Sentiment Analysis of Short Russian Texts Using BERT and Word2Vec Embeddings

Ekaterina Popova [1] and Vladimir Spitsyn [1]

[1] *National Research Tomsk Polytechnic University, 30, Lenin Avenue, Tomsk, 634050, Russia*

Abstract

This article is devoted to modern approaches for sentiment analysis of short Russian texts from social networks using deep neural networks. Sentiment analysis is the process of detecting, extracting, and classifying opinions, sentiments, and attitudes concerning different topics expressed in texts. The importance of this topic is linked to the growth and popularity of social networks, online recommendation services, news portals, and blogs, all of which contain a significant number of people's opinions on a variety of topics. In this paper, we propose machine-learning techniques with BERT and Word2Vec embeddings for tweets sentiment analysis. Two approaches were explored: (a) a method, of word embeddings extraction and using the DNN classifier; (b) refinement of the pre-trained BERT model. As a result, the fine-tuning BERT outperformed the functional method to solving the problem.

Keywords

Natural language processing, sentiment analysis, deep learning, BERT, CNN, LSTM

1. Introduction

Sentiment analysis is a technique by which one can analyze a piece of text to determine the sentiment behind it. It combines machine learning and natural language processing (NLP) to achieve this. Using basic Sentiment analysis, a program can understand if the sentiment behind a piece of text is positive, negative, or neutral.

The relevance of this topic is associated with the development and growing popularity of social networks, online recommendation services, news portals and blogs, where a large number of people's opinions on various issues are collected.

There are three major types of algorithms used in sentiment analysis:

- Rule-based systems automatically perform sentiment analysis based on a set of manually crafted rules. This approach to sentiment analysis involves searching for keywords in the text and matching each of them with a numeric value in a dictionary or associative array.
- Automatic methods, contrary to rule-based systems, do not rely on manually crafted rules, but on machine learning techniques. A sentiment analysis task is usually modeled as a classification problem, whereby a classifier is fed a text and returns a category, e.g. positive, negative, or neutral.
- Hybrid systems combine the desirable elements of rule-based and automatic techniques into one system. One huge benefit of these systems is that results are often more accurate.

The aim of this work is to study modern neural network approaches to automatically determining the sentiment of short texts in Russian using the example of data from the social network Twitter.

GraphiCon 2021: 31st International Conference on Computer Graphics and Vision, September 27-30, 2021, Nizhny Novgorod, Russia
EMAIL: esp9@tpu.ru (E. Popova); spvg@tpu.ru (S. Spitsyn)
ORCID: 0000-0003-1459-7955 (E. Popova); 0000-0001-5978-1321 (S. Spitsyn)

2. Data set

Data analysis from social networks is one of the popular research areas in the field of sentiment analysis. Social media contain huge amount of the sentiment information in the form of tweets, blogs, and updates on the status, posts, etc. Therefore, in this work, messages from the social network Twitter will be used as data.

Twitter is one of the most popular social media platforms in the world, with 330 million monthly active users and 500 million tweets sent each day. By carefully analyzing the sentiment of these tweets – whether they are positive, negative, or neutral, for example – we can learn a lot about how people feel about certain topics.

However, the task is complicated by the fact that Twitter messages contain a large number of slang words and misspellings and repeated characters. Maximum length of each tweet in Twitter is 140 characters. Therefore, it is very important to identify correct sentiment of each word.

The corpus of short texts by Yulia Rubtsova (RuTweetCorp) [1], formed based on Russian-language messages from the social network Twitter, was chosen as a training sample this work. The corpus was automatically labeled based on emoticons and contains 114,991 positive and 111,923 negative messages.

The peculiarity of the dataset is that RuTweetCorp was initially designed for the creation of a sentiment lexicon, not for sentiment classification. The dataset was collected automatically, i.e. each text was associated with the sentiment class based on the emoticons it contained. Therefore, even a simple rule-based approach is able to demonstrate good results. So that, the authors recommend removing emoticons from the dataset at the preprocessing stage in order to solve the problem of automatic sentiment analysis. Thus, when analyzing the literature, only those articles were taken into account in which the preprocessing procedure included the removal of emoticons.

In paper [7] compared logistic regression, XGBoost classifier and Convolutional Neural Network on RuTweetCorp for binary classification and achieved F1 = 78.1% using a convolutional neural network. The authors of [8] tested the possibilities of fine-tuning Multilingual BERTBase, RuBERT and two versions of Multilingual USE for RuTweetCorp. The best results in binary classification were F1 = 83.69% for RuBERT.

3. Text pre-prossessing

The text classification results depend on the input text preprocessing quality. In this work all the text were reduced to lower case, punctuation marks were removed, links and usernames were replaced with "URL" and "USER ", respectively, and the word "RT", denoting retweet, is also removed at the preprocessing stage

4. Word embedding

The main idea of word embedding is to match each word with some numerical vector of fixed dimension. Vectors are constructed in such a way that words found in similar contexts have similar vector representations. Word embeddings are mostly used as input features for other models built for custom tasks.

An important advantage of word embedding is that no tagged data is required to train them. Attachments are retrieved from a very large unmarked enclosure. Pretrained word embeddings are available online and can be used by researchers around the world for a variety of NLP tasks.

There are two main approaches to generate word embeddings:

- Context-independent (Bag of Words, TF-IDF, Word2Vec, GloVe);
- Context-aware (ELMo, Transformer, BERT, Transformer-XL).

In this work, word embeddings based on Word2Vec and BERT will be investigated:

- Word2Vec models generate embeddings that are context-independent. There is just one vector (numeric) representation for each word. Different senses of the word (if any) are combined into one single vector. Word2Vec embeddings do not take into account the word position in the sentence.

- BERT model generates embeddings that allow us to have multiple (more than one) vector (numeric) representations for the same word, based on the context in which the word is used. BERT model explicitly takes as input the position (index) of each word in the sentence before calculating its embedding. Thus, BERT embeddings are context-dependent.

4.1. Word2Vec embedding

Word2Vec is a set of algorithms for calculating vector representations of words, developed by a group of researchers from Google in 2013 [2]. Word 2Vec implements two main architectures CBOW (Continuous Bag of Words) and Skip-gram. CBOW predicts a word based on the current context, and skip-gram, on the contrary, predicts a context based on the current word.

In this work, the Skip-gram architecture was chosen, which is more suitable for small corpora of texts (less than one hundred million words), since it better takes into account rare words [3].

To train the word2vec model an implementation from the Gensim library was chosen. The model was trained with the following hyperparameters:

- Dimension of vector space – 200;
- Scanning window size – 10.

The model was trained on 17.5 million unlabeled Russian-language Twitter posts. The size of the formed dictionary was 712,991 words. Figure 1 shows the visualization of clusters of similar words of the trained Word2Vec model.

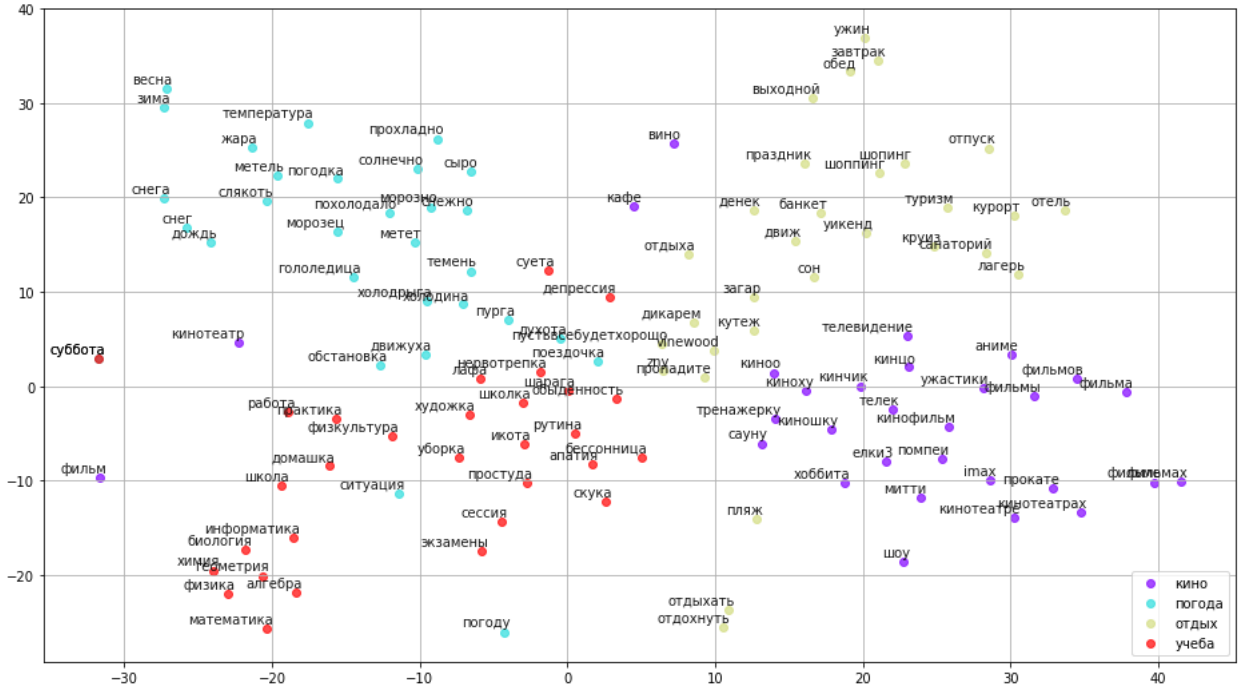

Figure 1: Visualization of clusters of similar words of the trained Word2Vec model

4.2. BERT embedding

BERT (Bidirectional Encoder Representations from Transformers) is a neural network developed by Google researchers in 2018 [4], which has shown high results on a number of NLP problems. BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks. The BERT model can be used in two ways:

- To generate word attachments, which are further used as input for DNN classifiers;

- To fine-tune a pretrained BERT model for a specific task.

In this work, we use the BERT model for the Russian language Conversational RuBERT from DeepPavlov [5]. The model has 12 stacked transformer encoder layers, with 12 attention heads. The embedding dimension is 768, 180M parameters.

5. Feature-based approaches

For feature-based approaches, were used pre-trained Word2Vec and BERT models to obtain the sequence of embeddings for a given messages:

- **Word2Vec model:** Were used pre-trained Word2Vec model and apply this model to generate one embedding for each word of comment.
- **BERT model:** Word-piece tokenization is performed on the comment and then used as input to a pre-trained BERT model. BERT model provides contextual embedding for the word-pieces [11].

The obtained embeddings from both Word2Vec and BERT models are then used as input to a DNN classifier. Deep learning models (CNN, LSTM, GRU) are used as DNN classifier:

- **CNN** (Convolutional Neural Network) has traditionally been used in image processing applications, but has recently become actively applied to various NLP tasks. For example, the article [6] demonstrates the effective use of CNN for natural language processing on various test data.
- **LSTM** (Long short-term memory) is a special kind of recurrent neural network capable of handling long-term dependencies. LSTM is an advanced RNN [9], a sequential network that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN.
- **GRU** (Gated recurrent units) are a gating mechanism in recurrent neural networks, introduced in 2014. The GRU [10] is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate.

The configurations of the DNN models that were used in this work are presented below:

- **LSTM and GRU:** One LSTM (GRU) layer with 128 internal nodes followed by an output layer containing one neuron for predicting values. It uses a sigmoid activation function to produce a probabilistic result ranging from 0 to 1, which can be converted to a clear class value.
- **CNN (Conv1D):** Two CNN layers, with 128 and 64 filters respectively, and the width of the window for filters is 5, activation function is ReLU, window step size (stride) is 1. The output layer contains one neuron with a sigmoid activation function.

We use a varying dropout up to 0.2. The models are trained using Adam optimizer with learning rate of 0.001.

6. BERT fine-tuning

The BERT pre-trained model can be fine-tuned to a specific task. This consists in the adapting of the pre-trained BERT model parameters to a specific task using a small corpus of task specific data. For the purpose of classification task, a neural network layer is used on top of fine-tuned BERT model. So, the weights of this layer and the weights of the other layers of the Bert model are trained and fine-tuned correspondingly using task specific data in order to perform the classification task.

For BERT fine-tuning we used Adam optimizer, with an initial learning rate of 2e–5, batch size of 32 and 3 epochs of training.

7. Results and discussion

To assess the quality of the obtained classification results, generally accepted metrics were used: Recall, Precision, F – measure, Accuracy. The following auxiliary parameters were calculated:

- TP is the number of true positive results;

- TN is the number of true negative results;
- FP is the number of false positive results;
- FN is the number of false negative results.

Precision can be interpreted as the proportion of objects called positive by the classifier and at the same time really positive, and recall shows what proportion of objects of a positive class from all objects of a positive class the algorithm found. The higher the recall and precision values, the better the classification result will be.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

While recall and precision are very important metrics, they will not tell the whole story on their own.

One way to summarize them is *F* – measure, a measure that is the harmonic mean of precision and completeness:

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

This version of calculating the *F* – measure is also known as the *F1* – measure. Since the *F1* – measure takes into account precision and completeness, it may be a more appropriate metric for binary classification of unbalanced data.

Accuracy is the proportion of right classified objects in the all classified objects:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \quad (4)$$

The results of classifiers efficiency evaluation are presented in the Table 1. As we see, approaches base on BERT fine-tuning show a significant gap in *F1* compared to the feature-based approaches.

Table 1
Results of the experiments

Model	Accuracy	Precision	Recall	F1-measure
Word2Vec + CNN	75,37	75,39	75,34	75,35
Word2Vec + LSTM	78,01	78,49	78,08	77,94
Word2Vec + GRU	78,58	78,66	78,61	78,57
Conversational RuBERT + LSTM	78,12	78,13	78,10	78,11
Conversational RuBERT + GRU	78,98	78,97	78,98	78,98
Conversational RuBERT Fine-Tuning	82,22	82,25	82,19	82,61

8. Conclusion

The article compares two approaches to automatically determining the sentiment of messages on social networks. These approaches are based on deep learning classifiers and word embeddings.

The combination of feature-based approaches and fine-tuning of pre-trained BERT model were proposed. In feature-based approaches, Word2Vec and BERT embeddings were used as input features to GRU and LSTM classifiers. Further, we have compared these configurations with fine-tuning of pre-trained BERT model.

As a result, BERT fine-tuning turned out to be more efficient than the functional approach to solve the problem. In the future, it is planned to explore the possibility of combining Word2Vec and BERT attachments.

9. Acknowledgements

The reported study was funded by the Russian Foundation for Basics Research under RFBR research project No. 18-08-00977 A and supported by Tomsk Polytechnic University Competitive-ness Enhancement Program.

10. Reference

[1] Y. Rubtsova, Constructing a corpus for sentiment classification training, Softw. Syst. 109 (2015) 72–78.
[2] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient Estimation of Word Representations in Vector Space, arXiv:1301.3781 (2013).
[3] X. Rong, word2vec Parameter Learning Explained, arXiv:1411.2738 (2014).
[4] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.
[5] DeepPavlov's documentation: BERT in DeepPavlov, 2021. URL: http://docs.deeppavlov.ai/en/master/features/models/bert.html.
[6] Y. Kim, Convolutional Neural Networks for Sentence Classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751.
[7] A. Zvonarev, A. Bilyi, A comparison of machine learning methods of sentiment analysis based on russian language twitter data, in: The 11th Majorov International Conference on Software Engineering and Computer Systems (2019).
[8] S. Smetanin, M. Komarov, Deep transfer learning baselines for sentiment analysis in Russian, Information Processing & Management 58(3) (2021).
[9] Understanding LSTM Networks, 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs.
[10] Illustrated Guide to LSTM's and GRU's: A step by step explanation, 2018. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-stepexplanation-44e9eb85bf21
[11] BERT Word Embeddings Tutorial, 2019. URL: https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial.