Machine Learning Model for Paraphrases Detection Based on Text Content Pair Binary Classification

Nataliia Kholodna [1], Victoria Vysotska [1,2], Oksana Markiv [1] and Sofia Chyrun [1]

[1] *Lviv Polytechnic National University, S. Bandera Street, 12, Lviv, 79013, Ukraine*
[2] *Osnabrück University, Friedrich-Janssen-Str. 1, Osnabrück, 49076, Germany*

Abstract

This article dwells process of ML-model development for detecting paraphrasing by binary classification of texts pair. For this study, the following semantic similarity metrics or indicators have been selected as features: Jacquard coefficient for shared N-grams, cosine distance between vector representations of sentences, Word Mover Distance, distances according to WordNet dictionaries, prediction of two ML-models: Siamese neural network based on recurrent and Transformer type - RoBERTa. Developed software uses principle of model stacking and feature engineering. Additional features indicate semantic affiliation of sentences or normalized number of common N-grams. Created model shows excellent classification results on PAWS test data: weighted accuracy (precision) – 93%, weighted completeness (recall) – 92%, F-measure (F1-score) – 92%, accuracy (accuracy) – 92%. Results of study have shown that Transformer-type NNs can be successfully applied to detect paraphrasing in a pair of texts with fairly high accuracy without need for additional feature generation. Fine-tuned NN RoBERTa (with additional fully connected layers) is less sensitive to pairs of sentences that are not paraphrases of each other. This model specificity may contribute to incorrect accusations of plagiarism or incorrect association of user-generated content. Additional features increase both overall classification accuracy and model sensitivity to pairs of sentences that are not paraphrases of each other.

Keywords 1

Machine learning, deep learning, model, WordNet, text, content, rewrite identification, paraphrasing detection, natural language processing, semantic similarity, vector embedding of words, text analysis, text classification

1. Introduction

Process of paraphrasing consists in text rewriting to change words and sequence while preserving original meaning. Rewrite detection plays important role in various NLP tasks, including plagiarism detection, authorship detection, use in QA systems, text summarization, machine translation, general text analysis, etc. More general task of semantic similarity measuring of texts is important in NLP field. Paraphrased plagiarism is one of the most difficult problems plagiarism detection systems face. Most plagiarism detection systems are designed to detect common words and minor changes, but are unable to detect major semantic and structural changes. Therefore, many cases of rewriting remain unnoticed. Paraphrase generation means transforming natural language sentence into new sentence that has the same semantic meaning but different syntactic or lexical form. Detection and generation of paraphrases are used in the following areas:

- Detection of copyright infringement - check for plagiarism, identification of text authors;
- Combination of duplicate content generated by users on information resources;

MoMLeT+DS 2022: 4th International Workshop on Modern Machine Learning Technologies and Data Science, November, 25-26, 2022, Leiden-Lviv, The Netherlands-Ukraine
EMAIL: nataliia.kholodna.sa.2018@lpnu.ua (N. Kholodna); victoria.a.vysotska@lpnu.ua (V. Vysotska); oksana.o.markiv@lpnu.ua (O. Markiv)
ORCID: 000-0002-6908-2900 (N. Kholodna); 0000-0001-6417-3689 (V. Vysotska); 0000-0002- 1691-1357 (O. Markiv); 0000-0002-2829-0164 (S. Chyrun)

- Combination of duplicate records of the same topic or question, which allows to obtain greater completeness (recall) for relevant content when searching;
- Machine translation (simplification of sentences);
- QA – obtaining additional information by generating request options for receiving answers from the database and paraphrasing the answers;
- Text summarization (a subtype of paraphrasing);
- Natural language generation (sentence rewriting);
- Change of writing style.

Detection of paraphrases is closely related to NLP task of semantic similarity evaluation of text. While machine learning model (ML-model) returns probability of paraphrasing or result of binary classification of sentences pairs, ML-model for evaluating semantic similarity returns degree of similarity according to certain metric, e.g. rating from 1 to 5 (SentEval task). When checking for paraphrasing or determining semantic similarity to main problems are the following:

- Lack of common words, for example: Is there Quora user who have seen UFO? Have you seen alien?;
- All words are common, for example: How did Portural team performed in match against Germany? How did Germany team perform in the match against Portugal?;
- Manual marking of documents pairs – result of annotation is subjective depending on situation;
- Spelling and syntactic errors, use of slang – incorrectly written words are identified by system as new or completely different from the correct meaning;
- Homonyms - words have the same spelling, but semantic meaning depends on the context.

Above-mentioned problems prevent rapid development of paraphrasing detection as one of tasks in NLP field. Main methods of detecting paraphrasing include the following:

- Vector space model, basis of which is vectorization or text embedding and calculation of distance/similarity between two texts (Jaccard coefficient, Euclidean distance, cosine distance, Word Mover distance, etc.);
- Artificial neural models of deep learning (convolutional, recurrent, Siamese, encoder-decoder, transformers with an attention algorithm, etc.);
- Calculated distances and results of NN-classification as separate features of final classifier training.

One of the first approaches to measure semantic similarity between textual content is vector space model (VSM) for task of information retrieval domain [1]. Purpose of VSM is to represent each entity of collection (letters in words, words in sentences, sentences in content, content in corpus) as point in n-dimensional space, that is, as vector in VSM [2]. The closer the points are located in this space, the more they are semantically similar, and vice versa. For a given set of k texts $D=\{D_1, D_2,\ldots, D_k\}$, text Di is presented in the form of vector $D_i=(w_{i1}, w_{i2}, \ldots, w_{in})$. In classical word-based VSM, each dimension corresponds to one term/word from set of texts. Weight is determined on the basis of various weighing schemes; Bag-of-Words and TF-IDF are commonly used in word-based VSMs. Similarity between two texts D_i and D_j is calculated by Jacquard coefficient, Euclidean distance, cosine distance, etc. Main disadvantages of this model are high dimensionality, sparsity and problems with the dictionary. Therefore, there are various modifications and generalizations of VSM. Like the previous methods in deep learning, documents or texts are presented as vectors using the Doc2Vec method. In addition, words are also represented as vectors based on the Word2Vec method [3]. There are options for learning the vector representation of words based on the matrix decomposition method, for example, LSA. Another algorithm uses methods based on context, for example, skip-grams, Continuous Bag of Words. These vectors are compared using the cosine distance or some other measure of similarity.

Emergence of Word2Vec model forced researchers to create other vector models such as Doc2Vec, FastText, GloVe, USE, and ELMO. All of these models are *2Vec models because they transform text (in the form of words, phrases, sentences, sections, and entire documents) into vector form, creating n-dimensional vector spaces. Training a neural network (NN-training) with texts from large unlabelled corpora leads to the creation of VSM using arbitrary parameters, the most important of which are dimension of vector space, the minimum word frequency, learning rate, and size of observation window/context of each word. Word2Vec consists of two sub-models: CBOW (continuous bag of

words), which predicts skipped word if we give model the context of skipped word, while skip-gram predicts the context of the given word.

Task of paraphrasing can be divided into two subtasks: detection and generation of paraphrasing.

In the task of detecting paraphrases, result is probability from 0 to 1, where value close to 1 means pair of sentences as paraphrases of each other, 0 - values that differ in semantic load.

Use of deep NNs for NLP has grown significantly in recent years. Siamese, convolutional, recurrent, complex model architectures based on combination of convolutional and recurrent NNs, transformers, etc. are used in research to identify paraphrases. Accuracy of model depends on such factors as number of classes into which the data are distributed, choice of traditional or deep learning methods and their combination with word vectorization (nesting) and TPP methods. When building IS for determining paraphrasing in text for new data set, it is necessary to evaluate not only various methods and architectures of ML models, but also TPP algorithms and text representation in numerical format and possible combinations to obtain the best possible quality (accuracy) of functioning of typical IS rewrite identification.

The study object is process of rewriting detection in text using optimal pipeline, which includes TPP, vectorization or words nesting, selection and features generation, binary classification using certain ML-algorithm and further processing of the obtained results to detect paraphrasing on informational sources Scientific novelty consists in NN application with new architecture, review of state-of-the-art methods and comparison of pipeline different stages will make it possible to determine such a combination, which will allow obtaining qualitative model for determining paraphrasing in text for selected control data sets. Designed IS will improve process of identifying rewrites in text. IS can be used by moderators of information resources and social networks to evaluate support of publication quality and combine data by topics or questions. In addition, paraphrasing detection can be module to existing systems for checking texts for plagiarism.

The research purpose is to develop IT for paraphrasing detection, which will simplify and at the same time improve texts checking for plagiarism or combining data on information resources on the same topics or questions. Designed system should detect duplicates for paraphrasing by searching for similar texts. To achieve the goal, the following tasks have been chosen:

- Develop and describe functional requirements of designed system according to methodology of Rational Unified Process and Unified Model Language;
- Analyze the state-of-the-art methods used for detection of paraphrasing;
- Develop and describe NNs with different architectures (convolutional, recurrent, Siamese NNs, etc.);
- Choose the most optimal model in context of TPP (text pre-processing), vector embedding or vectorization, feature selection and generation, ML algorithm and relevant parameters;
- Implementation of relevant IS identification of rewrite and corresponding approbation of obtained results.

2. Related work

Most information systems that check for plagiarism compare parts of sentences or common words, but paraphrasing detection is still relevant task in the field of natural language processing and is not implemented in most online platforms and applications. In addition, existing plagiarism checking systems cannot detect paraphrasing and indicate the original source with sufficient accuracy [4].

So, detection of paraphrasing is urgent problem, and currently in most academic works, researchers use deep learning methods and architecture of speech models that have millions of parameters. However, the history of solving this problem originates from simpler methods of calculating the semantic difference as the distance between two terms in the WordNet database [5]. This is database of semantic connections between words (synsets). Conjunctions contain synonyms, hyponyms (subtype), meronyms (part), etc. Hierarchical structure of database makes it possible to calculate the semantic relation of individual words using various formulas. In the work [6], approach based on semantic relatedness values of words from the WordNet dictionary is proposed. Method is based on the idea of comparing a pair of words according to a part of speech.

Researchers have used six different semantic similarity distances based on WordNet [7-12]. Maximum similarity is looked for only within word classes with the same part of speech. To implement bidirectionality, arithmetic mean of two values of proposed function is used, the value of which depends on the value of maximum similarity of a pair of words and their specificity. Similarly, all the six distances can be combined by averaging indicators of final evaluation of semantic similarity. According to work [7], semantic similarity of words depends on the length of the shortest path between synsets as graph nodes. According to work [8], word similarity is defined as the intersection of sets of definition words corresponding to given terms. Distance function in the work [9] depends on the depth of two concepts in taxonomy and the depth of the nearest common ancestor, according to work[10] - on the probability of meeting a common ancestor in a large corpus, according towork [11-12] - on the probability of meeting both a common ancestor and two synsets being compared.

Authors in the work [13] have proposed new algorithm for measuring semantic relatedness of short sentences using methods based on the corpus (pointwise mutual information and latent semantic analysis) or on the above-mentioned distances according to data from WordNet. Formula of semantic relatedness of sentences combines metrics of similarity of a pair of words and their specificity. This formula is potentially a good indicator of input texts smilarity, as the obtained values of precision, accuracy, completeness and F-measure are better compared to the basic approach of measuring the cosine distance between two input sentences converted to vector format based on TF-IDF.

In the work [14], method for determining semantic relatedness is proposed, which generates a semantic profile for words based on the main conceptual features collected from encyclopedic knowledge. Main idea of model is that the meaning of word is determined by concepts that are in the direct context. This method consists of two main steps. First, based on Wikipedia, annotated corpus of main conceptual words with relevant references is created. Next, the corpus is used to measure semantic similarity of words and texts.

In the work [15], formula of cosine distance between vectors was adapted to calculate semantic similarity of texts. Proposed algorithm uses word similarity information obtained from WordNet dictionaries. To calculate the similarity between pairs of sentences, cosine distance between the vectors representing sentences and semantic similarity matrices containing information about similarity of word pairs obtained from six hierarchy-based semantic distances of WordNet are used. Each sentence is represented as a binary vector (with elements equal to 1 if the word is presented in the sentence and 0 otherwise).

In the work [16], comparative study has been conducted between vector embeddings of words obtained using NN and traditional vector representations of words based on counting co-occurrences. Authors have performed two small-scale tasks (parsing word meanings and sentence similarity) and two large-scale ones (identifying paraphrasing and tagging dialogue acts). For the task of recognizing paraphrases, researchers have used cosine distance between vectors. Use of vector embeddings of words obtained using NN significantly improves the quality of the model for detecting paraphrasing and tagging dialogue acts, but does not give a significant increase in accuracy in solving small-scale tasks.

In the work [17], authors have proposed method for measuring semantic similarity of texts using corpus-based measurement of semantic similarity of words (pointwise mutual information) and normalized and modified version of the longest common sequence (LCS) string matching algorithm. Research focuses on measuring the similarity between two sentences or between two short paragraphs. Method is evaluated on the basis of Microsoft Paraphrase Corpus (MSRP) data.

In the work [18], process of text checking for plagiarism has been improved using various NLP algorithms: number of identical 3-grams, Language Model Probability, Longest Common Subsequence, Dependency Relations Matching. Obtained features have been used to train a naive Bayesian classifier in two categories depending on presence of plagiarism and in four categories depending on level and type of paraphrasing. Experimental results are satisfactory for classifying documents in the corpus into two categories (rewritten and unique): 37 out of 38 non-plagiarized documents are correctly classified, and only a few plagiarized documents are classified as non-plagiarized (5 out of 57). Distinguishing between the three different levels of plagiarism proved to be a much more difficult task. The unsatisfactory accuracy of the plagiarism classification model depends both on the complexity of the task and on the possible error of the annotators.

In works [19-20] two systems have been developed for determining semantic similarity of short sentences pairs. All word similarity measures are based on WordNet. To calculate the similarity score

for a pair of words, authors took the maximum similarity score for all possible pairs of concepts - WordNet synsets and used the NLTK library to calculate the Leacock and Chodorow and Lin similarity measures. For corpus-based word similarity, authors used a distributive lexical semantic model. They used latent semantic analysis (LSA) on a large corpus to estimate distributions. The system is trained to predict sentence similarity scores using a support vector regression model measuring several characteristics of the degree of overlap of common words, syntactic similarity, and semantic proximity of words.

In the work [21], method for identifying the presence of plagiarism using paraphrasing is presented. This method uses the classic ML algorithm - logistic regression. In this system, certain features of vocabulary, syntax, semantics, and structure, which are obtained from the "suspicious" document and the original, act as signs. Features of the lexicon include: Dice Coefficient (the number of common symbols), Jaro Distance, Jaccard Coefficient (the ratio of common terms to the total number of terms), Levenshtein Distance, adapted Manhattan Distance, N-gram Distance - an analogue of Manhattan Distance for N-grams, Soundex Distance. Syntactic features include POS N-gram Distance and Noun Ratio, semantic features – Semantic Similarity Distance, structural features – Stopword N-gram Distance, Word Pair Order, String Length Ratio. Experimental results on the PAN@CLEF2013 dataset show that using different types of features leads to more accurate results.

The article [22] has proposed new Bi-CNN-MI deep learning architecture for rewrite detection/identification by comparing sentences at multiple levels of detail (unigrams, N-grams, and sentences) based on a convolutional NN and modelling interaction features at each level. Resulting features are input features for a binary classifier based on logistic regression.

In the work [23], system is presented for solving problem of paraphrasing detection by revealing differences between sentences and evaluating how different the sentences are. This method also makes it possible to detect paraphrasing that contains a small amount of additional information. Method of support vectors was used as the classification algorithm.

In the work [24], classical ML algorithms (support vector method, naive Bayes classifier, maximum entropy classifier) were compared. Signs are data on the lexical and semantic relatedness of sentences. The "word overlap" feature group contains the ratio of the number of identical N-grams to the total number of words in the sentences. The lexical group of features also includes skip-grams and longest common subsequence. Semantic features include noun/verb semantic similarity measure – the number of common nouns or verbs, proper name – the number of common proper names. Semantic feature cardinal number allows to capture numbers with comparison "greater than" and "less than", as well as numbers written in words. According to results of research, the best indicators are achieved using method of support vectors.

In the work [25] it was shown that the quality metrics of machine translation (BLEU, NIST, WER, PER) can be used as features in the system of definition and detection of paraphrases for training the classifier and obtaining quality results. In addition, the researchers developed a classifier based on Position independent word error rate (PER) and information on the distribution of parts of speech in sentences.

In the work [26], authors have re-examined the hypothesis that metrics developed for automated evaluation of quality of machine translation can be used as features of classifier for the detection of paraphrasing. The results showed that the meta-classifier, whose features are only machine translation quality metrics, significantly outperforms the accuracy indicators obtained in previous studies. A total of three classic machine learning algorithms were used for classification: logistic regression, SMO implementation of the support vector method, and a variation of the nearest neighbour algorithm. Machine translation quality metrics include: BLEU, NIST, TER, TERp, METEOR, SEPIA. Using the TERp metric alone provides good results and outperforms many other classification methods based on multiple complex features.

In the work [27] authors have developed a system for detecting paraphrasing in short texts based on deep learning methods. Architecture of classifier is based on layers of convolution and recurrent (long-short term memory) NNs that process the obtained result. The result of processing by recurrent NNs is a semantic representation of a sentence, so the features of the final classifier are the difference between two output vectors from recurrent NNs. Convolutional NN transforms the pairwise similarity matrix of all words into a vector of semantic similarity of sentences, which is used as features for the classifier. Additional features are the cosine distance between the vectors of two sentences, the average value of

the distance between nouns, verbs, adjectives, calculated on the basis of WordNet, the ratio of the number of common uni-, 2-, 3-grams to the total number of N-grams, etc.

In the work [28], approach to identification of paraphrasing based on recursive autoencoders that examine feature vectors for phrases using syntactic trees is presented. These features are used to measure the similarity of words and phrases between two sentences. Since the sentences are of arbitrary length, the resulting similarity matrix is of variable size. Added a new dynamic pooling layer that computes fixed-size representations from variable-size arrays. This representation was used as input to the classifier.

In the study [29], in contrast to the creation and selection of large number of features, recurrent NN with long-short-term memory layers was used. NN receives sentences of variable length as input and learns the regression task of predicting the degree of semantic relatedness of a pair of sentences. The NN has a Siamese architecture and is trained using the Mean Squared Error loss function. To represent words in the form of vectors, pre-trained word2vec vector embeddings obtained on large unlabelled data corpora were used.

Authors in the work [30] have combined Siamese recurrent NN with bidirectional recurrent networks with long-short-term memory layers at the symbol level. Such a model is insensitive to spelling errors, substitution of synonyms and redundant words.

In the work [31], several different types of Siamese NNs were tested for the problem of paraphrasing detection: LSTM, Bi-directional LSTM, GRU, Bi-directional GRU, LSTM + Attention, GRU + Attention, GRU + Capsule + Flatten. The best results are obtained for NN with cells of the gated recurrent unit type.

In the work [32], Siamese NN with long-short-term memory was used to classify a pair of Arabic texts according to whether they are paraphrases of each other. For the vector representation of words, the researchers used the method of vector embedding of the word2vec family, which is called Glove. To calculate the semantic similarity of texts, the cosine distance between two vector representations of sentences was used.

In the work [33], Siamese NN was used to detect paraphrases in Telugu texts. Most of the data was collected manually from various newspapers. In addition, the researchers compared three methods of vector embedding (Word2Vec, Glove, Fasttext) and their combination. The best accuracy on the test data set is achieved for the combination of these three methods.

In the work [34], deep learning approach with reinforcement for the generation of paraphrases is presented. New structure for solving this problem is described, which consists of a generator and an estimator. Generator, built as an ML model based on recurrent NNs of the sequence-to-sequence architecture, paraphrases the input sentence. Evaluator classifies whether two sentences are paraphrases of each other. Generator is first trained using deep learning and then further tuned using reinforcement learning in which the evaluator provides a reward. To train the ratter, researchers propose two methods based on teacher-directed learning and inverse reinforcement learning, respectively, depending on the type of training data.

In the work [35], SimAll system was developed that combines methods based on tape comparison, corpus, and knowledge, and supports English and Arabic languages. In total, the system supports 61 algorithms for evaluating the semantic similarity of sentences, the evaluation results of each algorithm after normalization belong to the interval [0, 1]. User must choose the method of aggregation of the results: average, sum or maximum value. Knowledge-based metrics include 6 semantic similarity distances: Path (path), Leacock & Chodorow (lch), Wu & Palmer, Resnik (res), Lin (lin), Jiang & Conrath (jcn). Metrics are available for English only.

In the work [36], two variants of the LSTM tree model are proposed for processing dependency trees or syntactic trees. The created models were tested on the task of determining the degree of semantic similarity of two sentences.

In the work [37] authors have developed system combining convolutional and recurrent NNs for measuring the semantic similarity of sentences. Researchers have used convolutional network to account for the local context of words and an LSTM to account for the global context of sentences. This combination of networks helps preserve relevant information about the sentences and improves the computation of sentence similarity.

In the work [38], large-scale corpus of parallel data was presented using deep learning models BERT, RoBERTa, Longformer of the Transformer architecture. Data set includes paragraphs from

scientific works in arXiv, theses, as well as Wikipedia articles and their paraphrased counterparts (1.5 million paragraphs in total). Deep learning architecture of Transformer makes it possible to accurately classify original and paraphrased texts using static vector embeddings (fastText) [39]. RoBERTa model achieved the best results in the task of detecting paraphrases.

In the work [40], new approach to creation of parallel corpora of paraphrased texts using generative NNs of the Transformer architecture was proposed and applied. According to research results, while NN-generated datasets can improve the accuracy of paraphrasing detection systems, a smaller but human-generated dataset improves the classification accuracy even more.

In the work [41], additional fine-tuning of the BERT deep learning model of the Transformer architecture was applied. BERT model has 110 million parameters in the basic version, 340 million in the "big" [42]. Pre-setting BERT consists of two tasks. The first task is language modelling, for which the model must predict a randomly selected and masked token. Another task is predicting the next sentence, where the model has to determine whether two sentences are consecutive. Additional adjustment of the model for the main task occurs with the help of a suitable data set.

In the work [43], metric of distance between documents, Word Mover's Distance, was presented based on vector embeddings. Word Mover's Distance measures the difference between as the minimum distance that the word vector embeddings of the first sentence "must travel" to the word vector embeddings of the second sentence.

3. Methods and materials

Information system concerning paraphrasing detection is intended for moderators of information resources and social networks to evaluate and maintain the quality of publications and group data by topic or question [44-63]. In addition, the paraphrasing detection algorithm can be added to existing plagiarism text checking systems. It is important to give example of requirements description in the context of IS verification of the academic works uniqueness. Functioning of information system is ensured by the scientific director, text author, person responsible for checking the text and software. Supervisor (or teacher) checks all stages of author writing of his work and conducts TPP before automated check for originality. Author actively interacts with the manager and responds to his criticism, corrects the work according to his comments and then submits his work for review. Responsible person uploads the text to the program and receives the result, informs the author and the scientific supervisor, forms a report. Precedent stakeholders and their requirements:

- Software for checking texts must meet the specified system requirements, must ensure maximum accuracy when checking results and generating reports; must authenticate the user, verify operation and return the result, store the result in user account.
- Author wants to receive confirmation of originality of work for its publication or protection; must write the text, prepare the work according to the requirements and submit it for further processing.
- Person responsible for the check must receive the result of the automated check, make sure that the program correctly identified the copied fragments and the original sources of the paraphrased sentences.

IS user: person responsible for verification, who will use the IS to check the texts for uniqueness. Prerequisites of the precedent:

- Scientific work must be properly designed and submitted in one of the formats supported by the system;
- Computer that will be used for the check, connected to the Internet and with the necessary software installed;
- Person responsible for the check must be successfully authorized.

Basic successful scenario:

- Person responsible for the check loads the text into the program and receives the result.
- Result of examination is subject to additional analysis regarding the number of identified matches and the adequacy of references to primary sources.

Alternative streams:

- Error in the operation of the program:

1. Person responsible for the check contacts the technical support service (developer) and reports an error in the program.
2. Developer fixes the error and provides a new version or provides information on how to fix it yourself. (return point).

Post-conditions: data on the result of the check are entered into the database.

Special requirements: IS must accurately set the percentage of text uniqueness, highlight parts of the text according to their types (copied text, original text, paraphrased text, quotation or incorrect indication of the original source) and provide a list of sites (works) with a percentage of coincidence, fully support Ukrainian and English languages, easy to install, publicly available, support various formats (.txt, .doc, .docx, etc.).

In the usage diagram (Fig. 1), main actors are users and applications that implement system functions in different ways. This diagram does not depict detailed use cases such as user registration, file upload, or administrator interaction with the system, which in turn includes system configuration, NN training, and choice of paraphrasing detection methods (Figures 2-3). Individual user of platform can use the system as an application to check for plagiarism and uniqueness of content. Such a system can be used to check the academic integrity of students and researchers. In addition, the plagiarism checker can be used to identify the author of the text. Thanks to the additional check for the presence of paraphrasing, such a system will have greater completeness (recall). In this case, the share of correctly indicated sources from all true sources will increase.

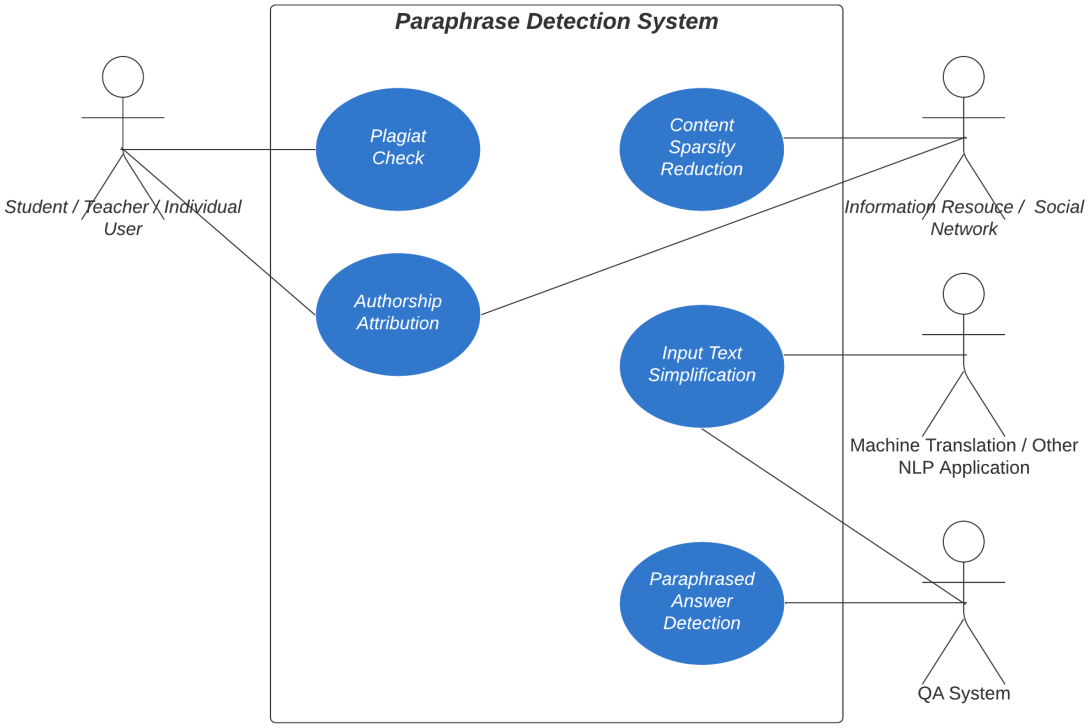


Figure 1: Use case diagram of rewrite detection IS

Information resource (including social networks) can use the system to reduce sparsity of content by combining publications or questions related to one common topic. Also, these resources can use the paraphrasing detection system to establish the author of the text. Both machine translation applications and QA systems can use the functionality of the proposed IS to simplify the input message. In addition, QA systems can also apply paraphrasing detection for more advanced search for answers to questions.

To construct class diagram, seven main classes are defined: Individual User (system user), Document (a document whose uniqueness must be checked), Paraphrase Detection Software (an application for plagiarism checking), User Database (user database), Documents Database (document database), Data Processing Server, ML Model. Class diagram in Fig. 5 indicates the use of the system only by an individual user to check the uniqueness of his own texts. Application class diagram of the paraphrasing detection system used by third-party applications must define additional methods for

communicating with the host application using queries. The user can download and modify the document, check it and get the results. Plagiarism checker application class denotes a program that implements a user interface and basic plagiarism checking methods by comparing tapes. Such a program can be installed locally on a computer or implemented as an online platform. Application sends requests to the computing server and user database.

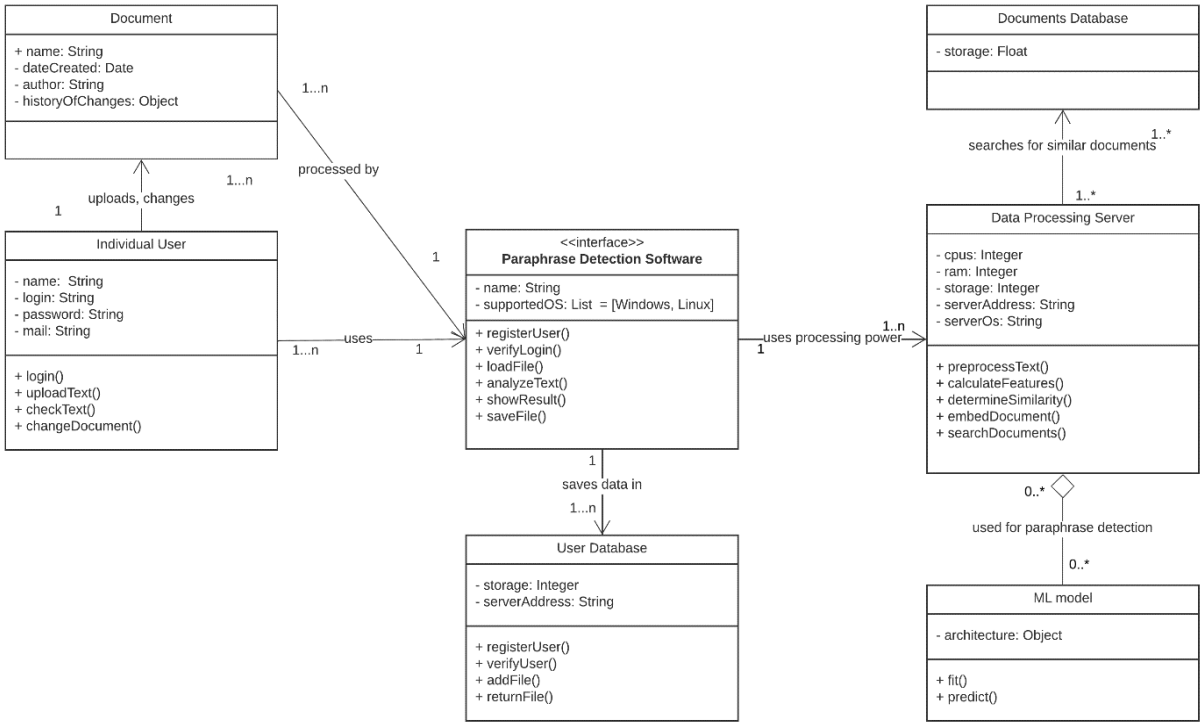


Figure 2: Diagram of IS classes for detection of paraphrasing in the text

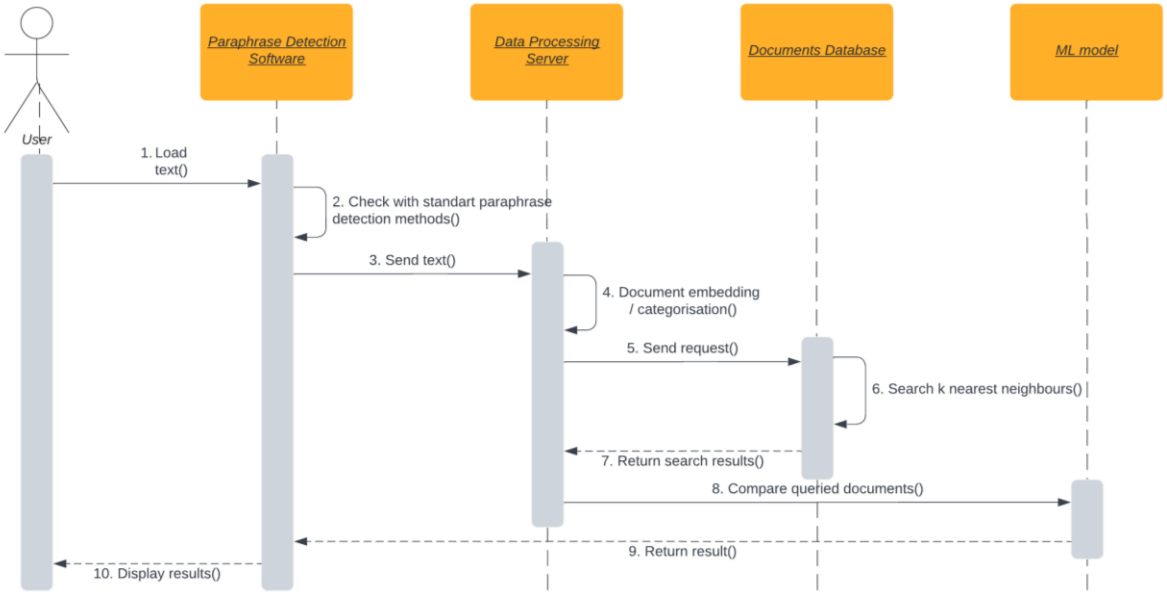


Figure 3: Sequence diagram of paraphrasing detection system

Data processing server includes methods for document comparison using ML algorithms and applies vector nesting or document classification to find nearest neighbors, since pairwise comparison of text with all documents available in the database is too resource-intensive. Data processing server must also calculate additional features that will be used as input to the classifier. Such additional features can be the total number of named entities or parts of speech (Fig. 3). Calculation of such features will require

additional implementation of special parsers. Sequence diagram shows the temporal aspects of the interaction of system components. Bypassing system settings, registration and user login, after uploading the file, the next step is to check the texts for uniqueness using basic algorithms based on the comparison of tapes and their fragments. For a more detailed check for the presence of paraphrasing, the document must first be classified or converted to a vector representation. This approach is used in order to save computing resources and reduce the time of processing a request.

Important internal system settings include number of documents that will be compared with the user. Too many documents will slow down the check process, very few will reduce its accuracy and completeness. Results of determining degree of semantic similarity or binary classification are returned and should be visualized together with results of previous simple check.

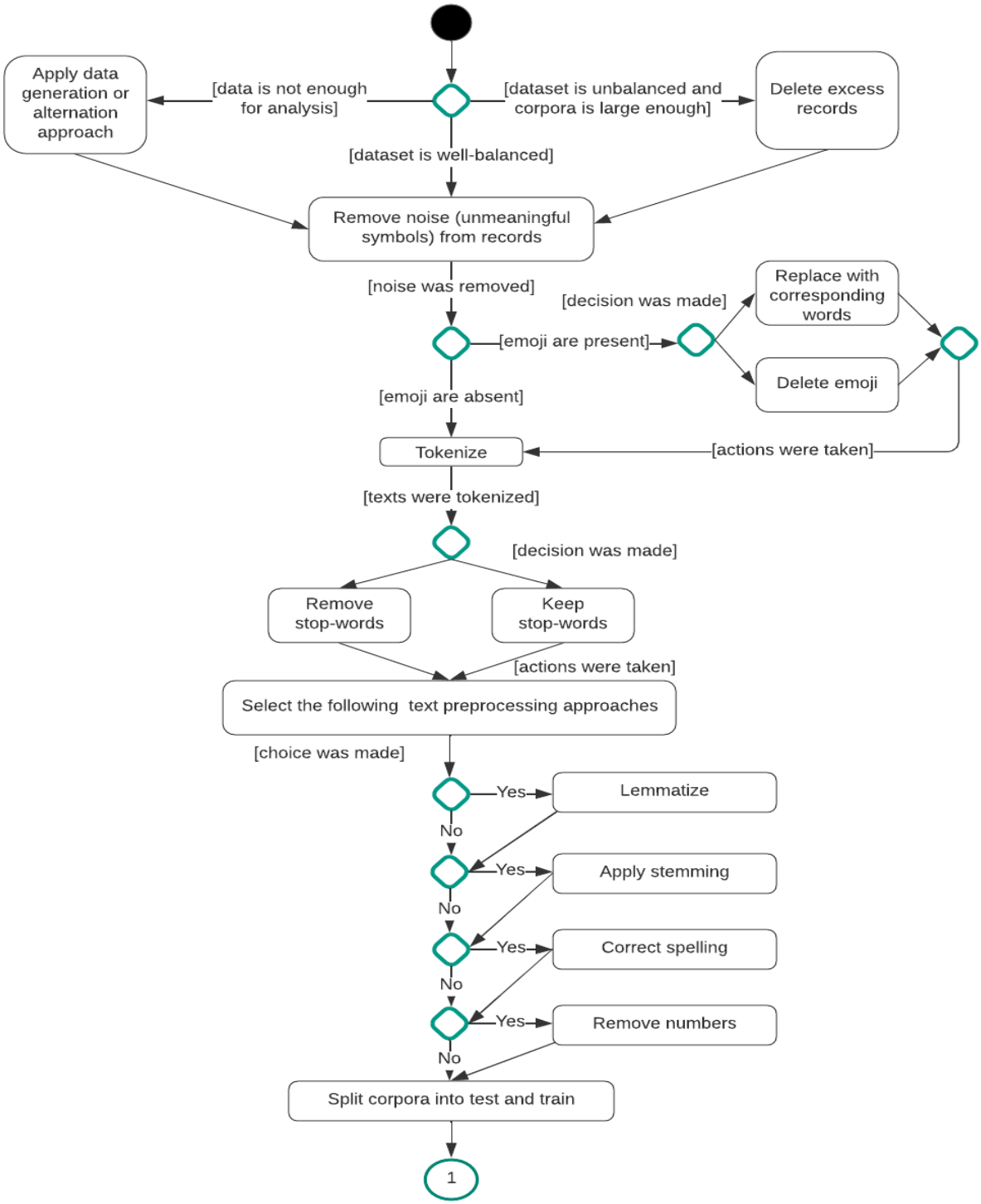


Figure 4: The first part of activity diagram creating paraphrasing detection system

Important part of developing system for automatic detection of paraphrasing is the creation and training of ML model for task of text classification. To obtain maximum quality of verification, it is necessary to investigate various methods of calculating distances, ML-algorithms, TPP-methods.

Activity diagram is divided into two parts (Fig. 4-5) and indicates all possible branches during TPP and system construction, respectively. So, the system developers face a branching system of all possible methods and their combinations. In the case of binary classification, it is first necessary to make sure that the volume of the corpus is sufficiently large and balanced, since the numerical advantage of records of a certain class can lead to a biased solution of artificial NN.

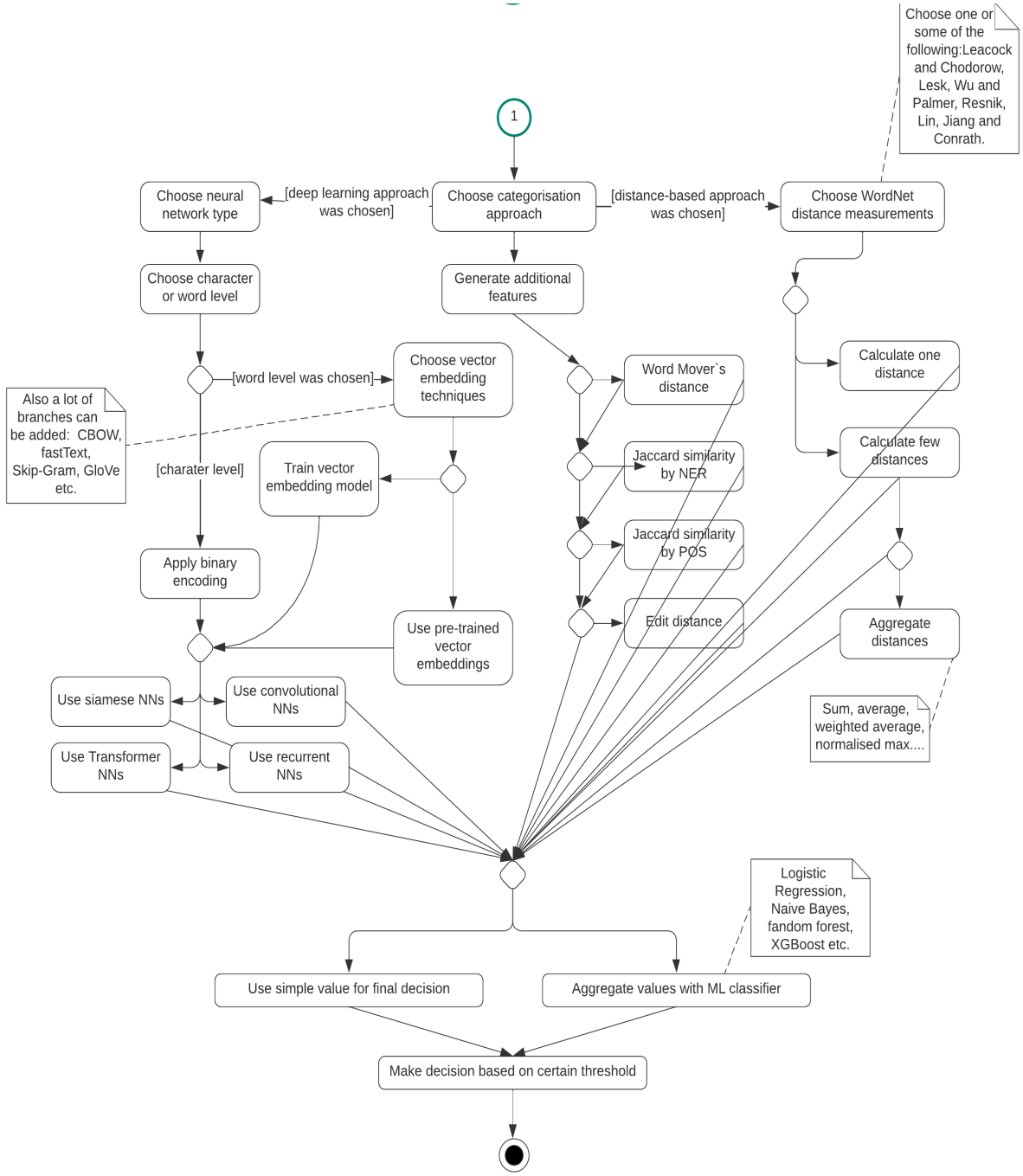


Figure 5: The second part of activity diagram creating paraphrasing detection system

If there is not enough data, methods of generating artificial data or changing existing records are used. After that, it is needed to remove "noise", which can be extra symbols that do not contain meaningful load. Short texts, especially social media posts, are more likely to contain emoji. Emoji can indicate tonal or emotional colouring of the text, so system developers can replace them with

appropriate words or remove them. The next step is to break the sentences into tokens. Stop words do not contain semantic colouring, so their removal is optional. The next steps of TPP include lemmatization (initial word form), stemming (word stem), correcting spelling errors, and removing digits. As with stop-word removal, applying the above steps is optional, but may have different effects on classification accuracy. After TPP, dataset is divided into training and training samples.

Fig. 5 shows all main methods and their combinations for binary classification of sentences pair depending on whether they are paraphrases of each other. System developers can apply single metric to the final decision or aggregate multiple values using ML methods such as logistic regression or random forest.

To simplify the diagram appearance, it partially lacks guarding conditions for transition between activities. "Notes" indicate additional possible ramifications of choosing the method of calculating the distance according to WordNet, method of aggregating distances, vector nesting of words, classification. Layout diagram of the automatic detection of paraphrasing system (Fig. 6) shows the data processing server processors and two databases. User has access to functionality of system using graphical interface of downloaded program or online platform. Database stores information about users and documents. Data processing server contains modules of ML-model, basic plagiarism check and vector attachment or document classification. Alternative to the deployment scheme is to use personal computer own computing resources to analyse the data, however, given that to obtain reliable result, it is necessary to apply several stages of validation, especially with the use of ML methods and the comparison of documents stored in the database. Such an architecture is not appropriate.

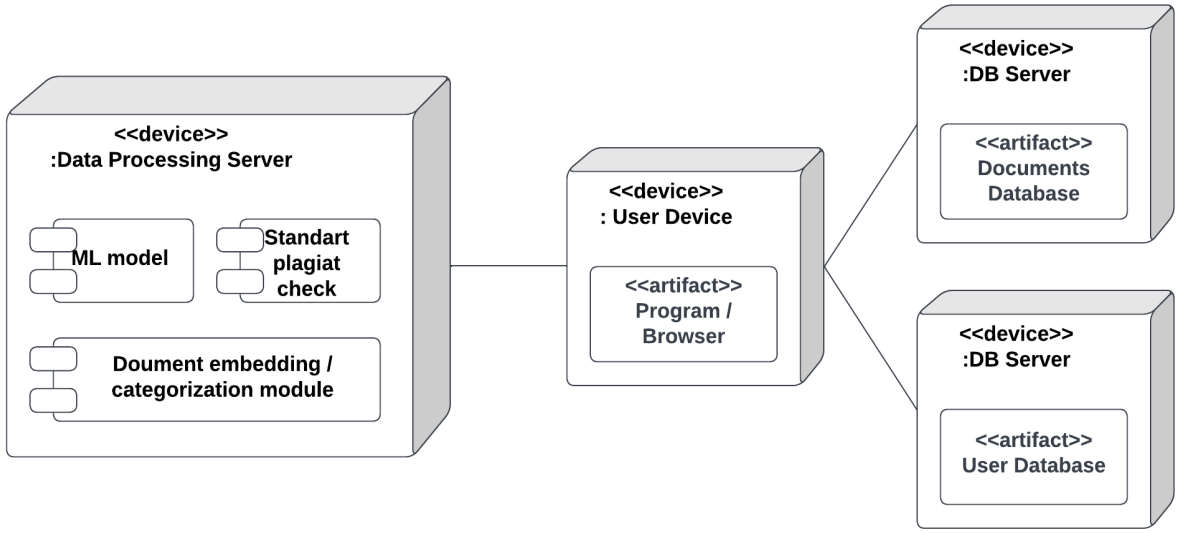


Figure 6: Deployment diagram of automatic paraphrasing detection

For ML, the most popular languages are Python, R, Java, C++. Advantage of Python over other languages for creating a paraphrasing recognition system is its support for a large number of libraries for:

- Working with ML methods: Scikit-Learn;
- Creating artificial NNs, deep learning: TensorFlow, Keras, PyTorch;
- Natural language processing: NLTK, spaCy, WordNet;
- Working with arrays, matrices: NumPy;
- Working with tables: pandas;
- Data visualization (including interactive): Matplotlib, seaborn, Plotly.

Scikit-Learn library supports TPP, data dimensionality reduction, selection of ML models for regression, classification or cluster analysis. However, Scikit-Learn does not have comprehensive support for building deep learning models. To create artificial NNs, Keras library was chosen, which serves as a high-level API for TensorFlow 2. Keras allows to build sequential models in the form of a graph, the vertices of which are layers of a certain type with a given number of nodes. Also, with the help of Keras, it is possible to combine results of the work of several separate parts of the NN for their further processing, such a structure is not linear. TensorFlow makes it possible to import a trained ML model for further use in other applications. TensorFlow also supports low-level tensor operations using

CPUs, GPUs, and tensor processing units. NLTK library in this study is used for TPP: tokenization, stop-word removal, stemming, lemmatization. Also, with the help of functions from this library, it is possible to detect the most popular N-grams and parts of the language of individual tokens, recognize named entities, etc. Additional libraries that simplify working with natural language include Regex and emoji - for using regular expressions and replacing emoji with words, respectively. Jupyter Notebook is a convenient tool for working on data research and ML applications, which allows to run the written code in small fragments - cells. One online service that allows to use Jupyter Notebook without a local installation is Google Colab. This service makes it possible to use GPU and TPU graphics processors, which significantly speed up NN training.

The dataset Paraphrase Adversaries from Word Scrambling [44], part of PAWS-Wiki Labeled (Final) have been selected for training and testing the ML-model, as well as constructing features. PAWS-Wiki contains 65,401 pairs of sentences, 44.2% of which are paraphrases of each other. Since various studies use different methodologies to determine semantic similarity and detect paraphrasing (from calculating the number of shared N-grams to applying deep ML methods) [45], in order to obtain the highest classification accuracy, it is necessary to compare and (or) combine different algorithms for measuring the semantic similarity of sentences. For this study, the following semantic similarity metrics or indicators have been selected as features: Jaccard coefficient for shared N-grams, cosine distance between vector representations of sentences, Word Mover's Distance [43], distances according to WordNet dictionaries [1-2], prediction of two ML- models: Siamese NN based on recurrent and Transformer type – RoBERTa [46].

N-gram is a sequence of n words. In the context of detecting paraphrases in a text, the number of common N-grams, normalized to the total number of N-grams in both sentences, helps to detect semantically similar sentences that are close in semantic load, but are not paraphrases, since the second sentence is obtained by permuting the words in the first sentences, and, accordingly, has a completely different meaning. Jaccard coefficient for two sets is calculated as follows:

$$J(A, B) = |A \cap B| / |A \cup B|. \quad (1)$$

To calculate N-grams in each sentence, sentences are first reduced to lower case, all separators and additional characters are removed. Total Jacquard coefficients for 2-, 3-, and 4-grams were calculated.

There is a wide variety of methods for obtaining vector embeddings of words GloVe, Word2Vec: CBOW / Skip-Gram, fastText. The research used vector embeddings of the deep learning model BERT [42] for NLP of the Transformer architecture. The basic BERT model has 110 million configurable parameters, the advanced version has 345. Feature of the Transformer architecture is the presence of attention mechanism, thanks to which data is processed simultaneously (as opposed to recurrent NN, where data is perceived sequentially). In addition, feature of BERT is the pre-training of NNs to solve two tasks: predicting a certain word in a sentence and determining whether the second sentence is a logical continuation of the first. In [42], this ML model is pre-trained on unlabelled data from BooksCorpus (800 million words) and English Wikipedia (2,500 million words). Pre-learning and the attention mechanism make it possible to obtain contextual vector embeddings of words. Using pre-trained model, vector representation matrix for each sentence has the following dimension: torch.Size ([1, 128, 768]), where 1 is the number of sentences in the batch, 128 is the maximum sentence length, and 768 is the dimension of the vector nesting. For the vector presentation of sentences, data are averaged for each word in the sentence, resulting in a nested vector with a length of 768 values.

Formula for calculating the cosine distance:

$$C = (\sum A_i B_i) / ((\sum A_i^2)^{1/2} (\sum B_i^2)^{1/2}). \quad (2)$$

This feature potentially detects semantically similar sentences, but sentences with the same words that are not paraphrases will have a small cosine distance. Word Mover Distance applies vector nesting to calculate the semantic distance between sentences. WMD distance measures the difference between two text documents as the minimum distance that the word vector embeddings of one document must "travel" to reach the points of the word vector embedding of the other document. Similar to the previous feature, such a distance metric will allow detection of semantically similar pairs of words, but will not help to detect examples of unparaphrased sentences with rearranged words. To measure semantic similarity of sentences, two metrics of the semantic distance of synsets according to WordNet dictionaries are used: Leacock and Chodorow [1], Wu and Palmer [2]. According to Leacock and Chodorow [7], the semantic similarity of words is determined using the following formula:

$$sim_{l\_ch} = -\log\left(length/(2*D)\right), \quad (3)$$

where *length* is the length of the shortest path between two concepts (number of nodes), D is the maximum depth of the corresponding taxonomy.

The semantic distance function of Wu and Palmer [9] depends on the depth of two concepts $d(concept_i)\text{depth}(\text{concept}_i)$ in the taxonomy and the depth of their nearest common ancestor d(LCS):

$$sim_{wu\_p} = 2*d(LCS)/(d(concept_1) + d(concept_2)). \quad (4)$$

Since in WordNet the distance is calculated for each pair of synsets separately, the approach in the work [3] is used to represent the distance between two sentences. To implement bidirectionality, the arithmetic mean of two values of a certain distance is used, the value of which depends on the maximum similarity of a pair of words. However, word specificity is not taken into account in this case (no multiplication by TF-IDF value). Thus, to calculate the distances between pair of sentences, it is needed to compare each word of the first with each word of the second.

Siamese NNs use special loss functions because this type of NN learns such intrinsic vector representations of the data that identical records will have a small cosine or Euclidean distance. In this study, the contrastive loss function was calculated for the Euclidean distance:

$$L = 0.5\,(1-Y)\,E^2 + 0.5\,Y\,\{\max(0, m-E)\}^2, \quad (5)$$

where E is the value of Euclidean distance between the predicted vector representations of records, Y is the true value, m is the threshold of records belonging to the same class: the predicted distance between vector representations of different classes must not be $< m$, $m = 2$.

4. Experiments

Since training sample of PAWS-Wiki data set contains 49,401 records and calculation of each feature requires significant computing resources, a repository with the corresponding processing script and calculation results was first created for each feature. Project has the following structure (Fig. 7).

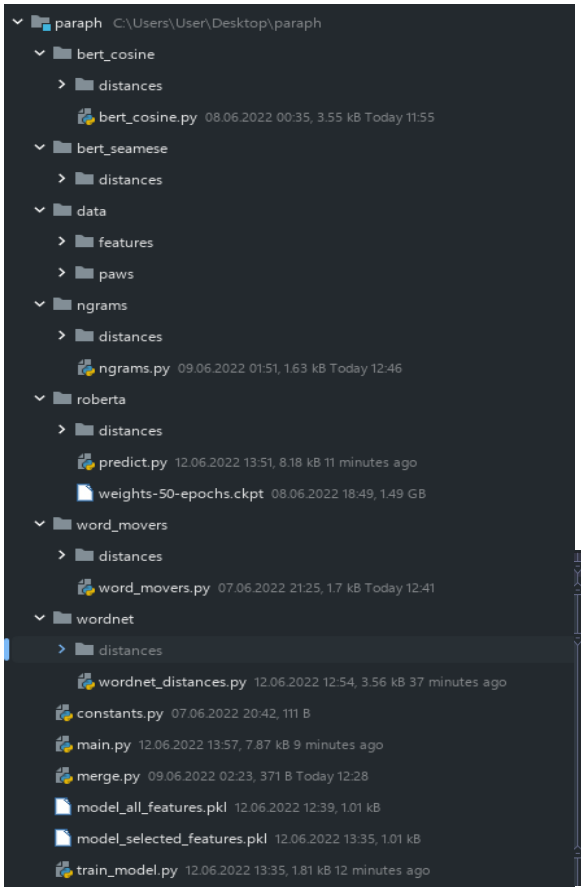


Figure 7: Structure of the project repository

```
import glob
import pandas as pd

for PART in ['dev', 'test', 'train']:

    files = glob.glob(f'C:/Users/User/Desktop/paraph/**/distances/{PART}_*.csv',
                      recursive=True)

    df = pd.concat(map(pd.read_csv, files), axis=1)
    df.drop(columns='id', inplace=True)

    df.to_csv(f'data/features/{PART}.csv', index=False)
    print('ready')
```

Figure 8: Merging of files with attributes

For each feature, there is a distances repository containing the corresponding processing results. Each such a repository contains three files - results of processing the test, training, and validation samples, respectively. For training NN (Siamese with recurrent connections and RoBERTa), computing resources of Google Colab were used, so predictions of Siamese NN were loaded locally, and in the case of RoBERTa, only saved weights in .ckpt format were loaded, processing and classification took place locally. The merge.py file is created to merge all features (Fig. 8), resulting in the data type of the Pandas DataFrame library. The same data is stored in .csv format in the data/feature's repository for further processing. From Fig. 9-13 it can be seen that predictions of NN RoBERTa almost completely coincide with the true labels corresponding to a pair of words. Since the other indicators are not integers and, by their principle, denote completely different signs, the logistic regression method was chosen for their unification. Its training is written in the train_model.py file, and the model itself (as well as other classical ML methods that were compared) is imported from the sklearn library. The file model_all_features.pkl contains the logistic regression weights, the model of which was trained on all generated features. Further processing of texts with the help of trained ML-models requires combining the calculation of all features, which is done in the main.py file. Corresponding functions and pre-trained models are imported from the corresponding files.

	id	sentence1	sentence2	label
1	1	This was a series of nested angular standards…	This was a series of nested polar scales , so…	0
2	2	His father emigrated to Missouri in 1868 but …	His father emigrated to America in 1868 , but…	0
3	3	In January 2011 , the Deputy Secretary Genera…	In January 2011 , FIBA Asia deputy secretary …	1
4	4	Steiner argued that , in the right circumstan…	Steiner held that the spiritual world can be …	0
5	5	Luciano Williames Dias (born July 25 , 1970 …	Luciano Williames Dias (born 25 July 1970) …	0
6	6	During her sophomore , junior and senior summ…	During her second , junior and senior summers…	1
7	7	The smallest number that can be represented i…	The smallest number that can be represented a…	0
8	8	His father emigrated to Missouri in 1868 , bu…	His father emigrated to Missouri in 1868 but …	1
9	9	The Villa Pesquera facilities are owned by th…	The facilities of Villa Pesquera are operated…	0
10	10	It is situated south of Köroğlu Mountains and…	It is situated south of Köroğlu - mountains a…	1

Figure 9: Example of input data

bert_cosine_distance	prediction_seamese_bert	3_grams_jaccard	2_grams_jaccard	4_grams_jaccard
0.99233836	0.2205543518066406	0.4117647058823529	0.5625	0.3142857142857143
0.98792297	0.4695497751235962	0.5517241379310345	0.8076923076923077	0.4827586206896552
0.9849439	0.6481984257698059	0.1724137931034483	0.44	0.0666666666666666
0.9759501	0.6250963807106018	0.2162162162162162	0.3823529411764705	0.1025641025641025
0.9834332	0.9546312689781188	0.375	0.5	0.25
0.9871587	0.7835149765014648	0.5769230769230769	0.72	0.5
0.98474044	0.856931209564209	0.2758620689655172	0.3928571428571428	0.2068965517241379
0.9874414	0.1621454060077667	0.8	0.88	0.72
0.9863758	1.0148042440414429	0.3478260869565217	0.4761904761904761	0.16
0.98016256	0.9825689196586608	0.4	0.5333333333333333	0.3571428571428571

Figure 10: Characteristics corresponding to the input data

s_jaccard	predictions_raw	predictions	shortest_path_distance	wup_similarity	wm_distance
57142857143	0.0003076210268773	0	0.8808753618444751	0.9334199974323896	0.1638981500140085
86206896552	0.0001363550400128	0	0.9666656444455284	0.9749989694455368	0.1046792674400867
66666666666	0.9953380823135376	1	0.7899293829589504	0.8570820710088407	0.1082155853879519
41025641025	0.0146335149183869	0	0.8833327625003693	0.9286204319799678	0.258732279284138
0.25	0.004495037253946	0	0.9999987500015626	0.9999987500015626	0.0
0.5	0.9972121119499208	1	0.8849394652984164	0.9198043154376784	0.1023514166322542
65517241379	0.0016741530271247	0	0.9220770838260276	0.9545445867776484	0.1253511271784937
0.72	0.9963889122009276	1	0.9666656444455284	0.9749989694455368	0.1046792674400867
0.16	0.0004754920082632	0	0.999998819445843	0.999998819445843	0.0
28571428571	0.996845006942749	1	0.999998000004	0.999998000004	0.0

Figure 11: Characteristics corresponding to the input data

```python
from roberta.predict import Pairs_Dataset, config, Classifier_Model, predict
from transformers import AutoTokenizer
model_name = config['model_name']

tokenizer = AutoTokenizer.from_pretrained(model_name)

dataset_test = Pairs_Dataset(path, tokenizer, 'Quality', '#1 String', '#2 String')

BATCH_SIZE = config['batch_size']
test_loader = DataLoader(dataset_test, BATCH_SIZE, shuffle=False)

model_roberta = Classifier_Model.load_from_checkpoint('roberta/weights-50-epochs.ckpt', config=config)
model_roberta.to(device)

preditions_train_raw, predictions_train_int = predict(test_loader, model_roberta)
```

Figure 12: Example of classification using RoBERTa ML model and functions imported from other files

```python
cos = get_cosine(df, 'sentence1', 'sentence2')
gr_2 = n_gr_sim(df, 2, 'sentence1', 'sentence2')
gr_3 = n_gr_sim(df, 3, 'sentence1', 'sentence2')
gr_4 = n_gr_sim(df, 4, 'sentence1', 'sentence2')
wn = wordnet_distance(df, 'sentence1', 'sentence2')
wm = word_movers(df, 'sentence1', 'sentence2')

X_test = pd.DataFrame({
    'bert_cosine_distance': cos,
    '2_grams_jaccard': gr_2,
    '3_grams_jaccard': gr_3,
    '4_grams_jaccard': gr_4,
    'predictions_raw': preditions_train_raw,
    'predictions': predictions_train_int,
    'shortest_path_distance': wn,
    'wm_distance': wm
})
```

Figure 13: Creation of table with features

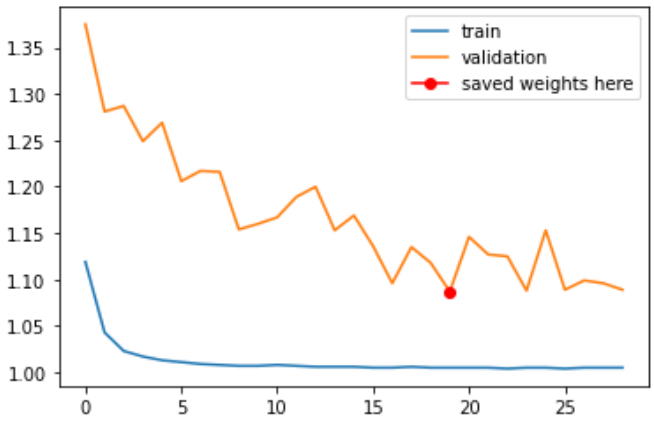


Figure 14: Training of Siamese NN

NN contains two layers of bidirectional long-short-term memory (LSTM), two hidden layers of fully connected neurons (512 and 128 neurons), output layer containing 16 neurons. Vector embeddings of words were obtained by direct propagation in the pre-trained NN BERT. NN was trained for 30 epochs on the training sample with validation after each epoch (Fig. 14). Significant difference between the loss functions on the training and validation samples indicates insufficient complexity of model (perhaps insufficient number of hidden layers or neurons). In addition, accuracy of classification may depend on the selected hyperparameters, setting of which requires additional training and testing of the model. For further classification, the NN weights are saved for the smallest value of the loss function (19th epoch). Pre-trained NN RoBERTa is improved analogue of BERT: its main difference is selection of hyperparameters during pre-training, increase in size of training corpus (16 GB of sentences from Books Corpus and English Wikipedia, CommonCrawl News dataset (63 million articles, 76 GB), Web text corpus (38 GB), Stories from Common Crawl (31 GB)), applying dynamic token masking to word-in-a-sentence prediction task: word to be predicted in a given sentence changes with each epoch. RoBERTa has 124 million adjustable parameters (Fig. 15). Since result of direct propagation is vector sentence embedding matrix with the dimension [batch size, max sentence length, embedding dimension], hidden and fully connected output layer with 256 and 1 neurons, respectively, is added for further classification. To find the vector representation of sentences, the average value for each coordinate of the nesting vectors is obtained. The resulting matrix of weights between the obtained vector representations of words and a hidden layer with 256 neurons will have a dimension of 768*256, between the hidden layer and the output layer - 256*1. The activation function of the last layer is the sigmoid, the loss function is the binary cross-entropy. Pre-trained NN is further fine-tuned for the task of detecting paraphrasing in the text using training and validation samples. In total, NN was additionally trained for 50 epochs, the weights were preserved under the condition of the smallest value of the loss function on the validation sample (Fig. 16).

	Name	Type	Params
0	pretrained_model	RobertaModel	124 M
1	dropout_layer	Dropout	0
2	hidden_layer	Linear	196 K
3	output_layer	Linear	257
4	loss_function	BCEWithLogitsLoss	0

Figure 15: Total number of NN parameters

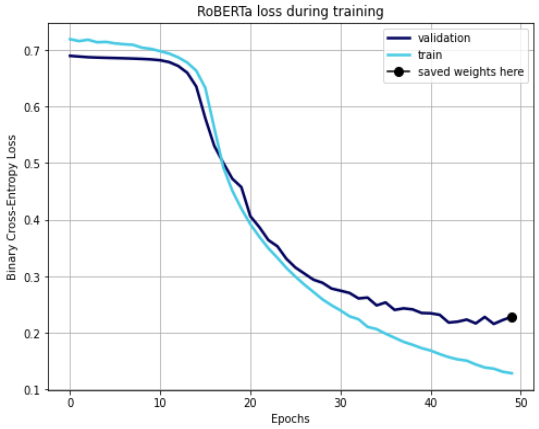


Figure 16: Loss function during NN training

For the final combination of features, both the probabilities of two records belonging to a certain class and the predictions themselves are selected. Decision threshold = 0.5, i.e. $\hat{y} = 1$ if p(y') > 0.5. The final table contains 10 features with the above metrics for each pair of sentences present in the selected dataset. These indicators are calculated for training, test and validation samples.

5. Results

To reduce the load on computing resources, another logistic regression model was trained without such features: prediction of Siamese NN and WordNet distance according to Wu and Palmer. Classification accuracy on the test sample increased from 92.4625% to 92.5%. For classification using NN RoBERTa, the data is provided in the following format: word indices, binary mask denoting part of the useful signal, true label corresponding to a pair of records as in Fig. 17. To obtain other signs, the data is reduced to the lower case, all symbols except letters are removed (Fig. 18). For prediction, the weights of the previously trained classifier (logistic regression) are loaded as shown in Fig. 19-20.

```
In [5]: dataset_test[0]
Out[5]: {'input_ids': tensor([    0,   713,    21,    10,   651,     9, 46902, 42970,  2820,  2156,
                98,    14, 19851,    11, 15001,   757,  5914,     8, 25361,   115,
                28,   626,  2024,    11, 13744, 34721,  5407,     7,     5,   364,
              3998, 10455,   636,   479,     2,     2,   713,    21,    10,   651,
                 9, 46902, 13744, 21423,  2156,    98,    14, 19851,    11, 15001,
               757,  5914,     8, 25361,   115,    28,  3744,  2024,    11, 42970,
             34721,  5407,     7,     5,   364,  3998, 10455,   636,   479,     2,
                 1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
                 1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
                 1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
                 1,     1,     1,     1,     1,     1,     1,     1,     1,     1,
                 1,     1,     1,     1,     1,     1,     1,     1]),
 'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0]),
 'labels': tensor(0.)}
```

Figure 17: Input data format to RoBERTa neural network

bert_cosine_distance	2_grams_jaccard	3_grams_jaccard	4_grams_jaccard	predictions_raw	predictions	shortest_path_distance	wm_distance
0.992338	0.5625	0.411765	0.314286	0.000308	0	0	0.163898

Figure 18: Received features for the test pair of sentences

```
import pickle
loaded_model = pickle.load(open('model_selected_features.pkl', 'rb'))
predicted = loaded_model.predict(X_test)

print('Predicted label', predicted)
```

Figure 19: Pre-trained model

```
Predicted label [0]
```

Figure 20: Prediction result

Classification result obtained for the next sentence is shown in Fig. 21. Logistic network has correctly predicted that two sentences are not paraphrases of each other, even though most of the words in sentences are in common (Figure 22). For the next sentence, logistic regression also has correctly predicted the label, in this case the pair of sentences are paraphrases of each other shown in Fig. 22.

```
['This was a series of nested angular standards , so that measurements in azimuth and elevation could be done directly in polar coordinates relative to the ecliptic .']
['This was a series of nested polar scales , so that measurements in azimuth and elevation could be performed directly in angular coordinates relative to the ecliptic .']
```

Figure 21: The first test pair of sentences

```
Predicted label 1
In [8]: print(df.loc[2, 'sentence1'])
   ...: print(df.loc[2, 'sentence2'])
   ...:
In January 2011 , the Deputy Secretary General of FIBA Asia , Hagop Khajirian , inspected the venue together with SBP - President Manuel V. Pangilinan .
In January 2011 , FIBA Asia deputy secretary general Hagop Khajirian along with SBP president Manuel V. Pangilinan inspected the venue .
```

Figure 22: The second test pair of sentences

Program does not require implementation of user interface, since its use is planned only as a module of the plagiarism detection system or aggregation of user-generated content.

6. Discussions

Classic ML-algorithm (logistic regression) was chosen for combining features and final classification. The results obtained are as follows:

- Accuracy on the test data set is 92.462%, the area under the ROC curve is 97.05%, and the area under the Precision-Recall curve is 94.96%.
- Accuracy on the validation data set - 93.71%, the area under the ROC curve is 97.66%, under the Precision-Recall curve - 96.12%.

According to accuracy, completeness, F0.5-measure (Fig. 23) and discrepancy matrix (Fig. 24) results of classification of test data set, logistic regression erroneously marked almost twice as many negative records as positive than vice versa, positive ones as negative. This result may be a consequence of the difficulty of determining a non-paraphrased pair of sentences that were formed as a result of replacing several words with places. Such sentences are very close semantically, but have completely different meanings.

	precision	recall	f1-score	support
0	0.95	0.91	0.93	4464
1	0.89	0.94	0.92	3536
accuracy			0.92	8000
macro avg	0.92	0.93	0.92	8000
weighted avg	0.93	0.92	0.92	8000

Figure 23: Recall, accuracy, F-score for the test data set

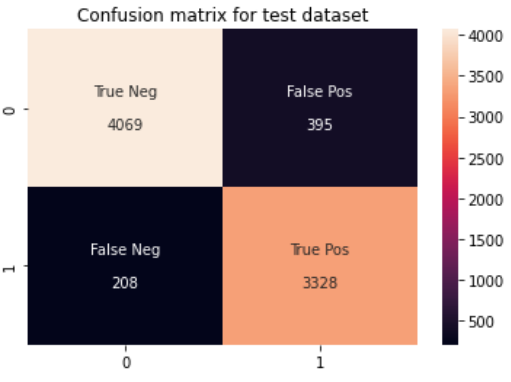


Figure 24: Confusion matrix for test dataset

According to bar chart in fig. 25, the most important features are RoBERTa ML model predictions, the cosine distance between solution vector representations, and the shortest path between WordNet dictionary synsets by Leacock and Chodorow. Significant coefficient of logistic regression, which corresponds to the predictions of the ML model, indicates its potential ability to qualitatively classify texts independently without the need to calculate additional features. Without applying deep learning methods, using only the logistic regression method to combine the results, we get: classification accuracy on the test data set is 71.15%, the area under the Precision-Recall curve is 73.6%.

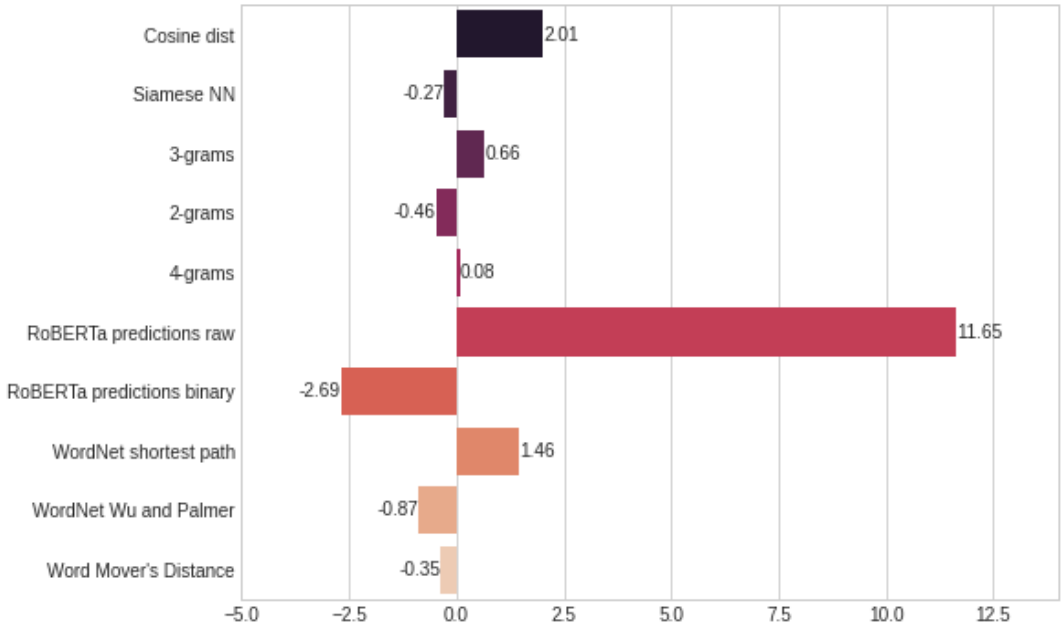


Figure 25: Weighting coefficients of logistic regression corresponding to certain signs of semantic similarity of a pair of sentences

The most important feature for classification is the Jaccard coefficient for 3-grams (the normalized number of common 3-grams) and the cosine distance between vector representations of sentences.

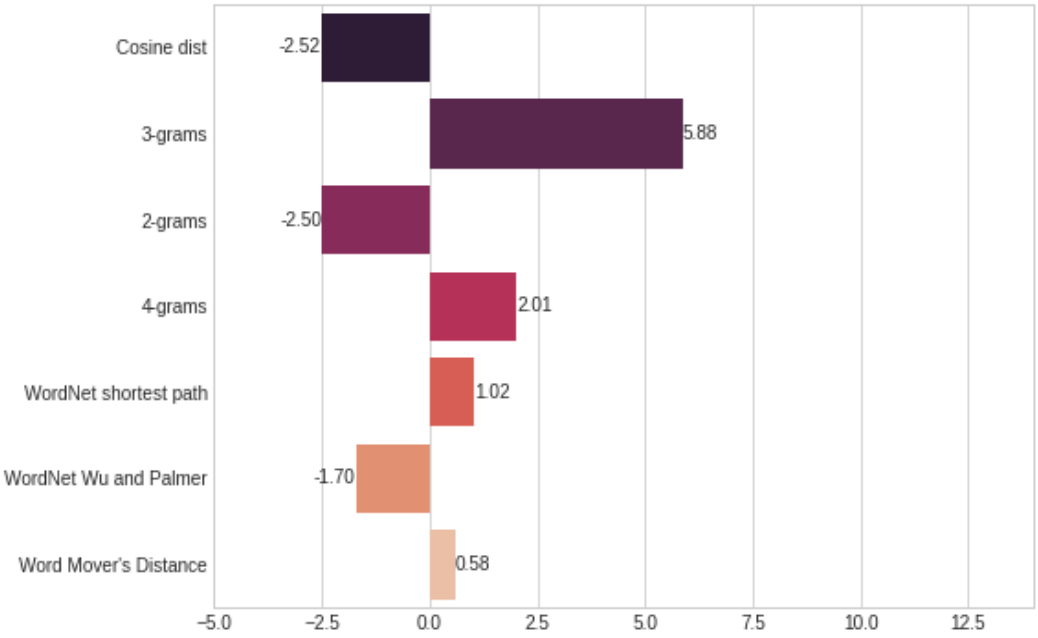


Figure 26: Weighting coefficients of logistic regression corresponding to certain signs of semantic similarity of a pair of sentences

In contrast to the previous result of classification using deep learning methods, in this case most of the positive cases were classified as negative, i.e. the logistic regression model "has difficulties" in identifying paraphrases. Use of only one feature is not sufficient for the qualitative classification of a pair of sentences, since the predictions of such a model are close in probability to "blind" guessing: the accuracy of the results of the classification of the test data set using Word Mover's Distance is 55.7%, the cosine distance between vector representations of sentences is 55.7% , Siamese NN predictions are 58%, while combining the same metrics with the Jaccard coefficient for 3-grams increases the classification accuracy to 70%, and applying the Jaccard coefficient for 2-grams adds another 1% accuracy. Despite the fact that these features can indicate the semantic similarity of sentences and can be used to detect pairs of similar sentences, the above-mentioned accuracy is not sufficient for full and high-quality detection of paraphrasing.

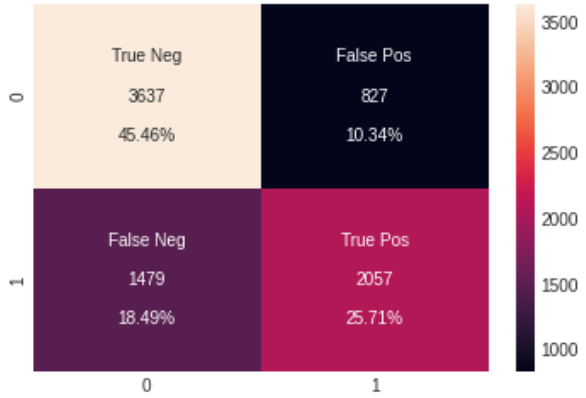


Figure 27: Confusion matrix for test dataset

Self-prediction of trained RoBERTa ML model is quite qualitative and has high quality indicators: accuracy – 91.96%, area under the Precision-Recall curve – 96.34%.

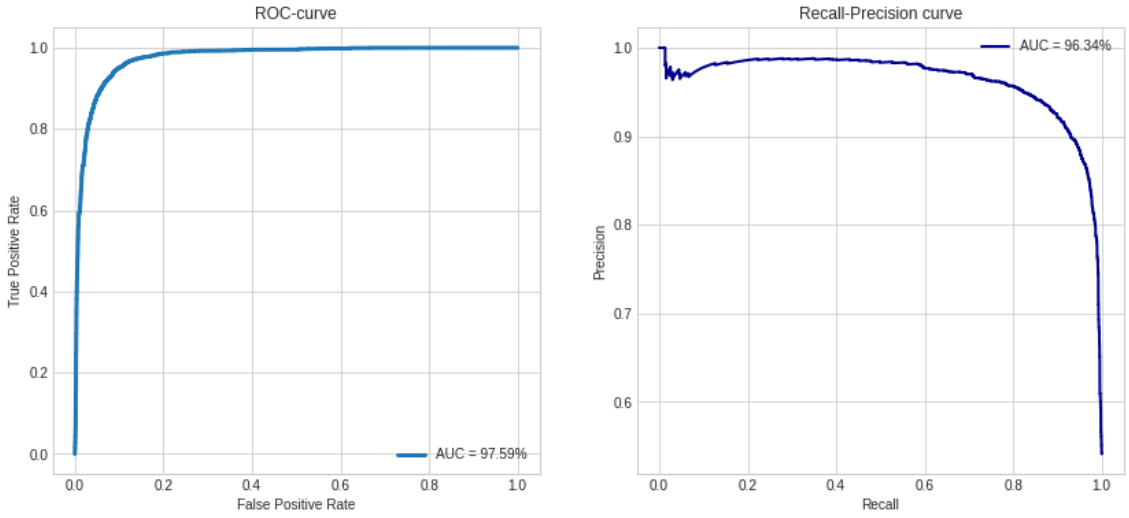
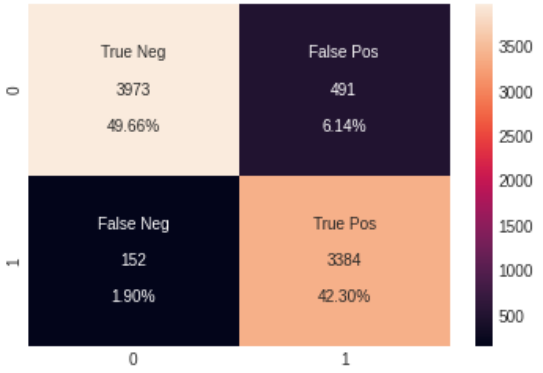


Figure 28: ROC and Precision-Recall curves for classification results of RoBERTa deep learning model

At the same time, RoBERTa classifies negative classes as positive classes much more (by 6 times) (Fig. 29), while significantly reducing the sensitivity / completeness (recall) to pairs of sentences that are not paraphrases of each other. When using such a model for plagiarism detection, more works will be misattributed as original sources, when used to aggregate user-generated content (questions, posts, forum threads), probability of misattributing a certain number of entries increases. In addition to comparison of ML methods, different number of features were tested simultaneously. Signs are selected depending on the appropriate weight of the logistic regression model. The best accuracy was obtained for the full set of features and the above algorithm. Other classical ML methods (tested naive Bayesian classifier, support vector method, random forest, k nearest neighbours multilayer perceptron) do not

give a significant increase in training accuracy. Regarding development of project, the following conclusions can be drawn:



	precision	recall	f1-score	support
0	0.96	0.89	0.93	4464
1	0.87	0.96	0.91	3536
accuracy			0.92	8000
macro avg	0.92	0.92	0.92	8000
weighted avg	0.92	0.92	0.92	8000

Figure 29: Discrepancy matrix, accuracy, completeness, F-measure

- For the method of stacking models and features, high indicators of accuracy, accuracy, completeness, F0.5-measure, areas under the ROC and Precision-Recall curves were obtained.
- "Classical" features of semantic similarity used in many studies are indeed capable of detecting semantically similar sentences, but they are "powerless" in cases where the sentence has permuted words (or there are many common words) and, accordingly, the second sentence is not is a paraphrase of the first and has a completely different meaning.
- Main premise of such a result is that the constructed features do not take into account the order of words in the sentence. To combat this, the Jaccard coefficient and the combination of features using classic ML methods are used.
- Pre-trained ML models based on the Transformer architecture show excellent classification accuracy. At the same time, the RoBERTa NN trained in the process of working (with additional fully connected layers) is less sensitive to pairs of sentences that are not paraphrases of each other. This specificity of model may contribute to incorrect accusations of plagiarism or incorrect association of user-generated content.
- Anti-plagiarism detection IS developers may be faced with choice between accuracy of classification and conservation of computing resources, since the calculation of features also contributes to the additional load on the computing device.
- NNs of the Transformer architecture do not require additional feature generation and are able to detect paraphrasing with fairly high accuracy. The disadvantage of this type of NN is a significant number of parameters (a long time to calculate the results).
- Transformer NNs have the advantage of pre-training models for "speech understanding" with the tasks of predicting the most likely words in a sentence and determining whether the second sentence is an idea continuation of the second.
- For the task of detecting paraphrases, the NN is "fine-tuned" on the Paraphrase Adversaries from Word Scrambling data set. The advantages of the selected data set are 1) a large number of educational records - 49 thousand 2) balanced classes: 44.2% of all pairs of sentences are paraphrases of each other. 3) some of the examples were formed by replacing or permuting words, in which case the sentences are close semantically, but have completely different meanings.
- Transformer NNs are used in many NLP tasks and can also be successfully used to detect paraphrasing in text with high accuracy. The only drawback of this type of networks is a significant number of configurable parameters 110+ million, training of such NN and its application require significant computing resources.

7. Conclusions

In the result of research ML-model for detecting paraphrasing by binary classification of texts pair has been developed. Software uses principle of model stacking and feature engineering. Additional features indicate the semantic affiliation of the sentences or the normalized number of common N-

grams. Created model shows excellent classification results on PAWS test data: weighted accuracy (precision) – 93%, weighted completeness (recall) – 92%, F-measure (F1-score) – 92%, accuracy (accuracy) – 92%. The study results have shown that Transformer-type NNs can be successfully applied to detect paraphrasing in a pair of texts with fairly high accuracy without the need for additional feature generation. Fine-tuned NN RoBERTa (with additional fully connected layers) is less sensitive to pairs of sentences that are not paraphrases of each other. This model specificity may contribute to incorrect accusations of plagiarism or incorrect association of user-generated content. Additional features increase both the overall classification accuracy and the model sensitivity to sentences pairs that are not paraphrases of each other.

8. References

[1] G. Salton, A. Wong, C.-S. Yang, A vector space model for automatic indexing, Communications of the ACM 18(11) (1975) 613–620. DOI: 10.1145/361219.361220.

[2] P. D. Turney, P. Pantel, From Frequency to Meaning: Vector Space Models of Semantics, Journal of Artificial Intelligence Research 37(1) (2010) 141-188. DOI: 10.1613/jair.2934.

[3] T. Mikolov, K. Chen, G. s Corrado, J. Dean, Efficient Estimation of Word Representations in Vector Space, ArXiv, 2013. DOI: 10.48550/arXiv.1301.3781.

[4] Do Online Plagiarism Checkers Identify Paraphrased Content? DotNek Software Development, 2021. – https://www.dotnek.com/Blog/Marketing/do-online-plagiarism-checkers-identify-paraph.

[5] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. Miller, Introduction to WordNet: An On-line Lexical Database, International Journal of Lexicography 3(4) (1990) 235–244. DOI: 10.1093/ijl/3.4.235.

[6] C. Corley, R. Mihalcea, Measuring the Semantic Similarity of Texts, ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Ann Arbor, Michigan, 2005, pp. 13–18.

[7] C. Leacock, M. Chodorow, Combining Local Context and WordNet Similarity for Word Sense Identification, WordNet: An Electronic Lexical Database 49(2) (1998) 265--283. DOI: 10.7551/mitpress/7287.003.0018.

[8] M. E. Lesk, Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone, Annual International Conference on Systems documentation, Toronto, Ontario, Canada, 1986, pp. 24-26. DOI: 10.1145/318723.318728.

[9] Z. Wu, M. Palmer, Verbs Semantics and Lexical Selection, Annual meeting on Association for Computational Linguistics, Las Cruces, New Mexico, 1994, pp. 133–138. DOI: 10.3115/981732.981751.

[10] P. Resnik, Using Information Content to Evaluate Semantic Similarity in a Taxonomy, ArXiv, 1995. DOI: 10.48550/arXiv.cmp-lg/9511007.

[11] D. Lin, An Information-Theoretic Definition of Similarity, ICML, 1998. – https://www.cse.iitb.ac.in/~cs626-449/Papers/WordSimilarity/3.pdf.

[12] J. J. Jiang, D. W. Conrath, Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy, Research on Computational Linguistics International Conference, Taiwan, 1997, pp. 19–33.

[13] R. Mihalcea, C. Corley, C. Strapparava, Corpus-based and Knowledge-based Measures of Text Semantic Similarity, Artificial intelligence, Boston, Massachusetts, 2006, pp. 775–780.

[14] S. Hassan, R. Mihalcea, Semantic Relatedness Using Salient Semantic Analysis, Conference on Artificial Intelligence, 2011, https://web.eecs.umich.edu/~mihalcea/papers/hassan.aaai11.pdf.

[15] S. Fernando, M. Stevenson, A Semantic Similarity Approach to Paraphrase Detection, Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics, 2008, pp. 45-52.

[16] D. Milajevs, D. Kartsaklis, M. Sadrzadeh, M. Purver, Evaluating Neural Word Representations in Tensor-Based Compositional Settings, ArXiv, 2014. DOI: 10.48550/arXiv.1408.6179.

[17] A. Islam, D. Inkpen, Islam A. Semantic similarity of short texts, Current Issues in Linguistic Theory: Recent Advances in Natural Language Processing 309 (2009) 227–236. DOI: 10.1075/cilt.309.18isl.

[18] M. Chong, L. Specia, R. Mitkov, Using Natural Language Processing for Automatic Detection of Plagiarism, in: International Plagiarism Conference, Newcastle-upon-Tyne, May 2010,

https://www.academia.edu/326444/Using_Natural_Language_Processing_for_Automatic_Detection_of_Plagiarism.
[19] F. Šarić, G. Glavaš, M. Karan, J. Šnajder, B. Dalbelo Bašić, TakeLab: Systems for Measuring Semantic Text Similarity, in: Lexical and Computational Semantics, Montréal, 2012, pp. 441–448.
[20] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity, in: The First Joint Conference on Lexical and Computational Semantics, Montréal, Canada, 2012, pp. 385–393.
[21] L. Kong, Z. Lu, H. Qi, Z. Han, Detecting High Obfuscation Plagiarism: Exploring Multi-Features Fusion via Machine Learning. International Journal of u- and e- Service, Science and Technology 7 (2014) 385–396. DOI: 10.14257/ijunnesst.2014.7.4.35.
[22] W. Yin, H. Schütz, Convolutional Neural Network for Paraphrase Identification, in: North American Chapter of the Association for Computational Linguistics: Human Language Technologies Conferences, Denver, Colorado, 2015, pp. 901–911. DOI: 10.3115/v1/N15-1091.
[23] L. Qiu, M.-Y. Kan, T.-S. Chua, Paraphrase Recognition via Dissimilarity Significance Classification, in: Empirical Methods in Natural Language Processing Conference, Sydney, Australia, 2006, pp. 18–26.
[24] Z. Kozareva, A. Montoyo, Paraphrase Identification on the Basis of Supervised Machine Learning Techniques, Lecture Notes in Computer Science 4139 (2006) 524–533. DOI: 10.1007/11816508_52.
[25] A. Finch, E. Sumita, Using machine translation evaluation techniques to determine sentence-level semantic equivalence, in: International Workshop on Paraphrasing, 2005, pp. 17-24.
[26] N. Madnani, J. Tetreault, M. Chodorow, Re-examining Machine Translation Metrics for Paraphrase Identification, in: North American Chapter of the Association for Computational Linguistics: Human Language Technologies Conference, Montréal, Canada, 2012, pp. 182–190.
[27] B. Agarwal, H. Ramampiaro, H. Langseth, M. Ruocco, A Deep Network Model for Paraphrase Detection in Short Text Messages, ArXiv, 2017. DOI: 10.48550/arXiv.1712.02820.
[28] R. Socher, E. H.-C. Huang, J. Pennington, A. Ng, C. D. Manning, Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection, in: International Conference on Neural Information Processing Systems, Granada Spain, December 2011, pp. 801–809.
[29] A. Thyagarajan, Siamese Recurrent Architectures for Learning Sentence Similarity, Artificial Intelligence 30(1) (2016) 2786-2792. DOI: 10.1609/aaai.v30i1.10350.
[30] P. Neculoiu, M. Versteegh, M. Rotaru, Learning Text Similarity with Siamese Recurrent Networks, in: Workshop on Representation Learning for NLP, Berlin, Germany, 2016, pp. 148–157. DOI: 10.18653/v1/W16-1617.
[31] T. Ranasinghe, C. Orasan, R. Mitkov, Semantic Textual Similarity with Siamese Neural Networks, in: International Conference on Recent Advances in Natural Language Processing, Varna, Bulgaria, Sep. 2019, pp. 1004–1011. DOI: 10.26615/978-954-452-056-4_116.
[32] A. Mahmoud, M. Zrigui, BLSTM-API: Bi-LSTM Recurrent Neural Network-Based Approach for Arabic Paraphrase Identification, in: Arabian Journal for Science and Engineering 46 (2021) 4163–4174. DOI: 10.1007/s13369-020-05320-w.
[33] D. Reddy, M. Kumar, K.P. Soman, LSTM Based Paraphrase Identification Using Combined Word Embedding Features, Advances in Intelligent Systems and Computing 898 (2019) 385–394. DOI: 10.1007/978-981-13-3393-4_40.
[34] Z. Li, X. Jiang, L. Shang, H. Li, Paraphrase Generation with Deep Reinforcement Learning, ArXiv, 2017. DOI: 10.48550/arXiv.1711.00279.
[35] W. Gomaa, A. Fahmy, SimAll: A flexible tool for text similarity, in: Language Engineering, 2017, https://www.academia.edu/35381793/SimAll_A_flexible_tool_for_text_similarity.
[36] M. Ahmed, M. R. Samee, R. E. Mercer, Improving Tree-LSTM with Tree Attention, ArXiv, 2019. DOI: 10.48550/arXiv.1901.00066.
[37] E. L. Pontes, S. Huet, A. C. Linhares, J.-M. Torres-Moreno, Predicting the Semantic Textual Similarity with Siamese CNN and LSTM, Actes de la Conférence TALN, 2018, pp. 311–320.
[38] J. P. Wahle, T. Ruas, N. Meuschke, B. Gipp, Are Neural Language Models Good Plagiarists? A Benchmark for Neural Paraphrase Detection, ArXiv, 2021. DOI: 10.48550/arXiv.2103.12450.
[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention Is All You Need, ArXiv, 2017. DOI: 10.48550/arXiv.1706.0376.

[40] A. Nighojkar, J. Licato, Improving Paraphrase Detection with the Adversarial Paraphrasing Task, ArXiv, 2021. DOI: 10.48550/arXiv.2106.07691.
[41] Y. Arase, J. Tsujii, Transfer fine-tuning of BERT with phrasal paraphrases, ArXiv, 2021. DOI: 10.48550/arXiv.1909.00931.
[42] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, ArXiv, 2018. DOI: 10.48550/arXiv.1810.04805.
[43] M. J. Kusner, Y. Sun, N. I. Kolkin, K. Q. Weinberger, From Word Embeddings to Document Distances, JMLR: W&CP 37 (2015) 957-966.
[44] Y. Zhang, J. Baldridge, L. He, Zhang Y. PAWS: Paraphrase Adversaries from Word Scrambling, ArXiv, 2019. DOI: 10.48550/arXiv.1904.01130.
[45] V.-A. Oliinyk, V. Vysotska, Y. Burov, K. Mykich, V. B. Fernandes, Propaganda Detection in Text Data Based on NLP and Machine Learning, CEUR workshop proceedings 2631 (2020) 132-144.
[46] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A Robustly Optimized BERT Pretraining Approach, ArXiv, 2019. DOI: 10.48550/arXiv.1907.11692.
[47] V. Vysotska, Y. Burov, V. Lytvyn, A. Demchuk, Defining Author's Style for Plagiarism Detection in Academic Environment, in: Proceedings of the International Conference on Data Stream Mining and Processing, DSMP, 2018, pp. 128-133. DOI: 10.1109/DSMP.2018.8478574.
[48] N. Shakhovska, I. Shvorob, The method for detecting plagiarism in a collection of documents, in: International Conference on Computer Sciences and Information Technologies, 2015, pp. 42-145.
[49] O. Karnalim, G. Kurniawati, Programming Style on Source Code Plagiarism and Collusion Detection, International Journal of Computing 19(1) (2020) 27-38.
[50] N. Khairova, A. Shapovalova, O. Mamyrbayev, N. Sharonova, K. Mukhsina, Using BERT model to Identify Sentences Paraphrase in the News Corpus, CEUR Workshop Proceedings Vol-3171, (2022) 38-48.
[51] N. Grabar, T. Hamon, Exploitation of the morphology for automatic extraction of general paraphrases of medical terms, Revue Traitement Automatique des Langues 57(1) (2016) 85–109.
[52] I. Eshkol-Taravella, N. Grabar, Paraphrastic reformulations in spoken corpora, Lecture Notes in Computer Science 8686 (2014) 425–437.
[53] P. Zdebskyi, et. al., Intelligent System for Semantically Similar Sentences Identification and Generation Based on Machine Learning Methods, CEUR workshop proceedings Vol-2604 (2020) 317-346.
[54] V. Shynkarenko, I. Demidovich, Natural Language Texts Authorship Establishing Based on the Sentences Structure, CEUR Workshop Proceedings Vol-3171 (2022) 328-337.
[55] V. Shynkarenko, I. Demidovich, Authorship Determination of Natural Language Texts by Several Classes of Indicators with Customizable Weights, CEUR Workshop Proceedings Vol-2870 (2021) 832-844.
[56] I. Khomytska, V. Teslyuk, The Multifactor Method Applied for Authorship Attribution on the Phonological Level, CEUR workshop proceedings Vol-2604 (2020) 189-198.
[57] I. Khomytska, V. Teslyuk, Authorship and Style Attribution by Statistical Methods of Style Differentiation on the Phonological Level, Advances in Intelligent Systems and Computing 871 (2019) 105–118. doi: 10.1007/978-3-030-01069-0_8.
[58] I. Khomytska, V. Teslyuk, I. Bazylevych, I. Shylinska, Approach for Minimization of Phoneme Groups in Authorship Attribution, International Journal of Computing 19(1) (2020) 55-62.
[59] I. Khomytska, V. Teslyuk, Statistical models for authorship attribution, Advances in Intelligent Systems and Computing 1080 (2020) 579–592.
[60] I. Khomytska, V. Teslyuk, Authorship Attribution by Differentiation of Phonostatistical Structures of Styles, in: CSIT of the Scientific and Technical Conference, Lviv, 2018, pp. 5–8.
[61] I. Khomytska, V. Teslyuk, The Software for Authorship and Style Attribution, in: CADMS: proceedings of the 15th International Conference, Polyana, 2019, pp. 23–26.
[62] I. Khomytska, V. Teslyuk, Mathematical Methods Applied for Authorship Attribution on the Phonological Level, in: CSIT: Scientific and Technical Conference, Lviv, 2019, pp. 7–11.
[63] I. Khomytska, V. Teslyuk, L. Bordyuk, The Kolmogorov-Smirnov's Test for Authorship Attribution on the Phonological Level, in: International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2020, pp. 259–262.