DOI: 10.24412/2413-2527-2022-331-68-73

Features of Text Preprocessing for Performing Sentiment Analysis

N. E. Kosykh, I. A. Molodkin
Emperor Alexander I St. Petersburg
State Transport University
Saint Petersburg, Russia
nikitosagi@mail.ru, molodkin@pgups.ru

Grand PhD A. D. Khomonenko
Emperor Alexander I St. Petersburg
State Transport University,
Mozhaisky Military Space Academy
Saint Petersburg, Russia
khomon@mail.ru

Abstract. **The object of the research is the analysis of the sentiment of the Russian-language corpus of texts. The subject of the research is a comparison of the effectiveness of the approaches of preliminary text cleaning before sentiment analysis. The aim of the research is to develop a generalized method for preliminary data cleaning to create a neural network model. A distinctive feature of the proposed solutions is the use of modern and lightweight libraries for the possibility of preliminary preparation of a text for training with a neural network, and the hypothesis of using a truncated dictionary based on the assumption of data redundancy has been tested. The results obtained show the usefulness of the developed algorithm in terms of obtaining improved results in the learning process and indicate that, due to its versatility, it can be extrapolated for further use on other text data.**

Keywords: **mining, data analysis, sentiment analysis, neural networks, text processing.**

Introduction

Websites, social networks, micro-blogs are sources of a huge amount of user information for analysts, entrepreneurs and scientists trying to extract useful information from a large amount of user-generated text. Thanks to the feelings, emotions, images contained in the texts, one can draw certain conclusions to improve the quality of service or the quality of the services provided. For example, moderators of large Internet services and platforms can find solutions for timely tracking of violators by analyzing comments and user messages. For these reasons, a vast area of neural network technologies is aimed at developing systems for the automatic classification of texts (for example, aspect extraction, opinion analysis, sentiment analysis).

Recently, significant progress has been made in the development of various methods, starting with a rule base and machine learning to deep learning.

Despite the impressive results, the developed systems work exclusively with the specifically considered language, since the studied text corpora can have different dimensions, and also include non-linguistic elements that impede the analysis of data.

On the other hand, the nature of the content generated by users in social networks requires more careful preprocessing [1] before the stage of neural network analysis. In the field of tweet sentiment analysis, most scientific articles have used the approach of learning from scratch on corpuses consisting exclusively of thematic content, so that the models can eventually better understand local jargon with a special lexical and syntactic structure [2], as well as using abbreviations, punctuation marks and emoticons, etc. The use of such approaches imposes two restrictions: the first requires a large body of text for training in Russian, and the second — the need for significant resources and time to train models from scratch.

The approach described in this article allows testing the developed algorithm on a truncated corpus of texts and, if satisfactory results are obtained, use it when obtaining results on an integral corpus.

Vector transformation methods

The mathematical algorithms of future neural networks work with numbers, therefore, in order to use the mathematical apparatus for processing and analyzing text data, it is necessary first of all to transform words and sentences into numerical vectors, and preferably without losing semantic connection.

There are two main approaches to transforming a text document corpus [3] into a vector space:

1. Thematic modeling allows you to find statistical hidden patterns in the body of documents: latent semantic analysis (PLSA), latent Dirichlet distribution (LDA).

2. Distributive hypothesis — linguistic elements with similar distributions have similar meanings. This approach includes word embedding (WB) — a method that converts text tokens, most often represented by words, into dense small-sized vector representations trained on large unlabeled text corpora, where each token in the system is linked to another within the context. The Word2Vec model [4] proposed by Mikolov was an implementation of the word embedding method. The algorithm can encode text using two main approaches: skip-gramm or bag-of-word model [5]. The former predicts words in the surrounding context starting from the current word, while the latter predicts words based on the words surrounding it in the context. Also, the Global Vectors approach, is a distributed computing model, allowing you to code words faster on a large amount of data.

Ready-made vector representations obtained as a result of transformations allow you to compare texts and search for relationships, to perform classification and clustering of texts.

Natural language processing

Natural language processing is a branch of data science that deals with textual data that is used to analyze and solve business problems. Before using the collected data for the analysis and forecasting of user values, it is necessary to ensure their suitability, their preliminary processing is important [6].

The ultimate goal of processing data in natural language is to transfer this data to artificial intelligence (AI), which is capable of understanding human language. From a practical point of view, processing can be thought of as a set of methods and algorithms for extracting some useful information from text data.

Today, there is a fairly large range of practical tasks where natural language processing is required:

- machine translation of texts from foreign languages;
- automatic annotation of texts;
- classification of texts by categories (sentiment analysis, spam filter, rubrication);
- chat bots;
- recognition of given entities.

TEXT PRECLEANING PROCEDURE

Raw data obtained using the official API (http://developer.twitter.com/en/docs/api-reference-index) or taken from ready-made datasets contains heterogeneous information due to the nature of colloquial speech, for example: emoticons, hashtags, URLs, phone numbers, numbers without context, dates, etc. An example of some noisy messages is shown in Table 1.

Table 1

Example of background information

Интернет уже 5 день не работает :(
RT @supreme_fm: Хочу тату :(http://t.co/2LuWQRS5H3
Почему многим нравится что я пишу? Я же ничего крутого не записываю :(А тем временем бабушка оладушки жарит :3 но это уже другая история
Мне нравится готовить, но не нравится, что я часто режусь ножом или чистилкой для картошки :(#печальбеда
Объелась я золотистой жареной картошки и квашеной капусты с красным луком под маслом. Вкусно, но зачем так много ;(
Я скоро уничтожу свой телефон полностью, опять его в лужу грохнула (

This work uses a set of procedures to transform chaotic information into a more traditional and understandable form. The sequence of actions for each procedure does not depend on specific language features [7]. Actions are usually based on linguistic rules contained in conversion tables or regular expressions.

The data must be preprocessed to perform mining tasks. First, you need to perform preprocessing of the text, which includes:

1. Removing punctuation marks and specials characters. Punctuation marks and other special characters should be removed to eliminate inaccuracies in determining polarity. In some cases, signs can stick together with words and be incorrectly identified in dictionaries, which makes them inaccessible for analysis.

2. Removing URL's. Typically, URLs are not used to analyze sentiment in informal texts. For example, consider the following sentence «I went to the site funny-jokes.rf, because I'm bored», which is negative, however, due to the presence of the word «funny» it can become neutral, and this is a false-positive result. To avoid this kind of collision, you need to remove the URLs beforehand.

3. Removing stop words: conjunctions, interjections. Words like what, like, like, etc. will not affect the polarity of expressions; to reduce the computational complexity, it is worth getting rid of them. Python contains stoplists for different languages in the NLTK module.

4. Removing forwarded references. Such references are usually marked with the name of the original author plus the letter combination RT. In a thematic analysis, additional information for statistics can be obtained from this data, but in sentiment analysis it is redundant (Fig. 1).

5. Breakdown of offers into tokens. Typically, this is the process of separating individual words into an array, separated by punctuation marks and whitespace.

6. Normalization of words. Normalization is the process of bringing words to their standard morphological form, which is implemented by the stemming method — the method of eliminating the endings of words, or lemmatization — bringing a word to its normal (dictionary) form («is» — «to be», «written» — «to write») [8].

7. Removing number sequences. The use of numbers in the analysis can increase the size of the studied vocabulary, which will complicate the process of training the model on unnecessary data.

To solve the problem, a universal pipeline (method) was developed, which can be used for preliminary text cleaning in any of the projects under study. It includes all of the above stages of processing textual data written in the form of regular expressions, as well as splitting into tokens and applying lemmatization or stemming methods to choose from (Fig. 1).

```
def preprocess_text(text):
    text = text.lower().replace("ё", "е")
    text = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', 'url', text)
    text = re.sub('@[^\s]+', '', text)
    text = re.sub('(?:(?:\d+,?)+(?:\.?\d+)?)',' ',text)
    text = re.sub('[^a-zA-Za-яА-Я1-9]+', ' ', text)
    text = re.sub(' +', ' ', text)
    text = re.sub('[a-zA-Z]+', '', text)
    text = re.sub(" \d+", " ", text)

    tokens = []
    for token in text.split():
        if token and token not in stop:
            token = token.strip()
            token = stemmer.stem(token)
            token = morph.normal_forms(token)[0]
            tokens.append(token)

    text = ' '.join(tokens)

    return text.strip()
```

Fig. 1. Data preprocessing pipeline

It is also worth noting that the processing time by truncating the endings is several times faster than searching for a dictionary form. If you sacrifice quality for speed you can opt for the Snowball algorithm otherwise it is better to use a lemmatizer.

COMPARISON OF TEXT NORMALIZATION ALGORITHMS

There are two types of stemmers in the NLTK library — Porter's stemmer and Snowball [9], the second, in turn, is an improved version of the first, and we will test it on an array of sentences. After processing as a basis, we get a lot of non-dictionary words. Unlike the previous approach, lemmatization shortens a word to its dictionary form and in most cases is semantically complete (Table 2).

Table 2

Comparison of text processing by stemming and lemmatization

Stemming (word processing)	Lemmatization (word processing)
'получа осторожн котлетк перекладыва др посуд'	['получаться осторожно котлетка перекладывать др посуда'
'единствен расстраива бал невозможн сход кинотеатр друз интересн фильм'	'единственный расстраивать бали невозможность сходить кинотеатр друг интересный фильм'
'базарн сегодн спрашива суш парен вес кг сказа пор лол'	'базарнов сегодня спрашивать сушиться парень весить кг сказать пора лола'
'маловат получа незлобин'	'маловатый получаться незлобин',
'любл всем тво истерик солнышк'	'любить весь твой истерик солнышко']

The next section displays experimental data run through the NLTK Snowball and MorphAnalyzer modules. The strengths and weaknesses of each approach are identified.

Vector text representation

In most mining tasks, after the text cleanup stage, we create a dictionary of index mappings, so that frequently occurring words are assigned a lower index, and then the word appears less frequently in the text corpus. One trivial solution would be to use a combination of the split () and strip () methods, splitting the input data stream into individual words and writing them to an array. Then, using the Tokenize module, we encode a random array of words into a vector, where the total TF-IDF coefficient is calculated for each word, and then we update the dictionary based on the list of received tokens by calling the fit_on_texts () method.

Results of statistical tests of algorithms for one dataset

Additionally, let us analyze the length of the records loaded into the array after the stage of splitting into tokens (Fig. 2). The figure shows a histogram that shows the distribution between the number of records and their corresponding length (by words).

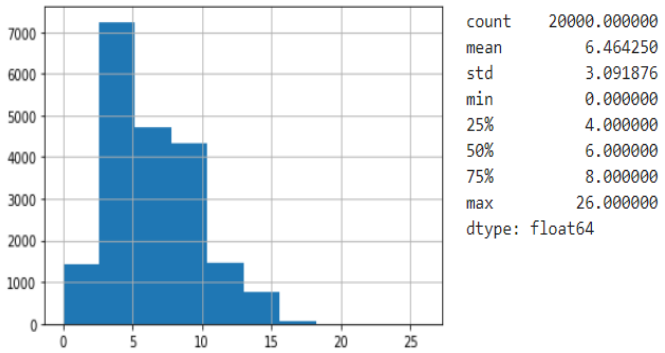

Fig. 2. Sampling before deleting unnecessary records

The average length of a message in the corpus is 6 words. A number of records have zero length. The presence of these values in the data array will not have any effect on the learning process of the model [10], it will only become redundant information. We also make a hypothesis that in the presence of sentences from 1 token there is a possibility of incorrect interpretation of the polarity, it follows that at the stage of preliminary data preparation, a check for the length of the loaded records should be added.

Based on the condition specified below, we check the initial data set and write into the new array only those sentences that, after the preprocessing stage, have more than two words:

```
sentences1 = []
labels1 = []
for id,item in enumerate(sentences):
  if len(item.split()) >= 2:
     sentences1.append(item)
     labels1.append(labels[id])

assert len(sentences1) == len(labels1)
```

After preparing a new dataset, let's build a histogram (Fig. 3) to check the condition's performance.

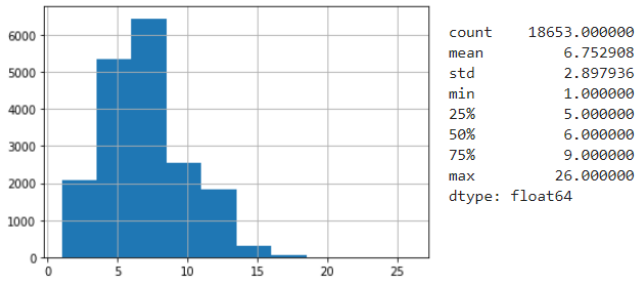

Fig. 3. Fetch after deleting unnecessary records

As you can see from the graph, words and their corresponding labels were removed, where there were no tokens and tokens less than three. Within a small number of test records, this may seem like an unnecessary step, but on a large body of texts, this can lead to dirty statistics. A sample of 20 000 records was taken as a basis, after deleting records that do not meet the condition, 18 653 remained. It is necessary to establish whether there is a difference between a more extensive dictionary or a better quality.

The next step involves building a neural network model based on long-term neural memory LSTM [11]. Its advantages include fast scanning and acceptable learning speed. However, it is not suitable for repeated training on different parameters due to its peculiarity to remember the states in past sessions. As a temporary solution to reset the weights, it is necessary to compile the trainable model each time.

The results of cross-validation in terms of accuracy for models trained using different text processing techniques are shown in Table 3. In the first case, the source text was divided into tokens, each of which was reduced to a stem or to a dictionary form, depending on the user's choice. These tokens became the vocabulary for future network learning. In another case, all the same stages except for those *n*-grams that contained less than 3 words.

Table 3

Results of cross-validation in training models

Data set 20000 words	Epoch No. 6	Epoch No. 7	Epoch No. 8	Epoch No. 9	Epoch No. 10
***n*-gramma >=0+lemmatization**	0.7601	0.7962	0.8192	0.8369	0.8569
***n*-gramma >=0+stemming**	0.7734	0.8035	0.8215	0.8443	0.8582
***n*-gramma >2+lemmatization**	0.7852	0.8097	0.8278	0.8497	0.8638
***n*-gramma >2+stemming**	0.7844	0.7844	0.8309	0.8480	0.8566

By the tenth epoch of learning, an approach that includes skipping *n*-grams (where $n <= 2$) and lemmatization at the preprocessing stage gives an accuracy advantage over the rest of the techniques under consideration.

Conclusion

Our studies have shown that training a neural network on a training dataset containing sentences longer than 2 tokens and processed by the lemmatization method gives a small increase in the training accuracy at higher epochs compared to the original (unprocessed) dataset, and also has a more intensively decreasing function losses (Fig. 4).

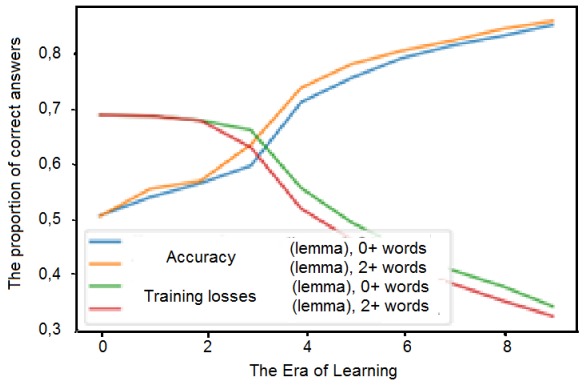

Fig. 4. Data processing + lemmatization

This difference is due to a more accurate distribution of weight coefficients [12] between neurons in the process of training the network. If we compare the use of an abbreviated or full vocabulary on stemming, then the difference becomes insignificant towards the achievement of the later eras of learning (Fig. 5).

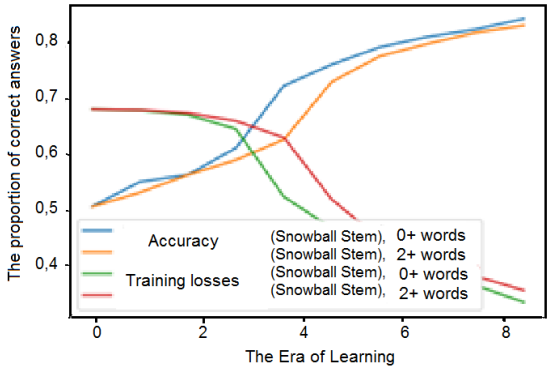

Fig. 5. Data processing + stemming

It is proposed to further study natural language processing algorithms on an existing dataset with subsequent *n*-gram network training, as well as improve the preliminary text cleaning algorithm for the following: automatic replacement of toxic words or selection of appropriate synonyms, replacement of double letters with single letters, replacement of jargon words with commonly used ones, as well as encoding emoticons and converting them to a common number format to improve accuracy and reduce losses in the process of training the model.

References

1. Gan N. Application of Algorithms for Natural Language Processing in IT-Monitoring with Python Libraries, *Towards Data Science*. Published online at 14 August 2021. Available at: http://towardsdatascience.com/application-of-algorithms-for-natural-language-processing-in-it-monitoring-with-python-libraries-57eddc45c4ac (accessed 04 Nov 2021).

2. Hemalatha I., Varma G. P. S., Govardhan A. Preprocessing the Informal Text for Efficient Sentiment Analysis, *International Journal of Emerging Trends and Technology in Computer Science*, 2012, Vol. 1, Is. 2, Pp. 58–61.

3. Lychenko N. M., Sorokovaja A. V. Sravnenie effektivnosti metodov vektornogo predstavleniya slov dlya opredeleniya tonalnosti tekstov [Comparison of Effectiveness of Word Representations Methods in Vector Space for the Text Sentiment Analysis], *Matematicheskie struktury i modelirovanie [Mathematical Structures and Modeling]*, 2019, No. 4 (52), Pp. 97–110. DOI: 10.24147/2222-8772.2019.4.97-110. (In Russian)

4. Church K. W. Word2Vec, *Natural Language Engineering*, 2017, Vol. 23, Is. 1, Pp. 155–162. DOI: 10.1017/S1351324916000334.

5. Paramonov I. Yu., Smagin V. A., Kosykh N. E., Khomonenko A. D. Metody i modeli issledovaniya slozhnykh sistem i obrabotki bolshikh dannykh: Monografiya [Methods and models for studying of complex systems and big data processing: Monograph]. Saint Petersburg, LAN Publishing House, 2020, 236 p. (In Russian)

6. Krivaltsevich E. V. Obrabotka estestvennogo yazyka (Natural Language Processing) pri ispolzovanii tekhnologii NLTK (Natural Language Toolkit) na baze yazyka programmirovaniya Python [Natural Language Processing using NLTK (Natural Language Toolkit) technology based on the Python programming language], *Ideas. Search. Decisions: Collection of Articles of the VII International Scientific and Practical Conference, Minsk, Belarus, November 25, 2014*. Minsk, Belarusian State University, 2015, Pp. 41–51. (In Russian)

7. Predvaritelnaya obrabotka dannykh v Python [Data preprocessing in Python], Machinelearningmastery.ru — Mashinnoe obuchenie, neyronnye seti, iskusstvennyy intellect [Machinelearningmastery.ru — Machine Learning, Neural Networks, Artificial Intelligence]. Published online at August 23, 2019. URL: http://www.machinelearningmastery.ru/data-preprocessing-in-python-b52b652e37d5 (accessed 05 Nov 2021). (In Russian)

8. Gupta I., Joshi N. Tweet Normalization: A Knowledge Based Approach, *Proceedings of the 2017 IEEE International Conference on Infocom Technologies and Unmanned Systems (ICTUS) (Trends and Future Directions), Dubai, United Arab Emirates, December 18–20, 2017*. Institute of Electrical and Electronics Engineers, 2017, Pp. 157–162. DOI: 10.1109/ICTUS.2017.8285996.

9. Malik U. Python for NLP: Tokenization, Stemming, and Lemmatization with SpaCy Library, *Stack Abuse*. Available at: http://stackabuse.com/python-for-nlp-tokenization-stemming-and-lemmatization-with-spacy-library (accessed 10 Nov 2021).

10. Morrissey M., Wasser L., Diaz J., Palomino J. Analyze the Sentiment of Tweets from Twitter Data and Tweepy in Python, Earth Data Science — Earth Lab. Last update 11 September 2020. Available at: http://www.earthdatascience.org/courses/use-data-open-source-python/intro-to-apis/analyze-tweet-sentiment-in-python (accessed 29 Nov 2021).

11. Huang Z., Hu W., Yu K. Bidirectional LSTM-CRF Models for Sequence Tagging, *arXiv*, 2015, Vol. 1508.01991, 10 p. DOI: 10.48550/arXiv.1508.01991.

12. Korobov V. B. Sravnitelnyy analiz metodov opredeleniya vesovykh koeffitsientov «vliyayushchikh faktorov» [Comparative Analysis of Methods for Determining the Weight Coefficients of «Influencing Factors»], *Sotsiologiya: metodologiya, metody, matematicheskie modeli (Sotsiologiya: 4M) [Sociology: Methodology, Methods, Mathematical Modeling (Sociology: 4M)]*, 2005, No. 20, Pp. 54–73.

DOI: 10.24412/2413-2527-2022-331-68-73

Особенности предварительной обработки текстовых данных при анализе тональности текстов

Н. Е. Косых, И. А. Молодкин
Петербургский государственный университет
путей сообщения Императора Александра I
Санкт-Петербург, Россия
nikitosagi@mail.ru, molodkin@pgups.ru

д.т.н. А. Д. Хомоненко
Петербургский государственный университет
путей сообщения Императора Александра I,
Военно-космическая академия имени А. Ф. Можайского
Санкт-Петербург, Россия
khomon@mail.ru

Аннотация. **Объект исследования — анализ тональности русскоязычного корпуса текстов. Предмет исследования — сравнение эффективности подходов предварительной очистки текста перед анализом тональности. Цель исследования — разработка обобщенного метода предварительной очистки данных для создания модели нейросети. Отличительной чертой предложенных решений является использование современных и легковесных библиотек для возможности предварительной подготовке текста к обучению нейросетью; также апробирована гипотеза использования усеченного словаря на основе предположения об избыточности данных. Полученные результаты показывают полезность разработанного алгоритма с точки зрения получения улучшенных результатов в процессе обучения и указывают на то, что благодаря своей универсальности он может быть экстраполирован для дальнейшего использования на других текстовых данных.**

Ключевые слова: **интеллектуальный анализ, анализ данных, сентиментный анализ, нейронные сети, обработка текста.**

Литература

1. Gan, N. Application of Algorithms for Natural Language Processing in IT-Monitoring with Python Libraries // Towards Data Science. — 2021. — 14 August. URL: http://towardsdatascience.com/application-of-algorithms-for-natural-language-processing-in-it-monitoring-with-python-libraries-57eddc45c4ac (дата обращения 04.11.2021).

2. Hemalatha, I. Preprocessing the Informal Text for Efficient Sentiment Analysis / I. Hemalatha, G. P. S. Varma, A. Govardhan // International Journal of Emerging Trends and Technology in Computer Science. 2012. Vol. 1, Is. 2. Pp. 58–61.

3. Лыченко, Н. М. Сравнение эффективности методов векторного представления слов для определения тональности текстов / Н. М. Лыченко, А. В. Сороковая // Математические структуры и моделирование. 2019. № 4 (52). С. 97–110. DOI: 10.24147/2222-8772.2019.4.97-110.

4. Church, K. W. Word2Vec // Natural Language Engineering. 2017. Vol. 23, Is. 1. Pp. 155–162. DOI: 10.1017/S1351324916000334.

5. Методы и модели исследования сложных систем и обработки больших данных: Монография / И. Ю. Парамонов, В. А. Смагин, Н. Е. Косых, А. Д. Хомоненко; под ред. В. А. Смагина, А. Д. Хомоненко. — Санкт-Петербург: Лань, 2020. — 236 с. — (Учебники для вызов. Специальная литература).

6. Кривальцевич, Е. В. Обработка естественного языка (Natural Language Processing) при использовании технологии NLTK (Natural Language Toolkit) на базе языка программирования Python // Идеи. Поиски. Решения: Сборник статей VII Международной научно-практической конференции (Минск, Беларусь, 25 ноября 2014 г.). — Минск: Белорусский гос. ун-т, 2015. — С. 41–51.

7. Предварительная обработка данных в Python // Machinelearningmastery.ru — Машинное обучение, нейронные сети, искусственный интеллект. — 2019. — 23 августа. URL: http://www.machinelearningmastery.ru/data-preprocessing-in-python-b52b652e37d5 (дата обращения 05.11.2021).

8. Gupta, I. Tweet Normalization: A Knowledge Based Approach / I. Gupta, N. Joshi // Proceedings of the 2017 International Conference on Infocom Technologies and Unmanned Systems (ICTUS) (Trends and Future Directions), (Dubai, United Arab Emirates, 18–20 December 2017). — Institute of Electrical and Electronics Engineers, 2017. — Pp. 157–162. DOI: 10.1109/ICTUS.2017.8285996.

9. Malik, U. Python for NLP: Tokenization, Stemming, and Lemmatization with SpaCy Library // Stack Abuse. URL: http://stackabuse.com/python-for-nlp-tokenization-stemming-and-lemmatization-with-spacy-library (дата обращения 10.11.2021).

10. Analyze the Sentiment of Tweets from Twitter Data and Tweepy in Python / M. Morrissey, L. Wasser, J. Diaz, J. Palomino // Earth Data Science — Earth Lab. Last update 11 September 2020. URL: http://www.earthdatascience.org/courses/use-data-open-source-python/intro-to-apis/analyze-tweet-sentiment-in-python (дата обращения: 29.11.2021).

11. Huang, Z. Bidirectional LSTM-CRF Models for Sequence Tagging / Z. Huang, W. Hu, K. Yu // arXiv. 2015. Vol. 1508.01991. 10 p. DOI: 10.48550/arXiv.1508.01991.

12. Коробов, В. Б. Сравнительный анализ методов определения весовых коэффициентов «влияющих факторов» // Социология: методология, методы, математические модели (Социология: 4М). 2005. № 20. С. 54–73.