Models and Algorithms for Automatic Labelling of Unstructured Texts (Text Tagging)

Gyuzel Shakhmametova and Ilshat Ishmukhametov
Computer Science and Robotics Department, Ufa State Aviation Technical University, K. Marks Str. 12, 450008 Ufa, Russian Federation
shakhgouzel@mail.ru, mail@ishmukhamet.xyz

Keywords: Automatic Labelling of Texts, Unstructured Text, Text Tagging, Multilabel Classification, Keywords Extraction.

Abstract: The article discusses the task of automatic labelling of texts to improve the efficiency of processing unstructured text data. An overview of existing software products for solving the problem is given, showing the need to develop its own solution specialized in the processing of Russian-language texts. The problem of assigning labels is considered from a mathematical point of view as a problem of multilabel classification, with corresponding mathematical models analysed and described. Based on this, models, algorithms, and a software product for automatically assigning labels to texts have been developed. Numerical experiments were carried out that showed the universality of the method and the possibility of application both in non-specialized and specialized fields, in particular, for processing medical documents.

1 INTRODUCTION

Information systems are becoming more and more loaded and complex year after year, and the volume of information is growing at a tremendous rate. According to experts [1], by 2025 the volume of accumulated information will reach 175 zettabytes compared to 33 zettabytes in 2018 (1 zettabyte = 10^{12} gigabytes = 1 trillion gigabytes).

At the same time, most of the information is stored in unstructured form, mainly, as texts. In particular, among medical documents, structured data account for up to 20% of all available information [2].

In such circumstances, natural language processing tools are needed, since structuring the accumulated data significantly increases the efficiency of their use.

In working with text information, several large tasks may be distinguished, such as categorization task, identification of authorship, extraction of keywords and sentences, extraction of emotional context etc.

This article discusses the task of automatic assigning labels to texts, i.e., the so-called text tagging task, where each document from the corpus (set of texts) is mapped to tags (keywords, labels) from a certain set, helping to determine the content or purpose of the considered document.

Automatic labelling of texts is an urgent problem, since its implementation is necessary when solving problems in a variety of areas: in recommendation systems, electronic document management, in knowledge bases, etc. Text tagging tasks can vary significantly in specific cases and depend on the purpose, subject area, type, quantity and format of documents, language, etc. Accordingly, the methods of solving this problem can vary, making the choice of the appropriate method even more complicated.

The second part of the article presents related works in the field of text tagging; the third part is devoted to setting the task; the fourth part describes the proposed solution; results are discussed in part 5, and the sixth part contains the main conclusions.

2 RELATED WORKS

The relevance of the text tagging task contributes to its study by many researchers. As a result, several methods have been developed to solve it.

TextRank [3] is an adaptation of the PageRank [4] algorithm developed by Google to rank web pages. PageRank, in general, can be used to rank any group

of objects represented as a graph. TextRank converts the text into a graph and extracts keywords or sentences from it.

LDA (Dirichlet Latent Allocation) [5] uses the concept of hidden groups to determine text topics. It is assumed that each document may address several topics, and the appearance of a word in the text is related to one of these topics. In this way, labels can be assigned to the text corresponding to the themes of the package. A modification of LDA has also been developed to work with bigrams (two-word phrases) [6].

The RAKE (Rapid Automatic Keyword Extraction) [7] algorithm is based on the observation that keywords often consist of several words and, as a rule, do not include stop words (service parts of speech and the most used words). RAKE extracts phrases from the text using stop words as delimiters, and then counts estimates for them depending on how often words from these combinations are found in the document. Combinations with the highest scores are selected as keywords.

An algorithm has also been developed that simultaneously uses graph representation of text, LDA and RAKE [8].

In the application software market, there are many finished products for automatic tagging of texts.

- Dcipher Analytics [9] is a set of analytical tools for working with data, including text analysis. As the main areas of its work, Dcipher identifies Data Mining in the field of social media, automatic image processing, analysis of customer opinions, analysis of processes in the enterprise, etc. The platform provides the ability to build pipelines of data processing from a set of ready-made operations: importing data, collecting statistics, cleaning, and filtering data, training a model, etc. [9]
- MonkeyLearn [10] provides a service for automatic analysis of text data, such as automatic tagging, routing, and prioritization of requests from customers, analysis of user reviews, determination of customer mood, etc. The platform provides a comprehensive set of ready-made models from the built-in set [10].
- TwinWord [11] is a set of tools for developing texts based on the extraction and analysis of keywords, including analysis of the emotional connotation of the text, classification of texts, recommendation systems [11].

The main disadvantage of most such products is low flexibility in configuring the models to be used. Most often, only ready-made templates can be employed. Almost all existing solutions are close-ended, most of them have to be purchased for a fee, and, as a rule, they require an additional configuration. Additionally, there are practically no ready-made solutions focused on Russian-language texts.

All solutions available on the market are SaaS-based and provide APIs for integration into their own products.

A comparison of the software product characteristics described above is shown in Table 1.

Table 1: Product Comparison Results.

	Dcipher Analytics	Monkey Learn	TwinWord
Russian language	—	—	—
Documentation	—	Sufficient	Not sufficient
Price	From $3600 per year + trial	From $3600 per year + trial	Depends on the number of requests
Model Flexibility	High	Low	Low
Usability	Low	High	High

As a result of the analysis of the current state of research and software solutions in this field, it may be concluded that to fully support Russian-language texts and the ability to flexibly customize models, it is necessary to develop specific targeted solutions to the text tagging problem with the final implementation in the form of software.

3 PROBLEM DEFINITION

The product required for development should allow to create, edit, and delete text documents, label them, and merge them into collections. When new documents are added to existing collections, they must be automatically labelled (Figure 1).

Figure 1: Task Setting.

Assigning labels to texts fits the classification task definition: given a set of classified objects $X = x_1, \dots, x_n$ (i.e. corpus of texts) and a set of classes $C = \{c_1, \dots, c_m\}$ (i.e., label set). Objects and classes are related by a $\Phi: X \to C$ relationship.

Labelling in this formulation is suitable, for example, for categorizing documents. In this way, news may be classified ascribing each one of them into one of the categories such as «society», «politics», «sports», etc.

However, labelling in practice usually implies that documents are assigned more than one label. Returning to the above example, the same news item can simultaneously have labels «politics» and «economy».

Such a relationship between objects and classes may be described as follows (1):

$$\Phi: X \times C \to Y = \{0; 1\}^{|C|}, \quad (1)$$

where $y_{ij} = 1$ means that the object x_i belongs to the class c_j.

This describes the task of multilabel classification, i.e., such classification when one object can belong to several classes [12].

Thus, the task of text tagging is to construct a classifier (2):

$$\Phi': X \times C \to Y = \{0; 1\}^{|C|}, \quad (2)$$

where $X = \{x_1, \dots, x_n\}$ is a corpus of documents, $C = \{c_1, \dots, c_m\}$ is as set of labels, and $y_{ij} = 1$ means that the label c_j assigned to the document x_i, while Φ is the desired dependency between documents and labels.

4 SUGGESTED SOLUTION

There are several approaches to solving the problem of multilabel classification, which, in fact, are approaches with training potential.

- Reduction to binary classification [13]: its own binary classifier is built for each label separately, and the final set of labels for the document is created by determining which of these classifiers will give a positive result. It should be noted that such a solution loses some information, since correlations between labels are not taken into account.
- Reduction to multiclass classification [14]: in this case, label sets assigned to documents are perceived as separate classes. For example, for a set of two labels, $C = \{[0,0], [0,1], [1,0], [1,1]\}$ will be considered classes. A clear disadvantage of this approach is the large computational costs (exponential complexity) and the tendency to retrain, since not all possible sets of labels may occur in test data.
- Adaptation of multiclass classification methods is based on multilabel variations of the methods of kNN (ML-kNN [15]), decision trees (modification of the algorithm C4.5 [16]), and artificial neural networks (BP-MLL [17]).

As part of this work, adapted methods are considered because they take into account the correlation between assigned labels and are almost as computation-efficient as their multiclass counterparts.

In general, as with the other cases of using methods with training potential, the proposed solution can be divided into the following steps (Figure 2):

- preparation of data – the corpus of documents;
- feature extraction;
- model training;
- labelling new documents using a trained model.

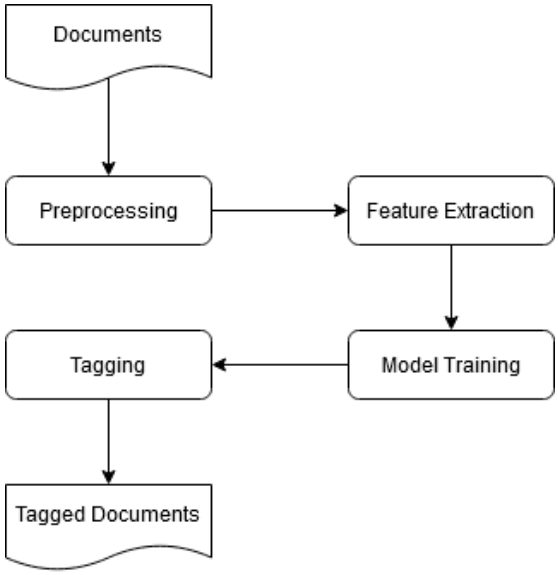

Figure 2: Algorithm of Text Tagging Task Solving as Multilabel Classification.

Let us take a closer look at each step.

4.1 Preprocessing

Among the obvious difficulties while solving this problem, one can distinguish the congestion of any text with service parts of speech: prepositions, conjunctions, particles, etc. They do not significantly affect the formation of the text topic but prevent the selection of keywords by frequency. Therefore, it is necessary to pre-process the text which is called normalization.

It is necessary to carry out lemmatization [18] – to bring all words to lemmas, their initial forms. For

example, for the word «stimulates», the lemma will be «stimulate», for «analysing» – «analyse».

An alternative to lemmatization is stemming – finding the basis (stem) of the word. For example, for the word «regulated», the stem will be «regul» which will allow to find such word forms as «regulate», «regulating», «regulation», etc.

The text normalization algorithm also includes the removal of punctuation signs and special characters, tokenization (division into word lists), the removal of stop words (using prepared in advance stop word lists) and the bringing of words to initial forms (using prepared in advance dictionaries).

Thus, the normalization of the text may happen according to the following algorithm (Figure 3):

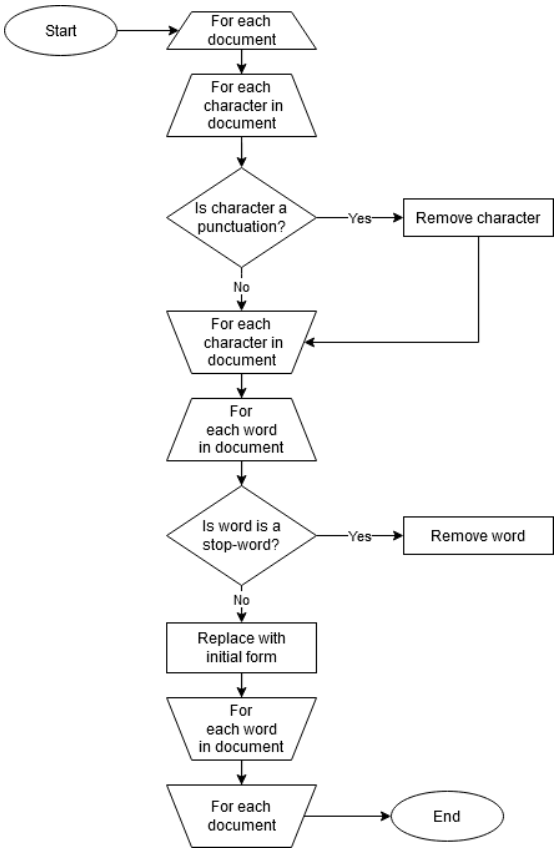

Figure 3: Text Corpus Preprocessing Algorithm.

4.2 Feature Extraction

TF-IDF model is used to extract features from pre-processed text corpus [19].

TF (Term Frequency) is the ratio for the number of occurrences of a given word to the total number of words in the text. The importance of the word is evaluated by the following (3):

$$tf(t,d) = \frac{n_t}{\sum_k n_k} \quad (3)$$

where n_t is the number of occurrences of the word t in the document d, and the sum in the denominator is the total number of words in the document.

IDF (Inverse Document Frequency) is an inversion of the frequency with which a word occurs in body texts. Accounting for this indicator reduces the weight of words often used. Each word within a document collection has a value (4):

$$idf(t,D) = log\frac{|D|}{|d_i \in D : t \in d_i|} \quad (4)$$

where $|D|$ is the total number of documents in collection D, and the denominator is the number of documents in the collection in which the word t appears.

The TF-IDF measure is calculated as the product of the multipliers (5):

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D) \quad (5)$$

4.3 Learning Algorithms

To train models in the work, the algorithms ML-kNN, Decision Tree and Random Forest were used.

4.3.1 ML-kNN

ML-kNN is a modification of the kNN method for multilabel classification. ML-kNN first determines the k nearest neighbours of the object. For those, it is already known which classes they belong to. Then, based on the maximum a posteriori estimation (MAP), it determines which labels to assign to the object in question.

The object x from the test sample with the label set Y_x is considered. Let $\overrightarrow{y_x}$ be the label vector for x, where the l-th component $\overrightarrow{y_x}(l)$ is 1 if the label l is assigned to the object x, that is, if $l \in Y_x$, and 0 otherwise. Let $N(x)$ be the set of indices k closest to x neighbors from the training sample.

Then, knowing a set of tags of these neighbours, we can define a membership counting vector (6):

$$\overrightarrow{C_x}(l) = \sum_{a \in N(x)} \overrightarrow{y_a}(l), l \in Y \quad (6)$$

That is, this vector counts the number of neighbours labelled l.

Let the event H_1^l indicate that the object in question has the label l, and H_0^l indicates that it does not. Let the event $E_j^l, j \in \{0, \dots, k\}$ show that among the k closest neighbours of object x there are exactly j objects with the label l.

Then, based on the membership calculation vector and using the maximum a posteriori estimation, it is possible to determine the vector of labels $\overrightarrow{y_x}$ for $l \in Y$ in this way:

$$\overrightarrow{y_x}(l) = argmax_{b \in \{0,1\}} P\left(H_b^l \left| E^l_{\overrightarrow{C_x}(l)} \right.\right) \quad (7)$$

Using Bayes' theorem, this expression can be brought to the form:

$$\overrightarrow{y_x}(l) = argmax_{b\in\{0,1\}} \frac{P\left(H_b^l\right)\cdot P\left(E^l_{\overrightarrow{C_x}(l)}\middle|H_b^l\right)}{P\left(E^l_{\overrightarrow{C_x}(l)}\right)} \quad (8)$$

Probability $P\left(E^l_{\overrightarrow{C_x}(l)}\right) = 1$, therefore:

$$\overrightarrow{y_x}(l) = argmax_{b\in\{0,1\}} P\left(H_b^l\right)\cdot P\left(E^l_{\overrightarrow{C_x}(l)}\middle|H_b^l\right) \quad (9)$$

Probabilities $P\left(H_b^l\right)$ and $P\left(E^l_{\overrightarrow{C_x}(l)}\middle|H_b^l\right)$ can be calculated on a training sample.

4.3.2 Decision Tree

The C4.5 algorithm uses the concepts of entropy and information gain criteria to determine an attribute for better splitting a training set into a tree.

Let be given a training set S containing m attributes and n objects belonging to k classes. The tree is built from the root node to the leaves, that is, from top to bottom.

In the first step, an empty tree is built, consisting only of a root that includes the entire set S.

Next, the root is split into subsets and child nodes are defined. To do this, one of the attributes is selected and a rule is formed that breaks the set of objects into p subsets, where p is the number of unique values of the selected attribute. The procedure is then repeated for each of the received subsets and the child nodes. The procedure continues until the stop condition is reached.

Let $N\left(c_j, S\right)$ denote the number of objects of class c_j in the set S, and $N(S)$ denote the total number of examples in the set S. Then, the relative frequency of class c_j in the set S can be determined:

$$p\left(c_j\right) = \frac{N(c_j,S)}{N(S)} \quad (10)$$

Variable

$$H(S) = -\sum_{i=1}^{k}[p(c_i)\cdot log(p(c_i))] \quad (11)$$

is an entropy of a set S and shows the average amount of information needed to determine an object class from that set.

After dividing the set by attribute A, this estimate can be written as:

$$H_A(S) = \sum_{i=1}^{k}[p(c_i)\cdot H(S_i)], \quad (12)$$

where S_i is the i-th node that was obtained during the partition. Then the best split attribute can be selected using the information gain criterion:

$$IG(A) = H(S) - H_A(S) \quad (13)$$

For partitioning, an attribute is selected, for which the gain in information is the greatest.

If an empty node is formed during the split process, it becomes a sheet, and a class which more often was met among objects of the parent node G is associated with it.

The above formulas apply to discrete attributes. In the case of a continuous attribute having n different values, the set of its values is divided into n subsets using $(n-1)$ threshold values. Using the information gain criterion, the threshold value that gives the largest information gain is selected.

To use the C4.5 algorithm for multilabel classification, the entropy count is changed as follows:

$$H'(S) = -\sum_{i=1}^{k}(p_i + q_i), \quad (14)$$

where $p_i = p(c_i)\cdot \log p(c_i)$, $q_i = q(c_i)\cdot \log q(c_i)$, $q(c_i) = 1 - p(c_i)$.

4.3.3 Random Forest

The standard implementation of the random forest method with trees described in paragraph 4.3.2 was used.

For a training sample S of size N with M attributes, a random forest is described as:

$$\{h(x,\Theta_k), k = 1, \ldots\}, \quad (15)$$

where $h(x,\Theta_k)$ is a separate tree built on a subset Θ_k of the training set.

The forest building algorithm includes the following steps:

1) From the training sample S, a subset Θ_k of size N is randomly generated with repetitions: some objects will be included more than once, some will not be included at all.
2) On the obtained sub-sample, a tree $h(x,\Theta_k)$ is built using not the entire set of features, but only m randomly selected.

Thus, several trees are built. Their optimal number is selected to minimize classification errors on the test sample.

5 RESULTS

The above algorithms were used to implement the software TextTagger [20], designed to automatically assign labels to texts.

The machine learning module is implemented using Python and the main libraries for machine learning and NLP (nltk, sclearn, etc). The business logic module is implemented on the .NetCore platform, while the web client is developed using the Vue framework.

During operation, the user generates collections consisting of text documents. The first added documents are labelled manually, and later this initial classification will be used to train a model.

After adding several documents to the collection, one of the above models (ML-kNN, Decision Tree or Random Forest) can be trained. The user selects the model manually.

When a new document is added to a collection with a trained model, the system automatically prompts to assign the most appropriate labels to it.

Figure 4 and Figure 5 show the main steps of working with the system using the example of accumulating a collection of medical documents.

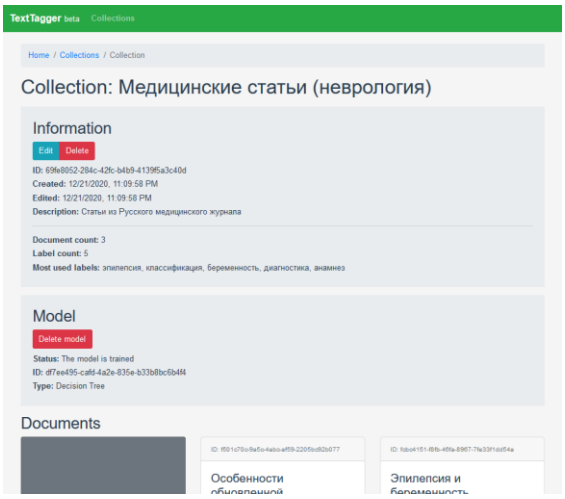

Figure 4: Collection Overview Page.

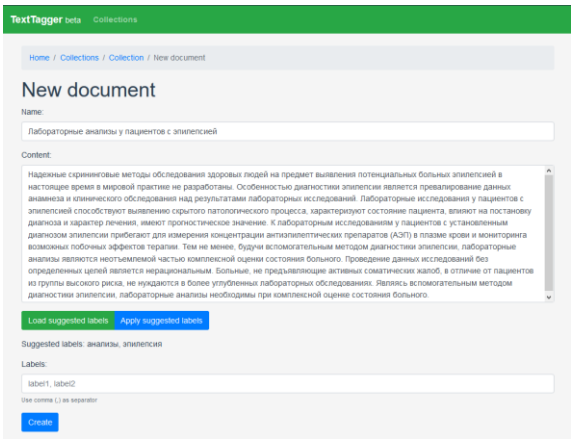

Figure 5: Adding a New Document with the System Proposed Labels.

Based on experimental data, the quality of the algorithms was analysed. Accuracy (A), precision (P), recall (R) and F1-score ($F1$) are used as quality metrics [21].

$$A = \frac{1}{n_i}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}, \quad (16)$$

$$P = \frac{1}{n_i}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Z_i|}, \quad (17)$$

$$R = \frac{1}{n_i}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i|}, \quad (18)$$

$$F = \frac{1}{n_i}\sum_{i=1}^{n}\frac{2\cdot|Y_i \cap Z_i|}{|Y_i|+|Z_i|}, \quad (19)$$

where $Y_i = \{0;1\}^k$ are the labels assigned to the document x_i (actual) and $Z_i = \{0;1\}^k$ are the labels predicted by the model.

The closer the scores are to 1, the higher the quality of the model.

Several prepared in advance and labelled datasets from different subject areas were used to conduct quality analysis of algorithms:

Set A: abstracts from literature texts; labels are age groups to which texts are oriented (for children, for adults, for children and adults at the same time). This dataset consists of 75 documents, the labels are evenly distributed.

Set B: synopsis of films; tags are their genres (drama, comedy, etc). This dataset consists of 50 documents, the labels are unevenly distributed.

Set C: annotations to articles from the Russian Medical Journal; labels are medical concepts referred to in the articles (cognitive impairment, therapy, etc. This dataset consists of 56 documents, the labels are unevenly distributed.

The quality metrics for the Decision Tree, Random Forest and ML-kNN algorithms are shown in Table 2.

Table 2: Quality Metrics.

Method	Data set	A	P	R	F1
DT	A	0.94	0.97	0.97	0.96
	B	0.27	0.50	0.41	0.38
	C	0.36	0.46	0.42	0.41
RF	A	0.78	0.86	0.80	0.81
	B	0.25	0.65	0.27	0.36
	C	0.23	0.27	0.26	0.26
ML-kNN	A	0.94	0.94	1.0	0.96
	B	0.32	0.56	0.38	0.42
	C	0.60	0.69	0.64	0.64

The quality of the models used depends significantly on the specific training data, especially on the uniformity of the distribution by labels. However, overall quality allows models to be used to solve some of the practical problems or as a basis for further development and improvement.

Random Forest (metric F1 on average in three datasets 0.47) generally showed itself worse than Decision Tree (0.58), and Multilabel KNN (0.67) showed the best result.

In the future, the quality can be increased, for example, due to heuristics. On different types of data, different models work with different efficiencies. It makes sense to identify the appropriate patterns and select the most suitable models for specific data sets, based on the uniformity of the data, the volume of texts and other characteristics of the cases.

Even though the solution is initially aimed at working with Russian-speaking corpuses, it can be applied to other languages as well, in particular, to English.

Considering further improvement of quality, the product can be used in decision support systems or be integrated into knowledge bases. It seems promising to use it to mark-up texts of medical topics for highlighting the main concepts in them, which can be used to structure texts and further process them. One of the methods of application planned by the authors is to highlight keywords in the texts of clinical recommendations to further structure them for use in clinical decision support systems.

6 CONCLUSIONS

The problem of automatically assigning labels to texts is current, and its solution is in demand in many tasks of processing unstructured texts. The analysis of methods and existing software products for the text tagging task showed the lack of ready-made tools for processing Russian-language texts and the need to solve the text tagging problem with the final implementation in the form of software.

As a solution, it is proposed to consider text tagging as a problem of multilabel classification. A comparison of the known methods of solving the problem of multilabel classification was made, and, subsequently, the following methods with training were selected: ML-kNN, Decision Tree (ML-C4.5), and Random Forest. Additionally, TF-IDF method was included to extract features.

Based on the selected models, algorithms have been developed for automatically assigning labels to texts, and then implemented as the TextTagger software product.

The computational experiment showed a fairly high efficiency of the developed models for ML-kNN (metric F1-Measure 0.67) and the average for Decision Tree (0.58) and Random Forest (0.47), which indicates the possibility of their use in practice. Further quality improvement is possible by refining the process of normalizing texts and introducing heuristics to select the best possible model for a specific data set.

The developed software product is universal, applicable in various subject areas for processing texts in Russian, and applicable to other languages.

ACKNOWLEDGMENTS

The reported study was funded by RFBR according to the research projects No 19-07-00780, 19-07-00709.

REFERENCES

[1] D. Reinsel, J. Gantz, and J. Rydning, “The Digitization of the World – From Edge to Core,” IDC white paper, November 2018, Doc# US44413318.

[2] A. M. Nancy and R. Maheswari, "Review on unstructured data in medical data," Journal of Critical Reviews, 2020, pp. 2202-2208, doi: 10.31838/jcr.07.13.342.

[3] R. Mihalcea and P. Tarau, “TextRank: Bringing Order into Text,” EMNLP, 2004.

[4] S. Brin and L. Page, “The Anatomy of a Large-Scale Hypertextual Web Search Engine,” Comput. Networks, vol. 30, 1998, pp. 107-117.

[5] S. Tasci and T. Güngör, “LDA-based keyword selection in text categorization,” 24th International Symposium on Computer and Information Sciences, 2009, pp. 230-235.

[6] A. Sedova and O. Mitrofanova, “Topic Modelling of Russian Texts based on Lemmata and Lexical Constructions,” Saint-Petersburg State University, 2017.

[7] S. Rose, D. Engel, N. Cramer, and W. Cowley, “Automatic Keyword Extraction from Individual Documents,” 2010.

[8] M. Thushara, M. Krishnapriya, and S. N. Sangeetha, “A model for auto-tagging of research papers based on keyphrase extraction methods,” International Conference on Advances in Computing, Communications and Informatics, 2017, pp. 1695-1700.

[9] Dcipher Analytics official web-site [Online]. Available: http://www.dcipheranalytics.com.

[10] MonkeyLearn official web-site [Online]. Available: https://monkeylearn.com/.

[11] TwinWord official web-site [Online]. Available: https://www.twinword.com/.

[12] J. Read , B. Pfahringer, G. Holmes, and E. Frank, “Classifier chains for multi-label classification,” Machine Learning, vol. 85, 2011, pp. 333-359.

[13] S. Godbole and S. Sarawagi, “Discriminative Methods for Multi-labeled Classification,” PAKDD, 2004.

[14] G. Tsoumakas and I. Katakis, "Multi-Label Classification: An Overview," Int. J. Data Warehous, Min. 3, 2007, pp. 1-13.

[15] M. Zhang and Z. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," Pattern Recognit, vol. 40, 2007, pp. 2038-2048.

[16] A. Clare and R. King, "Knowledge Discovery in Multi-label Phenotype Data," PKDD, 2001.

[17] M. Zhang and Z. Zhou, "Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization," IEEE Transactions on Knowledge and Data Engineering, vol. 18, 2006, pp. 1338-1351.

[18] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: A comparison of retrieval performances," 2014.

[19] B. Trstenjak, S. Mikac, and D. Donko, "KNN with TF-IDF based Framework for Text Categorization," Procedia Engineering, vol. 69, 2014, pp. 1356-1364.

[20] TextTagger online demo [Online]. Available: https://texttagger.ishmukhamet.xyz.

[21] A. Luque, A. Carrasco, A. Martín, and A. D. L. Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," Pattern Recognit., vol 91, 2019, pp. 216-231.