Toxic Comment Classification Service in Social Network

Mikhail Dolgushin, Dayana Ismakova, Yuliya Bidulya(✉), Igor Krupkin, Galina Barskaya, and Anastasiya Lesiv

University of Tyumen, Tyumen, Russia

Abstract. The article discusses the development of an online tool for moderating the content of social network groups. The use of classification using machine learning methods is proposed as the main element of the system. The creation of the feature set of messages is assumed by extracting the content features of the text, as well as the use of word embeddings vectors. The authors conducted a series of experiments to find the best combination of vector representation, content features and classification method. Tests on a dataset of 11 thousand messages in Russian showed the result of 87% accuracy. The architecture of the group moderator's web application with the ability to automatically apply classification results to control users and display posts is proposed.

Keywords: Social media · Moderation · Toxic detection · Feature extraction · Text classification

1 Introduction

The object of study in this work is the moderation of messages and comments in a social network group to maintain a comfortable climate in this group, encouraging people to correctly express their opinions on the topic. Inappropriateness is most often expressed as the presence of toxicity, which is a rude, disrespectful comment that can cause someone to leave the discussion [1].

This paper presents the initial stage of developing a message moderation system intended for administrators of social network groups. For the most part, the automation of moderating messages on social networks comes down to filtering them based on the list of forbidden words compiled by the moderator. These filters allow you to remove messages that contain offensive language or some forbidden words. This approach often leads to the deletion of harmless messages, or, on the contrary, to the omission of comments with more subtle insults, built on the context and outwardly looking quite neutral. It is for this reason that social media comments are often moderated manually by moderators either before or after the [2] publication.

Automatic determination of toxicity in a text using machine learning methods has been discussed for a long time in publications, however, in the Russian-speaking segment of social networks, this problem has not yet been sufficiently developed. Research into

A. Karpov and R. Potapova (Eds.): SPECOM 2021, LNAI 12997, pp. 157–165, 2021.
https://doi.org/10.1007/978-3-030-87802-3_15

toxic detection is mainly focused on English, but some work in other languages also exists [3].

The aim of our work is to develop a software that could detect toxicity in the Russian segment of social media VKontakte (https://vk.com/). We study the possibility of using machine-learning algorithms to create a pre-trained model that is stored on the server side and connected as a web-service. One of the main development problems was the lack of Russian-language tagged datasets for training. Another problem is the need to use limited computing resources when developing a web service for moderating messages online.

We consider the problem of toxic detection as a text classification task with a binary choice: a system must predict whether a message should be blocked or non-blocked. Previously, experiments were carried out to select the best set of message properties to determine the toxicity, the presence of which determines the blocking of the message or verification of its author. Our research was carried out using two datasets. The first dataset is provided on the site of the well-known platform Kaggle [4]. With its help, we have selected the most effective algorithms for classifying messages by the presence or absence of toxic statements. A second dataset of 5000 messages we collected and annotated manually [5]. It was used to extract additional features for pre-filtering messages, as well as to train classifiers at the developed web application.

2 Related Work

2.1 Content Moderation of User Generated Content

A huge number of works are devoted to the problem of user generated content moderation. The relevance of the topic is due to the rapid growth in the number of resources with the necessary feedback from users. Content moderation is used in advertising, politics, business and other areas of activity, and the goals may vary. For example, in the electronic media, comments from news readers are screened, and moderation is used to decide which comments should be blocked in accordance with the policy of a particular newspaper [6]. Authors considered eight different categories of comments: disallowed content, threats, hate speech, obscenity, deception & trolling, vulgarity, language, and abuse. Each category provided its own scenarios of actions: from blocking an account to a temporary ban.

One more article [7] was also devoted to media moderation, which used a dataset of a newspaper of more than million comments with labels received from the newspaper's moderators and journalists. More specific categories were used as labels: calumniation, discrimination, disrespect, hooliganism, insult, irony, swearing, threat. Labels were used as attributes of objects, not target classes.

Many works are also devoted to the analysis of messages on social networks. The review article [8] shows the main features of the content and related problems of analysis: general terms, jargon, memes, vocabulary and cultural preferences of social groups can vary significantly; attackers can use special methods to "deceive" moderation algorithms on social networks; the presence of messages with spelling errors and typos complicates the technical processing of texts. In addition, messages represent short texts, which make

it difficult to consider the context and build models that reflect fixed expressions and word order.

2.2 Text Classification

Two approaches are used to classify texts during the moderation process. First, for each comment, semantic characteristics associated with toxicity markers are determined, such as the use of harsh words, negative or positive sentiment, the presence of emojis, the mention of proper names or pointers to locations, the type of utterance, characteristics of the message distribution, and so on. The values of these features make up the vector of each message, which is then used to train the system or determine the class using rules [9–11].

The second approach involves constructing a vector of each message using bag of word n-grams or word embeddings for further training of classifiers and neural networks [12]. The following text preprocessing scheme is used: tokenization, removal of stop words, POS markup, extraction of text tags, which results in the display of a bag of words or n-grams [13]. In addition, the representation of texts in embedding models is also widely used in classification problems. The most popular model is Word2vec, representing words from a dictionary mapped to vectors of real numbers [14].

A hybrid of these two approaches is also widely used, when the vector representation of texts is supplemented with content attributes [15].

About classification methods, the most popular and effective are convolutional neural networks, on which such systems for analyzing user content as Perspective (http://www.perspectiveapi.com/) are based. However, traditional methods such as SVM, naive Bayesian method, random forest, etc. [10] also show good results and provide more opportunities for implementation in the form of web applications, since they require less extensive computational resources.

3 Data and Preprocessing

We used two different datasets in this work. First dataset Kaggle Russian Language Toxic Comments Dataset (Dataset 1) [4] is a collection of annotated comments which was published on Kaggle in 2019 and consists of 14,412 unique comments, where 4,826 texts were labeled as toxic, and 9,586 as non-toxic. The length of comment ranges from 21 to 7400 characters, the average length is 175 characters.

For our own dataset (Dataset 2), 5,000 comments were collected and marked up from the social network Vkontakte [5]. It includes 1,150 texts labeled as toxic, and 3,850 texts labeled as non-toxic. The length of comment ranges from 11 to 4100 characters.

Message preprocessing includes tokenization, stop-word and punctuation removal, lemmatization provided by NLTK [16] and Pymorphy2 [17] libraries.

4 Features

4.1 Vector Models

We implemented the set of features that used as the input data for the machine learning classification. At the output, we get labels for messages, depending on presence of toxic in the message text. The features are described below.

In our experiments, we use the following vector models for feature extraction:

1. Term Frequency (TF) vector model is a matrix of unigrams usage frequencies in documents [18]. In this work, we didn't use the TFIDF model, since it shows itself better on long text messages, the use of the TF-matrix is more reasonable. For the same reason higher-level n-grams are not considered. For the TF implementation matrix, the Count Vectorizer acquired in the Scikit-learn library [19] is used with parameter: size 300.
2. Word2Vec [20] is a tool for calculating vector representations of words that implements two main architectures: Continuous Bag Of Words and Skip-gram. The input is a text corpus, and the output is a set of word vec-tors. We applied Word2vec model of Gensim with the following model parameters: size 150, window 10. The final value for each word is obtained by averaging all the numbers in its vector.
3. Doc2Vec [21] is a tool like Word2Vec, but the input is a whole text document. We applied Doc2vec model of Gensim with the following model parame-ters: size 10. The final value for each word is obtained by averaging all the numbers in its vector.
4. FastText [22] is a library containing pre-trained ready-made word representations and a classifier, that is, a machine learning algorithm that breaks words into classes. We used FastText model of Gensim with the following model parame-ters: size 150, window 5.

We applied scaling from 0 to 100 results using the MaxMinScaler class from the Scikit-learn library [19] for all described models except Count Vectorizer.

4.2 Content Features

We have expanded the word vector features with additional features determined from the message content:

- the presence in the text of persona's names,
- the presence in the text of names of locations,
- the presence in the text of names of organizations,
- positive, negative, neutral sentiment,
- the presence of rude words.

These features were extracted using the Dostoevsky [23], SpaCy [24] libraries, as well as the vocabulary of rude words.

5 Experiments and Results

5.1 Classification of Toxic Comments

The goal of the first experiment was to find the best combination of a vector model as feature set and a classification algorithm from the following:

1. Support vector classification (SVC) [25] is a set of similar supervised learning algorithms used for classification and regression analysis problems.
2. The Multinomial Naive Bayesian Classifier (MNB) [25] is a modification of the Naive Bayesian Classifier where function vectors represent the frequencies with which certain events were generated by a polynomial distribution.
3. A support vector machine modification using a naive Bayesian algorithm (NBSVM) is a support vector machine implementation in which vectors are constructed based on the coefficients of the logarithm of the naive Bayesian algorithm as characteristic values. The NBSVM implementation is taken from the Kaggle [26].
4. The Random Forest Classification (RFC) [25] is a classification model using a machine learning algorithm that uses an ensemble of decision trees (decision trees).

For the classifiers Multinomial Naive Bayesian, Support vector classification and Random Forest the classes presented in the SciKitLearn library were used [19]. The experiments were carried out on the Dataset 1. We used cross-validation with stratification to prevent overfitting and to decrease the influence of class imbalance in all classification methods. In addition, a grid search was used in the implementation of the Scikit-learn library for selection of classification parameters [19].

Table 1 demonstrates a F1-measure and accuracy values calculated for features described above in Sect. 4.1.

As follows from the results of Table 1 (in bold), the best values of the F-measure and accuracy in this experiment were obtained using the FastText vectorization and the SVC classifier.

5.2 Content Features Selection

After feature extraction (see Sect. 4.2) we calculated the Pearson correlation for each feature with the target sign of the presence or absence of toxicity in the tagged dataset. The calculated values are shown in Table 2.

The largest coefficient is shown by features: negative, neutral sentiment and the presence of rude words. We expanded word vectors by features with the highest correlation and trained again on the same dataset. The best result was shown by the Word2Vec representation and the SVC classifier. It turned out that the expansion of the feature set led to a low increase in accuracy from 0.864 to 0.872. This model was later used in the development of a web application for moderating social network comments.

Table 1. The results of message classification.

Vector model	Classifier	Class "Toxic", F1	Class "Un-toxic", F1	Accuracy
Term frequency	SVC	.571	.858	.787
	MNB	.385	.839	.745
	NBSVM	.568	.859	.787
	RFC	.410	.844	.753
Doc2Vec	SVC	.745	.892	.851
	MNB	.744	.889	.845
	NBSVM	.715	.864	.816
	RFC	.763	.891	.851
Word2Vec	SVC	.769	.902	.862
	MNB	.742	.894	.849
	NBSVM	.778	.894	.857
	RFC	.771	.903	.864
FashText	SVC	**.786**	**.906**	**.870**
	MNB	.776	.889	.852
	NBSVM	.779	.904	.867
	RFC	.748	.899	.856

Table 2. Pearson's correlation coefficients.

Feature	Coefficient
Person	−0.013
Location	−0.063
Organization	−0.017
Positive	−0.047
Negative	0.361
Neutral	−0.267
Rude words	0.413

6 System Architecture

After analyzing the results of the experiments, we chose an approach to implement a Web application for the automatic determination of toxicity in VKontakte messages.

The application implements functions executed within the following modules (see Fig. 1):

- "ToxicAuth" is a Laravel (https://laravel.com) framework-based backend service that is present an application programming interface (API) for moderator authentication.
- "MainAPI" is a backend service for moderating a social network group. It works with data about comments, users, messages and groups and allows you to assign restrictions to messages or users who have received toxicity labels as a result of classification. Essentially, it links the classification module to the moderator interface.
- "ClassifierAPI" is an application implemented in the Python programming language and the Flask (https://flask.palletsprojects.com) framework. It trains the classifier and displays the class label for an individual message or comment text. The trained model is saved in.csv format and stored on the server. The module interacts with *vk-api* (https://pypi.org/project/vk-api/) to receive messages and comments from the social network, as well as with the "MainAPI" module, transmitting data about toxic messages and users who will be penalized in any way for toxic messages.
- "ClassifierGUI" is a separate graphical interface developed for working with the training sample and the output of training metrics. A graphical interface was developed using the PyQT5 library and developed using the PyQT5 Designer (https://pythonscripts.com/pyqt5).

"ToxicFront" is a frontend application developed using the React.js framework (https://reactjs.org). This is an interface for a group moderator that receives information about users submitting toxic messages or comments and can warn or block such users.

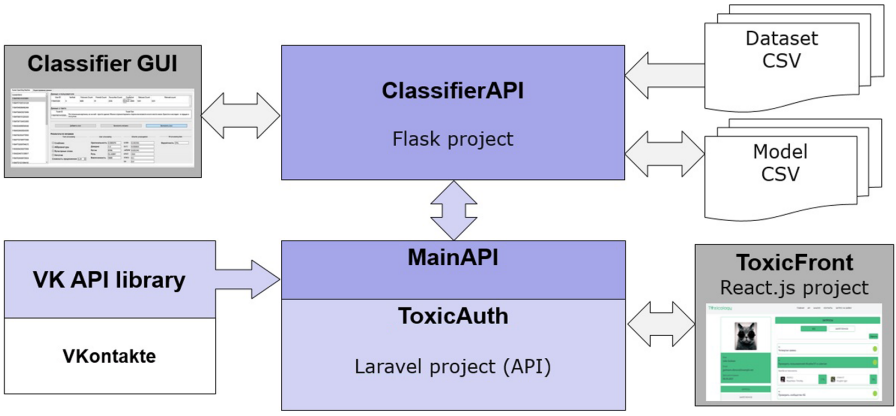

Fig. 1. System architecture of the message moderation service.

7 Conclusion

We investigated the possibility of using machine learning methods to develop the web-application for moderation of Russian-language messages and comments on a social network. Experiments on a sample of eleven thousand messages have shown that the most

suitable result is achieved using vectorization of the Word2vec model and classification by the SVM method with the expansion of content features: the presence of rude words, negative and neutral sentiment. The maximum accuracy has reached 87%.

As a result of this work, the first version of software for moderation of social network messages in the Russian-speaking group of VK has been developed. It trained on a hand-made dataset and currently being tested on five groups of social network users.

References

1. Georgakopoulos, S.V., Tasoulis, S.K., Vrahatis, A.G., Plagianakos, V.P.: Convolutional neural networks for toxic comment classification. arXiv preprint arXiv:1802.09957 (2018)
2. Medialogiya–monitoring and analysis of media and social networks (rus.). https://www.mlg.ru
3. Corazza, M., Menini, S., Cabrio, E., Tonelli, S., Villata, S.: A multilingual evaluation for online hate speech detection. ACM Trans. Internet Technol. Assoc. Comput. Mach. **20**(2), 1–22 (2020). https://doi.org/10.1145/3377323.hal-02972184
4. Russian Language Toxic Comments. https://www.kaggle.com/blackmoon/russian-language-toxic-comments
5. "Toxicology" project: vk_comments_DS. https://github.com/mihatronych/files/blob/main/ds_of_toxic_messages_from_vk/our_toxic_vk_comments_data.csv
6. Shekhar, R., Pranjić, M., Pollak, S., Pelicon, A., Purver, M.: Automating news comment moderation with limited resources: benchmarking in croatian and estonian. J. Lang. Technol. Comput. Linguist. **34**, 49–79 (2020)
7. Pavlopoulos, J., Malakasiotis, P., Androutsopoulos, I.: Deeper attention to abusive user content moderation. In: EMNLP, pp. 1125–1135. Copenghagen, Denmark (2017)
8. Levonevskiy, D., Malov, D., Vatamaniuk, I.: Estimating aggressiveness of russian texts by means of machine learning. In: Salah, A.A., Karpov, A., Potapova, R. (eds.) SPECOM 2019. LNCS (LNAI), vol. 11658, pp. 270–279. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26061-3_28
9. Lee, J.-T., Yang, M.-C., Rim, H.-C.: Discovering high-quality threaded discussions in online forums. J. Comput. Sci. Technol. **29**(3), 519–531 (2014)
10. Plaza-del Arco, F.M., Molina-Gonzalez, D., Martın-Valdivia, T., Urena-Lopez, A.: SINAI at SemEval-2019 Task 6: incorporating lexicon knowledge into SVM learning to identify and categorize offensive language in social media. In: The 13th International Workshop on Semantic Evaluation (SemEval) (2019)
11. Chernyaev, A., Spryiskov, A., Ivashko, A., Bidulya, Y.: A rumor detection in Russian tweets. In: Karpov, A., Potapova, R. (eds.) SPECOM 2020. LNCS (LNAI), vol. 12335, pp. 108–118. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60276-5_11
12. Pavlopoulos, J., Thain, N., Dixon, L., Androutsopoulos, I.: ConvAI at SemEval-2019 Task 6: offensive language identification and categorization with perspective and BERT. In: SemEval, Minneapolis, USA (2019)
13. Pietro, M.D.: Text Classification with NLP: tf-idf vs Word2Vec vs BERT. https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794
14. Camacho-Collados, J., Pilehvar, M.T.: From word to sense embeddings: a survey on vector representations of meaning. arXiv:1805.04032. Bibcode:2018arXiv180504032C (2018)
15. Waseem, Z., Hovy, D.: Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, pp. 88–93 (2016)

16. NLTK documentation. https://www.nltk.org
17. Morphological analyzer pymorphy2. https://pymorphy2.readthedocs.io
18. Document-term matrix. https://en.wikipedia.org/wiki/Document-term_matrix
19. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830. JMLR (2011)
20. Rehurek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50. Valletta, Malta, May. ELRA (2010). http://is.muni.cz/publication/884893/en
21. Gensim: Doc2vec. https://radimrehurek.com/gensim/models/doc2vec.html
22. Mestre, M.: FastText: stepping through the code. https://medium.com/@mariamestre/fasttext-stepping-through-the-code-259996d6ebc4
23. Dostoevsky: Sentiment Analysis Library for Russian Language. https://pypi.org/project/dostoevsky
24. SpaCy: Industrial-Strength Natural Language Processing. https://spacy.io
25. Wang, S., Manning, C.D.: Baselines and bigrams: simple, good sentiment and topic classification, Department of Computer Science, Stanford University, Stanford 94305. https://nlp.stanford.edu/pubs/sidaw12_simple_sentiment.pdf
26. Wang, Z.: NBSVM. https://www.kaggle.com/ziliwang/nbsvm