Sentiment Analysis of Short Texts from Social Networks Using Sentiment Lexicons and Blending of Machine Learning Algorithms

Anastasia Novikova[1,2] ✉ and Sergey Stupnikov[1,2,3]

[1] Mail.Ru Group, Moscow, Russia,
[2] Lomonosov Moscow State University, Moscow, Russia,
[3] Institute of Informatics Problems, Federal Research Center "Computer Science and Control"of the Russian Academy of Sciences, Moscow, Russia,
novikovaa@abc.math.msu.su sstupnikov@ipiran.ru

Abstract. This paper focuses on the field of sentiment analysis of natural language texts. To be precise, short texts extracted from social networks are analyzed. At present two groups of applied methods can be distinguished in this field: machine learning methods and methods based on sentiment lexicons. The paper reviews the principal methods in this field and proposes an approach for short text sentiment analysis, combining sentiment lexicons and blending of machine learning algorithms for the problem of three-class text classification. Various formulas for determining weights of the words in vector representation of texts are considered as well. This approach is applied to a dataset that consists of 10,000 manually marked posts extracted from VKontakte social network. Class distribution of dataset objects for the studied area often tends to be unbalanced. Standard models show moderate quality but only due to the fact that the majority of data points are classified as belonging to the dominant class. In this work an approach that shows good metric scores for all the three classes is described.

Keywords: Sentiment analysis, Sentiment lexicons, Blending, Machine learning, TF-IDF

1 Introduction

The amount of data collected from users around the world has grown greatly over the past decade as websites such as Twitter, Amazon and Facebook have facilitated publication and aggregation of micro opinion pieces that allow individuals to record their sentiments towards things, people and events.

Sentiment analysis is the process of determining whether a text relates to one of given classes. It is also known as opinion mining and aims to determine the attitude of a writer with respect to some topic or the overall contextual polarity or emotional reaction to a document, or event, or some other object. The information received after the analysis is clearly of value to researchers, organizations and companies trying to understand sentiment both for individuals and

CEUR-WS.org/Vol-2268/paper21.pdf

on average, as well as to identify trends [2].

At present two groups of applied methods can be distinguished in the field: machine learning methods and methods based on sentiment lexicons. The aim of this work is to develop a method combining the two mentioned approaches as well as a meta ensemble of machine learning models that would cope with the mentioned problem of strong class imbalance.

Blending of *Logistic Regression*, *Random Forest Classifier*, *Gradient Boosting Classifier*, and set of features is used to combine the information received from all the parts of the ensemble. Among added features there are metrics received from using emojis extracted from texts, sentiment lexicons, application of *Relevant Frequency* method [17]. Various TF-IDF formulas are examined [3] and best variants for the given task are chosen.

The developed method exceeds quality of standard classifiers on given dataset and shows high value of $F1$ metric even with a strong class imbalance.

The paper is organized as follows. In section 2 related work is overviewed, then the proposed approach is described. In section 4 conclusion is provided.

2 Related Work

Most commonly applied methods in text classification and sentiment analysis tasks in particular can be divided into two groups: methods that use sentiment lexicons [6][7][8] and machine learning methods[16][10][4][9][11]. In some works combination of these approaches is suggested[5].

Lexicon based techniques rely on an assumption that the collective polarity of a document or sentence is the sum of polarities of the individual words or phrases. In [6] modifying words are used together with sentiment lexicons. As word or words written before a certain word or phrase can influence its sentiment polarity there is an idea to change a weight of the word according to its context. In [7] sentiment lexicon is created manually with application of Relevant Frequency method. The authors of [8] use hashtags for detecting sentiment polarity of tweets.

For machine learning methods texts are usually preprocessed by removing stop words and irrelevant information[11][5][10]. Lemmatization or stemming, spell correction for words is applied in some cases [11][5]. Vector space model is often used for representation of texts [5][9][4]. Classifiers are then applied over vector representations of texts to make predictions after being fit on train dataset [11][5][9]. In [5] the authors suggest methods for extracting key words from texts using Relevant Frequency method for classification, and a combination of this method with SVM classifier. Relevant Frequency method is also used for adding words to lexicons in this paper. In some works the information about parts of speech of words is used for text classification [10][11]. However there is still conflict whether Parts-of-Speech are a useful feature for sentiment classification or not. Some researchers argue in favour of good POS features [13] while others do not recommend to use them for this purpose [14]. In [11] and [12] an ensemble of algorithms is used for classification and rule based classifiers

are applied. The authors in [6] use D-gramms together with sentiment lexicons for text classification. They suggest adding words to dictionaries using Relevant Frequency method and to use modifier words to change words weighs.

Neural networks are often applied to text classification tasks and for sentiment analysis in particular. CNN or RNN are used very often. In [16] authors conduct comparison of neural networks architectures used in tasks related to sentiment analysis for Russian texts. The use of neural networks is not applicable for our dataset in straight-forward way because these methods require much larger amount of labeled data for training.

Sentiment analysis is a growing area of research. There have been a lot of related papers at conferences of different levels for some years and new works still appear very often.

3 The Proposed Approach

3.1 Dataset

The dataset for three-class sentiment classification was provided by Mail.Ru Group. It was retrieved from *VKontakte* social network in July, 2017 and consists of about 10,000 manually marked posts. Four people took part in the marking process and each post was assessed by one person. The quality of the classification was evaluated by leading assessor.

Posts from social networks have several specific features. They are usually short, contain mistakes and slang expressions, as well as emojis, useful for polarity identification. Additionally, their content is dynamic and often dependent on current events. So, the vocabulary is very rich and training a classifier on embedded texts is rather difficult because of certain inconsistency due to the mixture of domains in the dataset. Most posts in the dataset are written in Russian, although there is a large amount of posts in English and some posts in other languages which were kept as they didn't significantly influence the quality of the model. The average length of texts is 29 words. The distribution of classes in the dataset is very unbalanced. There are 6380, 2700 and 600 posts of neutral, positive and negative classes respectively.

3.2 Data Preprocessing

Raw data usually needs to be preprocessed before being passed to any machine learning algorithm as there are typically a lot of symbols which are meaningless for a given task and keeping them can reduce the quality of the model. The preprocessing of text means cleaning of noise such as cleaning of stop words, punctuation, terms which doesnt carry much weightage in context to the text.

When dealing with vector space models stemming or lemmatization is usually applied to the text to reduce the size of the dictionary and to give a classifier an ability to learn information from the words not running after words forms.

In this paper the following preprocessing is conducted. Stop words are removed using Python *nltk* library. The standard set of Russian stop words is

Table 1. F1 score for Logistic Regression over different stemmers

Stemmer	Positive class	Neutral class	Negative class	General score	Pos-Neg score
MyStem	0.47	0.82	0.06	0.68	0.40
SnowballStemmer	0.46	0.82	0.03	0.67	0.38
MorphAnalyzer	0.41	0.82	0.0	0.66	0.34

extended. HTML tags are removed using regular expressions.

Yandex stemmer *Mystem*[1], *Snowball Stemmer*[2] and *Morph Analyzer*[3] are applied for stemming. After stemming *TfIdf Vectorizer* from *sklearn* Python library with default parameters is used for vector representation of the texts. The *LogisticRegression* from *sklearn* with default parameters is then applied for classification. The results are shown in table 1. $F1$ metric score for each of the three classes is provided as well as an aggregated $F1$ score. $F1$ metric is calculated as weighted average of the $F1$ scores for all classes. In this section and in all further sections *General score* represents $F1$ metric calculated for all the three classes and *Pos-Neg score* represents $F1$ metric calculated as weighted average for positive and negative classes.

For our task MyStem showed better results in comparison and it was chosen for future investigations. In [5] it is mentioned that MyStem performed better.

3.3 Applied Classifiers

Among models that are commonly used for text classification are *Logistic Regression*, SVM, *Multinomial Naive Bayes*, *Random Forest Classifier*, neural networks.

The dataset is divided into train and test parts and cross validation method is used. The following classifiers over TF-IDF feature vectors with preprocessing described in the previous section are applied: *Logistic Regression*, *Random Forest Classifier*, *SVM*, *Gradient Boosting Classifier*, *KNeighbors Classifier*, *Multinomial Naive Bayes*. The results are shown in table 2.

SVM, *Logistic Regression*, *Random Forest Classifier* and *Gradient Boosting Classifier* demonstrated better results and are chosen for future consideration. The use of neural networks is not applicable in straight-forward way because it requires a large amount of labeled data for training. The *SVM* classifier over standard TF-IDF model form *sklearn* from this section is used as a baseline model, and the *Logistic Regression* classifier over standard TF-IDF model is shown as *simple Logistic Regression* for comparison with other *Logistic Regression* models in the next sections.

[1] `https://tech.yandex.ru/mystem/`
[2] `http://www.nltk.org/api/nltk.stem.html`
[3] `http://pymorphy2.readthedocs.io/`

Table 2. F1 score for different classifiers

Classifier	Positive class	Neutral class	Negative class	General score	Pos-Neg score
SVM	0.54	0.81	0.23	0.70	0.48
Logistic Regression	0.47	0.82	0.06	0.68	0.40
Random Forest Classifier	0.52	0.76	0.14	0.66	0.45
Gradient Boosting Classifier	0.40	0.82	0.11	0.66	0.35
KNeighbors Classifier	0.22	0.80	0.10	0.59	0.20
Multinomial Naive Bayes	0.25	0.81	0.0	0.61	0.20

3.4 Extracting Additional Data for Feature Engineering

It is rather difficult to achieve good quality just using standard classifiers. So the task was to add features that would help the model to learn information about the objects of all the classes. As there are a few examples of positive and negative classes the specified problem is hard to be solved using only available dataset. That is why it was decided to get unlabelled objects from VKontakte social network and to build a classifier that would extract more examples for negative and positive classes.

The following approach is used for this task. The *LogisticRegression* classifier is fitted on labelled data to predict probability for each of the three classes. The data is preprocessed as it is described in the previous section. The idea is that if the trained classifier is very sure that the object should be related to a certain class the probability of being mistaken when adding this object to the dataset with corresponding label is low. The thresholds for probabilities returned by trained model for positive an negative classes were chosen manually to supplement the positive and negative classes. About 3000 of new examples were extracted. These objects were used to calculate some statistics. The more detailed description is provided in the next sections.

3.5 Use of Emoticons as Features

There are many emoticons in the social network posts and they are very useful when identifying the sentiment polarity of the text. Smiley symbols received from the text are used in some ways. First they are added as features to TF-IDF model together with lemmatized words.

The probability distribution for each smile symbol to belong to each of three classes is calculated using both data from the original dataset and objects from extracted dataset which is described in the previous section.

$$P_{i,k} = \frac{N_{i,k}}{N_k},\ k \in \{-1,\ 0,\ 1\}$$

Where $N_{i,k}$ is the number of posts in which smiley symbol i is found and its label is equal to k in the united dataset, and N_k is the number of posts in which smiley symbol i is found in the united dataset.

Table 3. F1 score for Logistic Regression with emoticons and without emoticons

Model	Positive class	Neutral class	Negative class	General score	Pos-Neg score
				General score	
Model with emoticons	0.54	0.83	0.06	0.70	0.45
Baseline model	0.54	0.81	0.23	0.70	0.48
Simple Logistic Regression	0.47	0.82	0.06	0.68	0.40

The following features are added to the classifier. The average probability distribution for the three classes for all the smiley symbols in each post is calculated and these three numbers are used as features for the classifier. The number of smiley symbols in each post is also added as a feature to the model. The smiley label is calculated for each post as the class with the highest probability considering all the smiley symbols: $C_j = \text{argmax}_k P_{j,k}$, where $P_{j,k}$ is the probability of class k for document j considering all the smiley labels in the document j.

Smiles were also added as features to TF-IDF model and this raised the value of *F1* metric. The results can be viewed in table below. The *LogisticRegression* classifier over TF-IDF that is described in section 2 is chosen for consideration.

3.6 Application of Sentiment Lexicons

The marked sentiment lexicons from [17] are used to add new features to the classifier for it to be more precise. The vocabulary size is 7545 words and there are 5 possible classes in the dictionary. Each word can be given the mark $-2,\ -1,\ 0,\ 1,\ 2$ if it is considered to be very negative, negative, neutral, positive, very positive respectively. For each post the mark is calculated as the sum of sentiment labels for all the words in the post that are found in sentiment lexicon and this mark is added as a feature to the model.

$$Sent_j = \sum_{n=1}^{N_j} sent_n$$

Where N_j is the number of words in the text j found in sentiment lexicon, and $sent_n$ is the sentiment mark for word n from sentiment lexicon.

3.7 Application of Relevant Frequency Method

Relevant Frequency [5] helps to determine some kind of probability for each word to belong to each of the three classes according to the number of times it is found in the objects of a certain class compared to the number of times it is met in the objects of other classes.

The value is calculated as $RF_i^c = \log_2(\frac{a}{\max(1,b)})$. Where a is the number documents containing word i that belong to class c in the training dataset and b is the number documents containing word i that not belong to class c in the training dataset.

For each word from train dataset the value of *Relevant Frequency* is calculated using the above formula using train dataset and additional dataset which is

described in section 3.4.

The metric is calculated for each post as the sum of differences of the relevant frequency for the positive class and the relevant frequency for the negative class for each word in the training set, and this figure is added as a feature for the classifier.

$$RF_{sentence} = \sum_{words}(RF^{+}_{word} - RF^{-}_{word})$$

Where RF^{+}_{word} is the *Relevant Frequency* for a given word for positive class and RF^{-}_{word} is the *Relevant Frequency* for a given word for negative class.

3.8 Vector Space Model

The vector space model is used very often in machine learning tasks related to texts. In this approach individual documents are represented as vectors in term space. *Bag of words*, *TF-IDF*, *word2vec* are three most commonly used vector space models. *Bag of words* model was tried but it showed lower metric score in comparison. *FastText* [18] which provide *word2vec* space model was used for classification but it showed a little worse result in comparison to TF-IDF model which is chosen for term weighting.

In TF-IDF vector space model the term weight is given by product of $L_{i,j}$, G_i, N_j where $L_{i,j}$ is a local weight for term i in the document j, G_i is a global weight for term $i$and N_j is the normalization factor for document j [3].

Local weights are functions of how many times each term appears in the document, global weights are functions of how many times each term appears in the entire collection, and the normalization factor compensates for discrepancies in the length of the documents.

The following weighting formulas are applied. For local weight, binary ($BNRY$), within-document frequency ($FREQ$), log ($LOGA$), normalized log ($LOGN$), augmented normalized term frequency ($ATF1$) and the one that is used in *sklearn*. For global weights: inverse document frequency ($IDFB$), probabilistic inverse ($IDFP$), entropy($ENPY$), global frequency idf ($IGFF$), *Idf* from *sklearn*, no global weight ($None$). For normalization factor: cosine normalization ($COSN$), pivoted unique normalization ($PUQN$), no normalization ($None$).

For this task of classification the following TF-IDF models were chosen: $\langle$ *ATF1, None, COSN* $\rangle$ and $\langle$ *LOGN, ENPY, PUQN* $\rangle$.

The local weights from chosen models are calculated as follows:

$$ATF1 = \begin{cases} 0.5 + 0.5\ \dfrac{f_{i,j}}{x_j}, & if\ f_{i,j} > 0 \\ 0, if\ f_{i,j} = 0 \end{cases}$$

Where $f_{i,j}$ is the frequency of term i in document j and x_j is the maximum frequency of any term in document j.

$$LOGN = \begin{cases} \dfrac{1 + \log f_{i,j}}{1 + \log a_j}, & if\ f_{i,j} > 0 \\ 0, if\ f_{i,j} = 0 \end{cases}$$

Where a_j is the average frequency of the terms that appear in document j.

The global weight $ENPY$ is calculated as follows:

$$ENPY = 1 + \sum_{j=1}^{N} \frac{\frac{f_{i,j}}{F_i} \log \frac{f_{i,j}}{F_i}}{\log N}$$

Where F_i is the frequency of term i throughout the entire collection and N is the number of documents in the collection.

The normalization factors from chosen models are calculated as follows:

$$COSN = \frac{1}{\sqrt{\sum_{i=1}^{m} (G_i \ L_{i,j})^2}}$$

Where $L_{i,j}$ is local weight of term i in the document j, G_i is global weight of term i, m is the number of terms in document j.

$$PUQN = \frac{1}{(1 - slope) \ pivot + slope \ l_j}$$

Where l_j is the number of distinct terms in the document j, *slope* is set to 0.2, *pivot* is set to the average number of distinct terms per document in entire collection.

The application of these formulas to the model is described in the next sections. The other formulas and description for them can be found in [3].

3.9 Blending Strategy for Meta Ensemble

Ensemble methods, such as blending and stacking, are designed to boost predictive accuracy by combining the predictions of multiple machine learning models. Recent work has shown that the use of meta-features, additional inputs describing each example in a dataset, can boost the performance of ensemble methods [12] [11]. Classifiers over TF-IDF themselves showed moderate quality for the classification task. However, this was due to the fact that the classifiers considered most of the objects to belong to the neutral class. The blending model is chosen to consider the classifier output together with all the calculated features.

The two chosen TF-IDF models which are described in the previous section could perform better if to combine their predictions. The model with ⟨ *ATF1, None, COSN* ⟩ weights performs better general metric score in comparison to other models while ⟨ *LOGN, ENPY, PUQN* ⟩ model show higher metric quality for negative class.

The following approach is used. Two *Logistic Regression* classifiers are applied over vector representations received from each of the two chosen TF-IDF models. If the classifier over ⟨ *LOGN, ENPY, PUQN* ⟩ is very sure that the object should be related to the negative class and the probability for this object to belong to negative class predicted by the classifier over ⟨ *ATF1, None, COSN* ⟩ model is not low then the prediction of the first classifier is chosen. Otherwise the result of the second classifier is taken. These two thresholds were chosen using cross-validation method. The comparison of performance of these two models and their combination is shown in table 4. The pseudocode for described rule is as follows.

if $clf1.result > thr.clf1$ and $clf2.result > thr.clf2$ **then**
  obj.class = clf1.result
else
  obj.class = clf2.result
end if

The following designators are used in pseudocode:

- $clf1.result$ is the probability for negative class predicted by classifier over $\langle$ *LOGN, ENPY, PUQN* $\rangle$ TF-IDF model;
- $clf2.result$ is the probability for negative class predicted by classifier over $\langle$ *ATF1, None, COSN* $\rangle$ TF-IDF model;
- $obj.class$ is the resulting label for the object;
- $thr.clf1$ and $thr.clf2$ are the lowest probability of negative class for the first and the second classifiers respectively for the object to be related to negative class;

Table 4. F1 score for two Logistic Regressions and their combination

Model	Positive class	Neutral class	Negative class	General score	Pos-Neg score
ATF1, None, COSN	0.56	0.83	0.08	0.71	0.47
LOGN, ENPY, PUQN	0.58	0.79	0.22	0.70	0.52
Combined	0.56	0.83	0.23	0.72	0.50
Baseline model	0.54	0.81	0.23	0.70	0.48
Simple Logistic Regression	0.47	0.82	0.06	0.68	0.40

The applied blending strategy is as follows.The training subset of the dataset is divided into two parts. The *Logistic Regression*, *SVM*, *Random Forest Classifier*, *Gradient Boosting Classifier* fitted on the TF-IDF vector representations of the texts from the first part are applied to get the predictions on the second part of the train dataset and on test dataset. For *Logistic Regression* the output from one of the two classifiers is chosen as it is described in the previous paragraph. For the other two models $\langle$ *ATF1, None, COSN* $\rangle$ variant of TF-IDF is used. For *Logistic Regression* the probability distribution of classes is taken and for the other classifiers the predicted class label is used. The predictions of the classifiers are used as features for another classifier together with the other calculated features described in the previous sections. The feature that corresponds to *SVM* is excluded from the dataset because without this feature the metric score is higher. At this final step of blending the *Logistic Regression* is fitted on these features on the whole training data and predictions for the test data are received.

3.10 Rules Over the Classifier

Table 4 shows that the applied ensemble method increases the value of $F1$ metric. However there are still mistakes in the predictions while the value of some features corresponded to true label. For example, the classifier relates the text to neutral class but the values of *sentiment score* and *relevant frequency score* are very low, and the true label is negative. The thresholds were chosen using cross validation according to which the decision is made despite the resulting label of the classifier. The pseudocode of the rules over the classifier for a single object is as follows.

if $clf.result \; != 0$ **then**
  obj.class = clf.result
else if $obj.rf < rf.thr$ and $obj.sent < sent.thr$
    and $obj.neg > neg.thr$ and $obj.pred^{-} > pred^{-}.thr$ **then**
    obj.class = -1
else if $obj.rf > rf.thr$ and $obj.sent > sent.thr$
    and $obj.pos > pos.thr$ and $obj.pred^{+} > pred^{+}.thr$ **then**
    obj.class = 1
else
  obj.class = clf.result
end if

The following designators are used in pseudocode:

- $clf.result$ is the label predicted for the object by the classifier;
- $obj.class$ is the resulting label for the object;
- $obj.rf$ is the value of *Relevant Frequency* metric described in the 2 section;
- $obj.sent$ is the value of *Sentiment* metric described in the 2 section;
- $rf.thr$, $sent.thr$, $neg.thr$, $pos.thr$, $pred^{-}.thr$, $pred^{+}.thr$ are chosen thresholds for *Relevant Frequency*, *Sentiment Value*, number of negative and positive words from the sentiment lexicon, the probability returned by the classifier for negative and positive class respectively;

The described method helps to improve the quality of the model. The results of this step in comparison to previous step are shown in table 5.

3.11 Summary of Results

In this section an overview of experiments and evaluated components is provided.

Three variants of stemmers were applied for preprocessing: *My Stem*, *SnowballStemmer* and *MorphAnalyzer*. *My Stem* demonstrated better results in comparison both for the first steps of the applied model and for the whole pipeline. Stop words were removed using $NLTK$ Python library. All meaningless symbols were also removed with help of regular expressions.

Table 5. F1 score for Blending Model compared to previous steps

Model	Positive class	Neutral class	Negative class	General score	Pos-Neg score
Model with rules	0.62	0.83	0.25	0.74	0.55
Blending model	0.59	0.83	0.20	0.73	0.52
Model with emoticons	0.56	0.83	0.23	0.72	0.50
Baseline model	0.54	0.81	0.23	0.70	0.48
Simple Logistic Regression	0.47	0.82	0.06	0.68	0.40

Different classifiers were applied over vector representations of posts from dataset. *Logistic Regression*, *Random Forest Classifier*, *SVM*, *Gradient Boosting Classifier*, *KNeighbors Classifier*, *Multinomial Naive Bayes* were tested over different TF-IDF formulas for local, global weights and normalization.

Different TF-IDF formulas were considered. About 500 models were tested altogether and the best variants were chosen. It was noticed that many models with *None* for global weights and normalization performed better. This can point to the fact that local global weights and normalization dont play significant role in the task of sentiment analysis classification. It was also noticed that using *PUQN* normalization leads to better results in $F1$ score for negative class.

Logistic Regression, *RandomForestClassifier*, *GradientBoostingClassifier* as well as $\langle$ *ATF1, None, COSN* $\rangle$ and $\langle$ *LOGN, ENPY, PUQN* $\rangle$ variants for TF-IDF were chosen according to their better performance for being parts of blending model together with the generated features received from using *Relevant Frequency* method, sentiment lexicons and emoticons extracted from posts. Rules were applied over the predictions of the classifier to correct its mistakes.

Final model shows much better quality then standard classifiers. The main achievement of the proposed model is the increase of $F1$ score for positive and negative classes.

4 Conclusion

In this paper an approach for the problem of sentiment analysis that combines machine learning methods and sentiment lexicons is demonstrated. Different classifiers and different variants of formulas for computing TF-IDF values for vector representation of texts are evaluated and the best of them for a given task are chosen.

Final model shows better quality according to $F1$ score in comparison to standard classifiers. The main achievement of the proposed approach is that it helps to increase $F1$ score for positive and negative classes as standard classifiers show moderate quality on test dataset but only due to the fact that the majority of data points are classified as belonging to the dominant class because of strong class imbalance.

References

1. Liu, B.: Sentiment Analysis and Opinion Mining Morgan&Claypool Publishers (2012).
2. Pang, B., Lee, L.: Opinion Mining and Sentiment Analysis Foundations and Trends in Information Retrieval (2008).
3. Chistolm, E., Kolda, T.: New term weighting formulas for the vector space method in information retrieval. Technical Report Number ORNL-TM-13756, Oak Ridge National Laboratory, Oak Ridge, TN (1999).
4. Psomakelis, E., Tserpes, K., Anagnostopoulos, D., Varvarigou, T.,: Comparing Methods for Twitter Sentiment Analysis KDIR 2014, 225-232 (2014).
5. Kotelnikov, E., Klekovkina, M.: Sentiment analysis of texts based on machine learning methods. International conference on Computational Linguistics and Intellectual Technologies "Dialogue", 27-36 (2012).
6. Kotelnikov, E., Klekovkina, M.: The Automatic Sentiment Text Classification Method based on Emotional Vocabulary. RCDL, 81-86 (2012).
7. Kotelnikov, E., Bushmeleva, N., Razova, E., Peskisheva, T., Pletneva, M.: Manually created sentiment lexicons: development and research International conference on Computational Linguistics and Intellectual Technologies "Dialogue", 318-332 (2015).
8. Kaushik, C., Mishra, A.: A Scalable, Lexicon Based Technique for Sentiment Analysis. CoRR abs/1410.2265 (2014).
9. Loukachevitch, N., Rubtsova, Y.: Tweet sentiment analysis International conference on Computational Linguistics and Intellectual Technologies "Dialogue", 416-426 (2015).
10. Pazelskaya, A., Soloviev, A.: Method of emotions detection in Russian texts. International conference on Computational Linguistics and Intellectual Technologies "Dialogue", 510-521 (2011).
11. Baqapuri, A.: Twitter Sentiment Analysis. CoRR abs/1509.04219 (2015).
12. Sill, J., Takacs, G., Mackey, L., Lin, D.: Feature-Weighted Linear Stacking CoRR abs/0911.0460 (2009).
13. Kouloumpis, E., Wilson, W., Moore, J.: Twitter Sentiment Analysis: The Good the Bad and the OMG! International AAAI Conference on Web and Social Media (ICWSM) (2011).
14. Barbosa, L., Feng, J.: Robust Sentiment Detection on Twitter from Biased and Noisy Data. International Conference on Computational Linguistics (COLING), 36-44 (2010).
15. Ustalov, D.: Terms extraction from Russian texts using graph models. Graph theory and applications, 62-69 (2012).
16. Arkhipenko, K., Kozlov, I., Trofmovich, J., Skorniakov. K., Gomzin. A., Turdakov, D.: Comparison of neural network archItectures for sentiment anaLysis of Russian tweets. International conference on Computational Linguistics and Intellectual Technologies "Dialogue", 50-58 (2015).
17. Koltcov, S., Koltsova, O., Alexeeva, S.: LINIS Crowd SENT - a sentiment dictionary and a collection of texts with sentiment markup. `http://linis-crowd.org/`
18. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of Tricks for Efficient Text Classification. arXiv preprint arXiv:1607.01759. (2016).