CNN, BERT	Supervised	Accuracy, F1-score	Word2Vec
[9]	K-NN	Supervised	Accuracy, precision, recall, F1-score	TF-IDF
[10]	K-means, GaussianMixture, MeanShift, SVM	Supervised and unsupervised	Precision, recall, F1-score	Word2Vec

Source: **compiled by the authors**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

Regarding metrics, we can note the traditional set consisting of accuracy, precision, recall and F1-score. Of the vectorization methods, the most popular in the conducted studies were TF-IDF and Word2Vec, TF-IDF vectorization was used in five out of ten articles, Word2Vec in three out of ten.

Among the features of the research, we should note the work [1] and its evaluation of models on four feature vectors: TF, TF-IDF-unigram, TF-IDF-bigram, TF-IDF-trigram. The study also calculated the training time, according to which it was determined that the random forest (RFC) learned 96 times faster than LSTM, 0.8587 seconds and 82.7816 seconds, respectively. In addition, it is worth noting the article [2] and its testing of feature vectors formed on nouns, which helped the MLR and MLP models achieve the highest accuracy value of 89%. It is also necessary to highlight the research [3] and its classification experiments on a dataset with and without stop words. Thus, the maximum obtained accuracy was improved by almost 1 %. On the dataset with stop words, the SVM model achieved an accuracy of 92.08 %, and without stop words, the XGBoost model demonstrated 92.82 %. Regarding the features of the article [5], the comparison of the learning speed should be emphasized, according to which it was determined that the R-CNN model learned the fastest with the value of epochs equal to 9, followed by HAN at 16, then TextCNN at 24 and FastText at 35 epochs. Among the features of the study [10], it is necessary to note the focus on solving the classification problem using unsupervised learning methods. The highest macro precision value was 80% and was obtained by the MeanShift algorithm, but this result was surpassed by the SVM teacher training method with an achieved macro precision of 87%.

2. RESEARCH METHODOLOGY

2.1. Text preparation techniques

Before modeling, the text goes through a preparation stage, during which various cleaning and modification operations take place. This is done to remove information that has little or no relevance to the modelling, such as punctuation. The main techniques of text preparation are: tokenization, removal of stop words, stemming, lemmatization and selection of parts of speech, work with the frequency of occurrence of words in the text [11].

Tokenization is the process of breaking sentences and paragraphs into smaller units, called tokens, which are easier to work with. Typically, tokens are obtained by separating sentences with spaces, but some libraries provide tools where the received tokens are both words and punctuation marks.

Stop word removal is the process of removing words that are considered uninformative. Depending on the language of the text, stop words may differ. Thus, in Ukrainian, stop words are, for example, “і”, «або» and «чи», in English, in turn, they are i.e. “a”, “an” and “the”. So, depending on which languages the text is made of, a list of stop words is formed by connecting the stop words of a specific language, after which the text is cleaned by checking whether the token is not in the formed list.

Stemming is the process of reducing a word to its base by removing a suffix. Thus, words that mean the same thing, but are written differently due to the specifics of the language, will be reduced to one word.

Lemmatization is a more complex approach to reducing a word to its base than stemming. The first step in this text preparation technique is to identify the part of speech of the word. Such information is very important and directly affects the correctness of the modification. Thus, knowing that “is” and “are” are verbs, they will be changed to “be”. In addition, lemmatization takes into account language exceptions, such as changing “better” to “good” and “children” to “child”.

To compare stemming and lemmatization, we may have an example of converting words such as “studies” and “studying”, after stemming we will get “studi” and “study”, and after lemmatization “study” and “study”, which is more grammatically correct.

Determining the frequency of occurrence of words in a text is a process, the result of which is the value of occurrence of tokens in a dataset. This information can be further used to remove those words that occur very rarely and very often. Thus, the cleaning of uninformative words allows to reduce the number of features, which, in turn, affects the training time and the quality of the models.

2.2. Vectorization methods

For modeling, the cleaned text must be converted into numbers. Vectorization allows such a conversion. There are many techniques, but Bag of Words, TF-IDF and Word2Vec can be highlighted.

Bag of Words is a way of presenting text in the form of a vector of fixed length, which consists of values of the frequency of occurrence of words. To do this, first, a dictionary is defined, which contains all the unique words of the dataset. The size of the dictionary corresponds to the size of the final vector, since each word in the dictionary is a column by which the result is formed. Next, to convert the text into a vector, the frequency of occurrence will be calculated for each word. For example, if the dictionary contains the words “the”, “bike”, “bus” and

"car", then the text "the bike and the car" will be represented as [2,1,0,1], since "the" occurs in the text twice, "bike" and "car" once, and "bus" never. It is worth noting that Bag of Words shows what words are in the document, and not exactly where they are, that is, the place of words in the text is not saved. In addition, it should be noted that the features of the vector can be not just words, but N-grams, then, for example, for bigrams, the dictionary would contain the following features: "the bike", "bike bus" and "bus car".

TF-IDF (term frequency – inverse document frequency) is a vectorization method that also has a dictionary, but the TF-IDF formula is used to determine the values in the vector. It consists of two parts, namely TF (term frequency) and IDF (inverse document frequency) [12]. TF is the ratio of the frequency of occurrence of a certain word in the text to the total number of words in the text.

The TF formula is given below:

$$TF = \frac{n_i}{\sum_k n_k}, \quad (1)$$

where n_i – the value of occurrences of the word in the text.

IDF is the inverse of the frequency with which a certain word occurs in a corpus of texts. The IDF formula is given below:

$$IDF = \log\left(\frac{N}{df_i}\right), \quad (2)$$

where N – is the total number of corpus texts; df_i – the number of texts containing a word.

The final TF-IDF formula has the form: $TF - IDF = TF * IDF$, so words that appear more often in a certain document and less often in other documents of the corpus will have a greater TF-IDF value. Words that occur in all texts will have a zero value, since $\log\left(\frac{N}{N}\right) = \log(1) = 0$.

Word2Vec is a vectorization method that uses a neural network model to learn associations between words in a text corpus. Thus, similar words will be next to each other in space. The dimensionality of the space is determined by the total number of words or can be set manually, usually choosing a value between 100 and 1000. It is worth noting that Word2Vec allows us to get a vector for a specific word, however, to get a vector for the data row, we need to perform additional operations. Such operations can be the sum of vectors, mean of vectors, etc. So, a separate vector will be obtained for each word of the text, then a list of vectors will be formed, after which a certain operation will be applied to this list and the final representation vector will be obtained.

It is worth noting that pre-trained vectorization models are also often used, which are presented in the form of a key-value, where the key is a word, and the value is the vector of this word. Training of such models is usually carried out on a large amount of data, which is especially useful when the size of the dataset is small and there is no possibility to train the vectorization model manually.

2.3. Metrics for evaluating the performance of models

Accuracy, macro F1-score, macro precision and macro recall were used to evaluate the performance of the classification models.

2.4. Machine learning algorithms

Support Vector Machines (SVM) are a family of supervised machine learning algorithms for solving classification and regression problems. SVM represents the rows of a dataset as points in space such that individual classes are separated by some plane that offers the largest gap between classes. The simplest version of SVM supports binary classification, but modern SVMs allow multiclass classification by dividing it into several binaries [13]. There are various SVM implementations for the classification task, such as: Linear Support Vector Classifier (LinearSVC), Support Vector Classifier (SVC), Nu-Support Vector Classifier (NuSVC), etc. They differ in kernel and other hyperparameters, as well as in their purpose, such as LinearSVC is recommended for use on large datasets, and SVC, on the contrary, on small ones, up to tens of thousands of records.

Logistic Regression (LR) is a supervised machine learning algorithm for solving classification and regression problems. It is worth noting that the algorithm is widely used to perform binary classifications. The logic of the algorithm consists in calculating the probability that a row of data belongs to a certain class using a set of parameters, the value of which is determined by the maximum likelihood method, according to which such parameters are selected that maximize the value of the likelihood function on the dataset.

Decision Tree (DT) is a supervised machine learning algorithm used to solve classification and regression problems. The logic of the algorithm consists in using features of the dataset to create yes/no questions, according to which the decision tree itself is built [14]. Thus, the leaves of the tree represent classes, and the branches represent the sequence of attribute values that led to this class. Decision trees are easy to interpret, but they are capable of overfitting, especially when the constructed tree is very tall.

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

Random Forest (RF) is a supervised ensemble machine learning method for classification and regression. Its logic consists in building a large number of decision trees and combining their results. Thus, a particular row of the dataset will be assigned the class that was classified by the majority of trees in the random forest. The advantage of this algorithm compared to decision trees is more accurate predictions and less prone to overfitting.

Naïve Bayes Classifier (NBC) is a family of probabilistic classifiers that use Bayes' theorem to determine the probability that a row of a dataset belongs to one of the classes. Depending on the assumption about the distribution of features, there are different implementations of NBC, such as: Bernoulli Naïve Bayes Classifier (B-NBC), Multinomial Naïve Bayes Classifier (M-NBC), Gaussian Naïve Bayes Classifier (G-NBC), etc. [15]. In the case of B-NBC, the features are assumed to be distributed according to the Bernoulli distribution, in M-NBC according to the polynomial distribution, and in G-NBC according to the normal distribution, also known as the Gaussian distribution.

Extreme Gradient Boosting (XGBoost) is an ensemble machine learning method for solving classification and regression problems. XGBoost is an implementation of gradient boosting trees with some improvements, such as automatic feature selection, application of Newton's method for optimization, use of regularization to prevent overfitting, implementation of parallel tree construction, etc. [16]. After winning various machine learning competitions, this method began to be widely used, integrated into all kinds of libraries and implemented in various programming languages.

2.5. Hyperparameter optimization methods

Models have parameters and hyperparameters, the difference between them is that the parameters are determined from the dataset during training, while the hyperparameters are fixed, they are set manually before training and actually help determine the values of the parameters.

The model can have various hyperparameters, but choosing the best combination of them on your own usually takes a lot of time. That is why there are various hyperparameter optimization techniques that help to determine such a combination that, for a certain model and dataset, shows the best results [17], [18].

For this work, grid search was used, which is considered one of the simplest. It accepts lists of values for each hyperparameter, then trains models with all possible combinations. Grid search is easy to understand, but quite time-consuming.

2.6. Ensembles of models

An ensemble of models is a supervised machine learning method that allows to combine multiple models during training and making predictions. A single model can achieve certain results, however, if a combination of models is used, there is a chance to obtain improved metrics, which is the motivation behind the ensemble of models method.

There are simple and complex types of ensembles, the simple ones are max voting, averaging and weighted averaging, and the complex ones are bagging, boosting, stacking, blending [19]. Stacking ensembles were used in this work.

Stacking is an ensemble technique that uses model predictions to build a new model. The logic of stacking is to transform a dataset by training a list of models on it and using their predictions as new features. Let there be two models, then the stacking will look like this: first, the training data is divided into N parts, for example into five, then the first model is trained on four subsets and makes a prediction on the fifth, this is repeated for all parts. After that, the first model is trained on the full set of training data and makes a prediction for the test data. A similar process for the second model. In this way, a dataset with two features that correspond to the predictions of the two models was obtained. The next step of stacking is the construction of the final model on the transformed dataset, which will make the final predictions. In this example, the first two models are called base, or zero-level models, and the final model is called the meta-model, or first-level model. It is worth noting that the number of base models and metamodels, as well as the number of levels in stacking, can be arbitrary.

2.7. Neural networks

RNN (recurrent neural networks) is a neural network that, thanks to its architecture, allows to store the context, the so-called "short-term memory" [20]. In RNN, the result of the previous step comes to the input of the current one, which, in turn, passes its result on, which is the implementation of internal memory. Neural networks with context are particularly useful when working with data in the form of sequences, such as text, where, for example, to predict the next words, information about the previous ones is critical. The disadvantage of RNNs is that they have a fairly short internal memory, which means that the data that was processed at the beginning of the neural network has almost no effect on the result at the end.

LSTM (long short-term memory networks) is a neural network that is considered an extension of

RNN. It has a more complex architecture, allowing it to store more context information. Thus, the neural network has the ability to determine and remember long-term dependencies between data [21].

CNN (convolutional neural network) is a neural network that is widely used to work with images. CNN is based on convolutional and aggregation layers [22]. The convolution layer performs a convolution operation on the image, thanks to which it is possible to reduce the number of parameters and highlight the most important features. The aggregation layer does something similar, but instead of a convolution operation, it applies a kernel, which selects the value from the kernel that is defined by the strategy, it can be the maximum value, the average, etc. Despite the fact that the typical data format of CNN is an image, it is also able to work with text, to implement this, the text is converted into vectors of numbers, that is, vectorization is performed.

BERT (bidirectional encoder representations from transformers) is a machine learning technique based on the architecture of the deep learning model transformer [23]. BERT is developed and widely used by Google. The original English-language BERT models, namely $BERT_{BASE}$ and $BERT_{LARGE}$, were trained on a dataset of 800 million words and on the English Wikipedia with 2500 million words. There are many implementations of BERT, the difference between which is the dataset on which they were trained, it can be datasets of one language, it can be of many languages. In addition, there are various variations of the BERT architecture, such as: the DistilBert model, which contains 40% fewer parameters and is a 60% faster version of BERT, the RoBERTa model, which is an improved version of BERT, and many others [24], [25].

2.8. Balancing data techniques

An unbalanced dataset is data in which the number of rows of one class significantly exceeds the number of rows of others. Since there are many problems where the classes of interest are precisely the minority classes, there are various ways of working with such datasets. There are three approaches, namely: oversampling; undersampling, hybrid [26].

Oversampling is an approach to working with unbalanced datasets, when the number of minority class rows is increased. Oversampling implementation methods are: Random Oversampling, SMOTE.

Random Oversampling is a way of implementing oversampling, in which minority rows are duplicated in order to balance the dataset. A characteristic feature of this method is that the selection of lines is carried out randomly.

SMOTE (Synthetic Minority Oversampling Technique) is an oversampling implementation method that, instead of duplicating, synthetically creates new rows of data that are similar to, but not a match to, existing minority classes.

Undersampling is an approach to unbalanced data in which the number of rows in the majority class is reduced. The methods of implementing undersampling include the following: Random Undersampling, Near-miss, Tomek links.

Random Undersampling is a method of undersampling in which the rows of the majority class to be removed are randomly selected.

Near-miss is a way of implementing undersampling that works with the distribution of values in a dataset to determine which rows will be removed for balancing.

Tomek links is an undersampling method that uses so-called “Tomek links" to select data of the majority class to be removed. These links between data allow to identify rows of different classes that are similar to each other. Thus, after applying this method, in addition to balancing the data, noise is also eliminated.

Hybrid is an approach that tries to combine oversampling and undersampling. Among the hybrid methods, it is worth highlighting **SMOTETomek**, which is a combination of the SMOTE oversampling approach and the Tomek links undersampling approach. Thus, using this method increases the dataset thanks to SMOTE and cleans it with the help of Tomek links.

3. EXPERIMENTAL RESULTS

3.1. Dataset

The dataset contained messages from social media users regarding the events of the large-scale Russian invasion of Ukraine from February 24, 2022. It consisted of 13,000 records and 29 columns, from which it is worth highlighting “Text digest” and “Political view”, which are text information and the political view of the message, respectively. There are two possible values of the political view, namely: pro-Ukrainian, pro-Russian.

It should be noted that the data sample contained words from different languages, but a decision was made to focus on only three, namely: Ukrainian, English, and Russian, which affected further processing.

First, rows with empty values, duplicates, and outliers were removed, after which the text of the message, i.e., the “Text digest” column, was processed using the following steps:

1. Updating text to be in the lower case;
2. Removal of links, hashtags and apostrophes;

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

3. Replacement of special characters, such as “ш” for “ш” and “$_р$” for “р”;

4. Splitting the text into tokens;

5. Removal of punctuation characters, as well as short and long words, the length of which was less than three characters and more than 21 characters, respectively;

6. Removal of tokens that are stop words of the Ukrainian, English or Russian languages;

7. Removal of words that contained symbols that do not exist in the alphabets of the Ukrainian, English and Russian languages;

8. Determining the language of a word using the Python library langdetect and removing tokens that are not Ukrainian, English or Russian;

9. Depending on the language of the word, launching the corresponding tool, which, for a certain token, performs the definition of a part of the speech and, according to the received information, performs lemmatization on it;

10. Repeating steps 1-6, since, since after lemmatization, some words were transformed as “сімя” to “Сім’я” and “київ” to “Київ”.

As a result, the size of the dataset became equal to 9358 records.

3.2. Machine learning algorithms

The problem of political leaning classification was solved using the following machine learning algorithms: Linear Support Vector Classifier (LinearSVC), Support Vector Classifier (SVC), Nu-Support Vector Classifier (NuSVC), Decision Tree Classifier (DTC), Random Forest Classifier (RFC), Bernoulli Naïve Bayes Classifier (B-NBC), Multinomial Naïve Bayes Classifier (M-NBC), Extreme Gradient Boosting (XGBoost), Logistic Regression (LR).

For each algorithm listed above, the search for the best hyperparameters was carried out using the grid search optimization method, which in the implementation of the scikit-learn library includes k-fold cross-validation with $k = 5$ by default. The results of the models were evaluated by accuracy, macro F1-score, macro precision and macro recall.

The scikit-learn and xgboost libraries containing LinearSVC, SVC, NuSVC, DTC, RFC, B-NBC, M-NBC, LR and XGBoost implementations were used, respectively [27], [28]. In addition to machine learning algorithms, hyperparameters were also searched for the vectorization methods.

Thus, the search for the best hyperparameters of each of the nine algorithms was carried out in combination with the search for the best hyperparameters of vectorization methods. The metrics for different algorithms are listed in Table2.

According to the table, it can be noted that the vectorization of Word2Vec significantly showed worse performance than BoW and TF-IDF. The macro F1-score was defined as the metric used to determine the best model, as it allows demonstrating the model effectiveness across all classes. Thus, the LR algorithm with TF-IDF vectorization performed best, which achieved a macro F1-score value of 0.7966. The second-best metric result of 0.7933 was also obtained using LR, but with the BoW vectorization method.

3.3. Neural networks

Next, neural networks were used to solve the problem of political leaning classification. The first network was CNN #1 with the following architecture:

- Embedding layer, vector size is 300;
- Conv1D layer, 32 filters, kernel size 3;
- MaxPooling1D layer, kernel size 2;
- Flatten layer;
- Dense layer, 250 neurons, ReLU activation;
- Dense layer, 1 neuron, sigmoid activation.

After that, the neural network CNN #2 was defined, which had the architecture:

- Embedding layer, vector size equal to 200;
- Conv1D layer, 128 filters, kernel size 5;
- GlobalMaxPooling1D layer, kernel size 2;
- Dense layer, 10 neurons, ReLU activation;
- Dense layer, 1 neuron, sigmoid activation.

Next, an LSTM neural network is built:

- Embedding layer, vector size is 100;
- Bi-LSTM layer, 64 neurons;
- Dense layer, 32 neurons, ReLU activation;
- Dense layer, 1 neuron, sigmoid activation.

Then a neural network consisting of CNN and LSTM was defined:

- Embedding layer, vector size is 100;
- Conv1D layer, 64 filters, kernel size 3;
- MaxPooling1D layer, kernel size 2;
- LSTM layer, 100 neurons;
- Dense layer, 64 neurons, ReLU activation;
- Dense layer, 1 neuron, sigmoid activation.

After that, two neural networks were tested using BERT. DistilBERT base multilingual and XLM-RoBERTa from the Hugging Face service were used [29], [30]. The choice of these models is justified by their training on texts of more than one hundred languages. Both networks had the following architecture:

- BERT layer;
- Dense layer, 1 neuron, sigmoid activation.

Table 2. **Results of machine learning algorithms**

Algorithm	Vectorization	Accuracy	F1-score	Precision	Recall
LinearSVC	BoW	0.9214	0.7813	0.799	0.7664
	TF-IDF	0.8719	0.7459	0.7075	0.8355
	Word2Vec	0.6846	0.569	0.5926	0.7287
SVC	BoW	0.9228	0.7656	0.8185	0.7314
	TF-IDF	0.9275	0.7918	0.8231	0.7679
	Word2Vec	0.823	0.5289	0.5294	0.5287
NuSVC	BoW	0.9211	0.7458	0.825	0.7039
	TF-IDF	0.9243	0.7754	0.8177	0.7459
	Word2Vec	0.8963	0.6724	0.7203	0.6463
DTC	BoW	0.8987	0.71	0.7332	0.6928
	TF-IDF	0.8979	0.7024	0.7288	0.6836
	Word2Vec	0.8256	0.5614	0.5589	0.5646
RFC	BoW	0.8806	0.7409	0.7099	0.7962
	TF-IDF	0.8925	0.7527	0.7269	0.7915
	Word2Vec	0.8958	0.5707	0.7212	0.5547
B-NBC	BoW	0.8763	0.7263	0.699	0.7726
	TF-IDF	0.8763	0.7263	0.699	0.7726
	Word2Vec	0.8658	0.6295	0.6363	0.6239
M-NBC	BoW	0.9238	0.7855	0.8072	0.7678
	TF-IDF	0.9065	0.7689	0.7547	0.7861
XGBoost	BoW	0.9235	0.7431	0.8502	0.694
	TF-IDF	0.9203	0.7312	0.8342	0.6848
	Word2Vec	0.8988	0.5374	0.8294	0.5338
LR	BoW	0.9245	0.7933	0.8059	0.7824
	TF-IDF	0.9226	0.7966	0.7963	0.7971
	Word2Vec	0.7642	0.5986	0.6	0.6827

Source: **compiled by the authors**

The performance metrics for different types of neural networks are shown in Table3.

According to the Table 3, it is possible to note the disappointing results of neural networks with BERT, as XLM RoBERTa showed a macro F1-score equal to 0.48, and DistilBERT 0.69, which are the worst values. The best result was obtained by CNN #2 and LSTM neural networks with an F1-score value of 0.76.

3.4. Working with unbalanced data

Since the dataset contains 88.8% of pro-Ukrainian records and 11.2% of pro-Russian records, it can be concluded that the classes of the dataset are unbalanced. This was the reason for testing methods of working with unbalanced data such as Random Oversampling, Random Undersampling, SMOTE and SMOTETomek. These techniques were applied together with the best LR and SVC machine learning algorithms and BoW and TF-IDF vectorizations in the grid search hyperparameter optimization method. The performance metrics for different models after data balancing are given in Table 4.

Table 3. **Results of neural networks**

Neural network	Accuracy	F1-score	Precision	Recall
CNN #1	0.9	0.75	0.75	0.75
CNN #2	0.92	0.76	0.84	0.72
LSTM	0.91	0.76	0.78	0.75
CNN і LSTM	0.91	0.75	0.77	0.73
DistilBERT base multilingual	0.91	0.69	0.87	0.64
XLM RoBERTa	0.89	0.48	0.94	0.5

Source: **compiled by the authors**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

Table 4. **Results of applying data balancing techniques**

Model	Accuracy	F1-score	Precision	Recall
LR, TF-IDF, Random Oversampling	0.9213	0.7908	0.8104	0.7755
LR, TF-IDF, Random Undersampling	0.8158	0.6963	0.6689	0.8268
LR, TF-IDF, SMOTE	0.9222	0.794	0.8118	0.7797
LR, TF-IDF, SMOTETomek	0.9217	0.7933	0.8106	0.7794
SVC, TF-IDF, Random Oversampling	0.919	0.7833	0.8039	0.767
SVC, BoW, Random Undersampling	0.8265	0.704	0.6737	0.8229
SVC, TF-IDF, SMOTE	0.9191	0.7837	0.8042	0.7675
SVC, TF-IDF, SMOTETomek	0.9191	0.7837	0.8042	0.7675

Source: **compiled by the authors**

According to the table, it can be noted that the implementation of data balancing did not improve the results of the algorithms, but only reduced them by 1-2% according to the macro F1-score. However, it is worth highlighting the application of SMOTE, which demonstrated the best F1-score metric value of 0.794, which was only 0.2% less than the achieved LR result with TF-IDF without SMOTE.

3.5. Ensembles of models

The next step was to create stacking ensembles of models using the best machine learning algorithms. With the help of grid search, the combination of the vectorizations, its hyperparameters, basic models, and meta-models was sorted out to find the combination that would demonstrate the best metrics values. The architecture of the constructed ensembles is summarized in Table5, while the performance metrics for these ensembles are listed in Table 6.

It can be seen from Table 6 that the best result of macro F1-score equal to 0.7966 was achieved by two ensembles stacking #3 and stacking #5. The next step in working with the ensembles was to try to improve the results of the models by testing different combinations of metamodel hyperparameters. The obtained results are presented in Table7.

According to Table 7, it can be noted that it was possible to improve only the metrics values of stacking #6, but not stacking #5. Accordingly, it can be noted that the achieved macro F1-score value equal to 0.7966 by ensembles of models corresponds to the best value of machine learning algorithms, namely the result from the LR model with TF-IDF.

Table 5. **Ensembles of models**

Name of ensemble	Vectorization	Base models	Meta-model
Stacking #1	BoW	LinearSVC, M-NBC, LR	B-NBC
Stacking #2	BoW	LinearSVC, LR	B-NBC
Stacking #3	TF-IDF	SVC, NuSVC, LR	B-NBC
Stacking #4	TF-IDF	SVC, NuSVC, LR	LinearSVC
Stacking #5	TF-IDF	SVC, LR	B-NBC
Stacking #6	TF-IDF	SVC, LR	SVC
Stacking #7	TF-IDF	SVC, LR	LinearSVC

Source: **compiled by the authors**

Table 6. **Results of ensembles of models**

Ensemble	Accuracy	F1-score	Precision	Recall
Stacking #1	0.9217	0.7902	0.811	0.7732
Stacking #2	0.9119	0.7804	0.7786	0.7828
Stacking #3	0.9246	0.7966	0.822	0.7769
Stacking #4	0.9261	0.7848	0.8428	0.748
Stacking #5	0.9246	0.7966	0.822	0.7769
Stacking #6	0.9179	0.7924	0.795	0.7909

Source: **compiled by the authors**

Table 7. **Results of an attempt to improve model ensembles**

Ensemble	Accuracy	F1-score	Precision	Recall
Stacking #5	0.9246	0.7966	0.822	0.7769
Stacking #6	0.9175	0.7931	0.7932	0.7938

Source: **compiled by the authors**

CONCLUSIONS

Therefore, in this study, the existing solutions to the problem of political leaning classification were analyzed, and then machine learning algorithms, neural networks, and techniques for working with unbalanced data, vectorization methods, and ensembles of models were used to solve the problem. Among the machine learning algorithms, the best results were demonstrated by SVC and LR with BoW and TF-IDF vectorizations. The Word2Vec vectorization method, in turn, was worse than BoW and TF-IDF for all algorithms. The highest macro F1-score value of 0.7966 was achieved by the LR with TF-IDF vectorization, followed by 0.7933 from algorithm LR with BoW and 0.7918 from SVC with TF-IDF.

After that, there was an attempt to solve this problem with CNN, LSTM and BERT neural networks. The largest macro F1-score value of 0.76 was obtained by CNN and LSTM networks. Regarding BERT, the values achieved by the two networks of 0.69 and 0.48 were the lowest among neural networks. Thus, it can be concluded that machine learning algorithms coped better with this task compared to neural networks, as they demonstrated higher results by 2-3%.

In addition, since the dataset was unbalanced, various data balancing techniques were tested with the best machine learning algorithms. However, the results did not improve and remained at the same level. The highest macro F1-score value of 0.794 was achieved using LR with TF-IDF and SMOTE.

Next, stacking ensembles of models were created from BoW and TF-IDF vectorizations and the best machine learning algorithms. The largest macro F1-score value of 0.7966 was achieved by two ensembles with TF-IDF vectorization, B-NBC metamodel and SVC, NuSVC, LR and SVC, LR base models, respectively. This was followed by a search for the best hyperparameters of the metamodel for some ensembles with the aim of metrics values improvement, but it was not possible to obtain a higher value of the macro F1-score.

Thus, by combining the existing solutions to the problem, three new classifiers were built, the LR machine learning algorithm and two ensembles of models, which demonstrated the largest value of the macro F1-score equal to 0.7966 and coped best with the task of political leaning classification.

Options for future experiments to improve results may include creating new ensembles, testing other machine learning algorithms, vectorizations, more complex neural networks, data balancing techniques, pre-trained models. In addition, different dataset preprocessing techniques can be testes, such as the use of various stemming and lemmatization algorithms, the creation of a dataset with only nouns, with nouns and verbs, with only Ukrainian words, etc.

REFERENCES

1. Ansari, M. Z., Aziz, M. B., Siddiqui, M. O., Mehra H. & Singh, K. P. “Analysis of political sentiment orientations on twitter,” *Procedia Computer Scienc*. 2020; 167: 1821–1828. DOI: https://doi.org/10.1016/j.procs.2020.03.201.

2. Di Giovanni, M., Brambilla, M., Ceri, S., Daniel, F. & Ramponi, G. “Content-based classification of political inclinations of twitter users”. In *2018 IEEE International Conference on Big Data (Big Data).* Seattle: WA USA. 2018. p. 4321–4327. DOI: https://doi.org/10.1109/BigData.2018.8622040.

3. Ullah, H. et al. “Comparative study for machine learning classifier recommendation to predict political affiliation based on online reviews”. *CAAI Trans on Intel Tech.* Sep. 2021; 6 (3): 251–264. DOI: https://doi.org/10.1049/cit2.12046.

4. Chun, S., Holowczak, R., Dharan, K., Wang, R., Basu, S. & Geller, J. “Detecting political bias trolls in twitter data”. In *Proceedings of the 15th International Conference on Web Information Systems and Technologies.* Vienna: Austria. 2019. p. 334–342. DOI: https://doi.org/10.5220/0008350303340342.

5. Alzhrani, K. M. “Political ideology detection of news articles using deep neural networks”. *Intelligent Automation & Soft Computing.* 2022; 33 (1): 483–500, DOI: https://doi.org/ 10.32604/iasc.2022.023914.

6. Alzhrani, K. M. “Politicians-based deep learning models for detecting news, authors and media political ideology”. *IJACSA.* 2022; 13 (2). DOI: https://doi.org/10.14569/IJACSA.2022.0130286.

7. Vijayaraghavan, P., Vosoughi, S. &. Roy, D “Twitter demographic classification using deep multimodal multi-task learning”. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics.* Vancouver: Canada. 2017; 2: 478–483. DOI: https://doi.org/10.18653/v1/P17-2076.

8. Bilbao-Jayo A. & Almeida, A. “Improving political discourse analysis on twitter with context analysis *IEEE Access.* 2021; 9: 104846–104863. DOI: https://doi.org/10.1109/ACCESS.2021.3099093.

9. Gu, F. & Jiang, D. “Prediction of political leanings of chinese speaking twitter users”. In *International Conference on Signal Processing and Machine Learning (CONF-SPML).* Stanford: CA, USA. Nov. 2021. p. 286–289. DOI: https://doi.org/10.1109/CONF-SPML54095.2021.00062.

10. Fagni, T. & Cresci, S. “Fine-grained prediction of political leaning on social media with unsupervised deep learning”. *Jair* Feb. 2022; 73: 633–672. DOI: https://doi.org/ 10.1613/jair.1.13112.

11. Agrawal, R. “Must known techniques for text preprocessing in NLP”. *Analytics Vidhya.* Jun. 14, 2021. – Available from: https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp. – [Accessed: 19, Nov. 2021].

12. “How to process textual data using TF-IDF in Python”, *freeCodeCamp.org.* Jun. 06, 2018. – Available from: https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/. – [Accessed 19, Nov. 2021].

13. Baeldung, “Multiclass classification using support vector machines | Baeldung on computer science”. Oct. 07, 2020. – Available from: https://www.baeldung.com/cs/svm-multiclass-classification. – [Accessed: 19, Nov. 2021].

14. Bento, C. “Decision tree classifier explained in real-life: picking a vacation destination”. *Medium.* Jul. 18, 2021. – Available from: https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575. – [Accessed: 19, Nov. 2021].

19. “Naive bayes classifier”. *Wikipedia*. – Available from: https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=1118900065. – [Accessed: 19, Nov. 2021].

20. Morde, V. “XGBoost algorithm: long may she reign!” *Medium.* Apr. 08; 2019. – Available from: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d. – [Accessed: 19, Nov. 2021].

21. Kumar, S. “7 hyperparameter optimization techniques every data scientist should know”. *Medium.* 26, May 2021. – Available from: https://towardsdatascience.com/7-hyperparameter-optimization-techniques-every-data-scientist-should-know-12cdebe713da. – [Accessed: 19, Nov. 2021].

22. “Hyperparameter optimization”. *Wikipedia*. Oct. 04, 2022. – Available from: https://en.wikipedia.org/w/index.php?title=Hyperparameter_optimization&oldid=1114024235. – [Accessed: 19, Nov. 2021].

23. Singh, A. “Ensemble learning | Ensemble techniques”. *Analytics Vidhya*, Jun. 18, 2018. – Available from: https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models. – [Accessed: 19, Nov. 2021].

24. Phi, M. “Illustrated guide to recurrent neural networks”. *Medium.* Jun. 28, 2020. – Available from: https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9. – [Accessed: 19, Nov. 2021].

25. Says, K. L. “What is LSTM – introduction to long short term memory”. *Intellipaat Blog.* May 28, 2020. – Available from: https://intellipaat.com/blog/what-is-lstm. – [Accessed: Nov. 19, 2021].

26. Saha, S. “A comprehensive guide to convolutional neural networks – the ELI5 way”. *Medium.* – Available from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. – [Accessed: 19, Nov. 2021].

27. “BERT (language model)”. *Wikipedia*. 17, Nov. 2021. – Available from: https://en.wikipedia.org/w/index.php?title=BERT_(language_model)&oldid=1122505336. – [Accessed: 19, Nov. 2021].

28. “DistilBERT”. – Available from: https://huggingface.co/docs/transformers/model_doc/distilbert. – [Accessed: 19, Nov. 2021].

28. “RoBERTa”. – Available from: https://huggingface.co/docs/transformers/model_doc/roberta. – [Accessed: 19, Nov. 2021].

29. Brownlee, J. “Random oversampling and undersampling for imbalanced classification”. *MachineLearningMastery.com.* Jan. 14, 2020. – Available from: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/. – [Accessed: 19, Nov. 2021].

30. “Scikit-learn: machine learning in python – scikit-learn 1.1.3 documentation”. – Available from: https://scikit-learn.org/stable. – [Accessed: 19, Nov. 2021].

31. “XGBoost documentation – xgboost 1.7.1 documentation”. – Available from: https://xgboost.readthedocs.io/en/stable. – [Accessed: 19, Nov. 2021].

32. “Distilbert-base-multilingual-cased Hugging Face.” – Available from: https://huggingface.co/distilbert-base-multilingual-cased. – [Accessed: 19, Nov. 2021].

33. “Jplu/tf-xlm-roberta-base Hugging Face”. – Available from: https://huggingface.co/jplu/tf-xlm-roberta-base. – [Accessed: 19, Nov. 2021].

Conflicts of Interest: the authors declare no conflict of interest

Received 05.11.2022
Received after revision 10.12.2022
Accepted 24.12.2022

DOI: https://doi.org/10.15276/aait.05.2022.24
УДК 004.912

Класифікація політичної забарвленості тексту трьома мовами з використанням методів опрацювання природної мови

Косів Юрій Андрійович[1)]
ORCID: https://orcid.org/0000-0001-7412-2025; yurii.kosiv.mknssh.2021@lpnu.ua
Яковина Віталій Степанович[1)]
ORCID: https://orcid.org/0000-0003-0133-8591; vitaliy.s.yakovyna@lpnu.ua. Scopus Author ID: 6602569305
[1)] Національний університет «Львівська політехніка», вул. С. Бандери, 12. Львів, 79013, Україна

АНОТАЦІЯ

У цій статті здійснюється розв’язання задачі класифікації політичної забарвленості текстового ресурсу. Спочатку виконано детальний аналіз десяти досліджень за темою роботи у вигляді порівняльної характеристики інструментарію. Літературні джерела порівнювались за методами розв’язання задач, здійсненим навчанням, метриками оцінки та способами векторизації. Таким чином визначено, що для розв’язання задачі найчастіше використовувались алгоритми машинного навчання та нейронні мережі, а також способи представлення ознак TF-IDF та Word2Vec. Далі було побудовано різноманітні моделі класифікації того, чи текстова інформація є проукраїнською, чи проросійською на основі набору даних, що містив повідомлення користувачів соціальних мереж про події широкомасштабного російського вторгнення в Україну з 24 лютого 2022 року. Розв’язання задачі здійснювалось за допомогою алгоритмів машинного навчання Support Vector Machines, Decision Tree, Random Forest, Naïve Bayes classifier, eXtreme Gradient Boosting та Logistic Regression, нейронних мереж Convolutional Neural Networks, Long short-term memory та BERT, технік роботи з незбалансованими даними Random Oversampling, Random Undersampling, SMOTE та SMOTETomek, а також ансамблів моделей stacking. З алгоритмів машинного навчання найкраще впорався LR, який продемонстрував значення макро F1-міри рівне 0.7966, коли ознаки були перетворені векторизацією TF-IDF, а коли BoW – 0.7933. З нейронних мереж найкраще значення макро F1-міри рівне 0.76 отримано за допомогою CNN та LSTM. Застосуванням технік балансування даних не вдалося покращити результати алгоритмів машинного навчання. Далі були визначені ансамблі моделей, які складались з алгоритмів машинного навчання. Двома з побудованих ансамблів було досягнуто те ж значення макро F1-міри 0.7966, що і за допомогою LR. Ансамблі, яким вдалося це зробити, складались з векторизації TF-IDF, метамоделі B-NBC та базових моделей SVC, NuSVC LR і SVC, LR відповідно. Таким чином три класифікатори, алгоритм машинного навчання LR та два ансамблі моделей, які були визначені шляхом здійснення комбінації наявних способів розв’язання задачі класифікації політичної забарвленості текстового ресурсу, продемонстрували найбільше значення макро F1-міри 0.7966. Отримані моделі можуть бути використані для детального огляду різних новинних видань за характеристикою політичної забарвленості, інформація про що може допомогти ідентифікувати перебування в інформаційній бульбашці.

Ключові слова: Класифікація тексту; політична забарвленість; алгоритми машинного навчання; нейронні мережі; ансамблі моделей; обробка природної мови

ABOUT THE AUTHORS

Yurii A. Kosiv - Student of Artificial Intelligence Department. Lviv Polytechnic National University, 12, Bandery Str. Lviv, 79013, Ukraine
ORCID: https://orcid.org/0000-0001-7412-2025; yurii.kosiv.mknssh.2021@lpnu.ua
***Research field*:** Natural language processing; artificial intelligence; sentiment analysis

Косів Юрій Андрійович - студент кафедри Систем штучного інтелекту. Національний університет «Львівська політехніка», вул. С. Бандери, 12. Львів, 79013,Україна

Vitaliy S. Yakovyna - Dr. Sci. (Eng), Professor, Professor of Artificial Intelligence Department. Lviv Polytechnic National University, 12, Bandery Str. Lviv, 79013, Ukraine
ORCID: https://orcid.org/0000-0002-9346-145X; asg@opu.ua. Scopus Author ID: 8393582500
***Research field*:** Software quality and reliability; machine learning

Яковина Віталій Степанович - доктор технічних наук, професор, професор кафедри Систем штучного інтелекту. Національний університет «Львівська політехніка», вул. С. Бандери, 12. Львів, 79013,Україна