Toxic analysis of Russian comments

Evgeny Bochkarev

January 2023

Abstract

This document is the report for the Toxic Analysis of Russian Comments. A link to project code here: https://github.com/bochkarev1906/nlp_course

1 Introduction

Currently, the use of data analysis leads to outstanding hypotheses and results in almost all branches of the modern world: from medicine to economics.

When there is too much data, their analysis becomes very laborious and almost impossible manually, without the use of modern data analysis methods, without the use of machine learning.

One of the branches of machine learning is the processing and analysis of natural languages, or in other words NLP. There are already a lot of problems in this area, ranging from chat bots and automated translators to question-answer systems and named entity recognition.

In this paper, such topics as the analysis of the toxicity of comments are considered. Unfortunately, the problem of inappropriate content and, in particular, toxic comments on the web is very relevant due to the aggravated political and economic situation in recent months. So, many communities in social networks and Telegram channels began to massively disable the ability for users to leave comments in order to avoid conflicts, provocations and offenses. Such decisions are clear evidence of how acute the problem of toxic online communication is, especially at the present time.

Toxic (unacceptable) comments are comments by network users containing inappropriate content: insults, obscene language, threats and aggression (the concepts of «unacceptable» and «toxic» comments are synonymous in this work). And the task of detecting such comments is now quite relevant.

2 Related Work

For this dataset, all the solutions of various users on the Kaggle website were considered [1]. It was observed that most of them did not use all the necessary pre-processing of comments, and also almost all used only stemming or did not use any word normalization methods at all. And as it turned out in the future, it is data lemmatization that will give the best result. Probably, many did

not use it, since it is quite difficult to find a more or less correct implementation of this method specifically for the Russian language.

Almost all existing solutions used Logistic Regression as a base model, and further users tried to improve the quality of classification using the selection of hyperparameters and the use of various other machine learning models.

As a result, after reviewing various existing solutions to this problem, we managed to fix the best value of the metric F1 = 0.7805.

We also found several articles on current topics that described general approaches and various machine learning models, but the total values of the metrics were not calculated or the calculation was made for other metrics.

3 Model Description

One of the main stages of preparing data before training is exactly the vectorization of texts, since it is after this part is completed that completely numerical vectors will be ready, which will only have to be submitted to the training model.

There are many different methods for vectorizing texts. Some of them are based only on various vocabulary statistics, for example, whether there is such a word in a particular text or not. It is clear that this is not enough, since when the words are rearranged, the very meaning of the sentence very often changes, this is especially noticeable in the not particularly structured Russian language. This architecture is called a «bag of words». We have already said that this method does not take into account the order of words in sentences, but it also turns out that all words have the same weight, which is also not correct.

Then the previous method is replaced by the TF-IDF method, which slightly improves and eliminates the shortcomings of the previous one. In this case, words that occur many times in texts, in almost every sentence, and these are almost always different conjunctions, particles and prepositions, will have much less weight than words that are only a couple of times found in different texts.

This method is almost the most popular and really simple and understandable, which is almost always used when analyzing texts. Also, this vectorization option often shows good results, which is especially good for a fairly straightforward method.

Other word vectorization options, such as «word2vec», «fastext» and «BERT», were also tested, but they did not significantly improve the quality of the classification compared to the fast TF-IDF. These pre-trained models turned out to be too heavy for a specific task and did not justify the use of a large amount of memory, and besides, the operating time increased many times over.

4 Datasets

4.1 Data description

The dataset includes 14,442 different comments written in Russian and 1 target variable (comment toxicity):

	comment	toxic
0	Верблюдов-то за что? Дебилы, бл...\n	1.0
1	Хохлы, это отдушина затюканого россиянина, мол...	1.0
2	Собаке - собачья смерть\n	1.0
3	Страницу обнови, дебил. Это тоже не оскорблени...	1.0
4	тебя не убедил 6-страничный пдф в том, что Скр...	1.0
...	...	...
14407	Вонючий совковый скот прибежал и ноет. А вот и...	1.0
14408	А кого любить? Гоблина тупорылого что-ли? Или ...	1.0
14409	Посмотрел Утомленных солнцем 2. И оказалось, ч...	0.0
14410	КРЫМОТРЕД НАРУШАЕТ ПРАВИЛА РАЗДЕЛА Т.К В НЕМ Н...	1.0
14411	До сих пор пересматриваю его видео. Орамбо кст...	0.0

14412 rows × 2 columns

Figure 2: The example of dataset

As you might guess, a value of 1 means that the comment is toxic, that is, it contains offensive or degrading overtones. A value of 0 means that the comment text consists entirely of acceptable words.

Using the info() function, it was found that there are no missing values in the data, and the target function is of type float. The result of the output of this function is shown in Figure 3.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14412 entries, 0 to 14411
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   comment  14412 non-null  object 
 1   toxic    14412 non-null  float64
dtypes: float64(1), object(1)
memory usage: 225.3+ KB
```

Figure 3: Function info()

4.2 Target variable analysis

Before building various charts and graphs, as well as using machine learning models, it is necessary to pre-process the data. And in our case, text analysis is a very time-consuming process that requires special attention.

In our data, there is only one categorical feature - the target variable. For further correct operation of machine learning algorithms, it must be cast to an integer type, which has already been done in our data. Before further data analysis, let's check how the target variable is distributed. Thus, in Figure 4, we obtain the following distribution of the target variable:

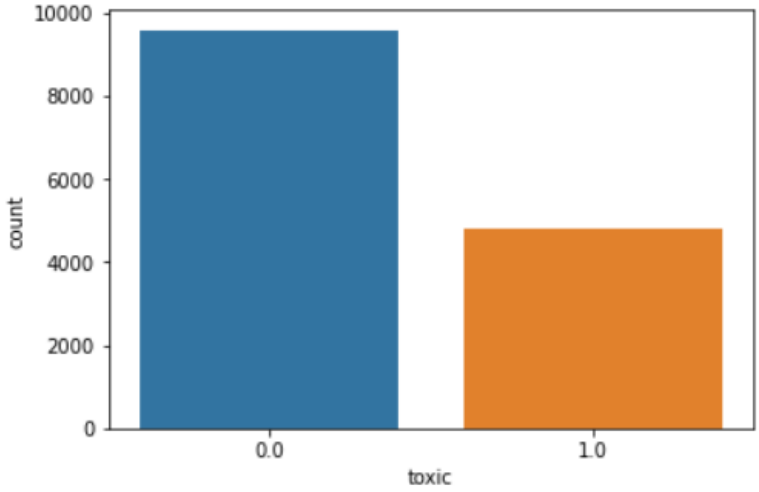

Figure 4: Distribution of target variable values

We get that about 33.5% of toxic messages, and 66.5% of acceptable messages. We see that there are almost two times fewer unacceptable comments than ethically correct ones. This tells us about the imbalance in the target attribute. This fact will need to be taken into account when further dividing the data into training and test samples.

4.3 Exploratory text analysis

All comment texts were examined to find various substrings that are unimportant for subsequent prediction.

Figures 5 and 6 provide examples of various comments that are non-toxic and toxic, respectively. With the naked eye, the difference in the comments is immediately noticeable.

```
Антонио, ты весь в красном!

Не поверите. Н о л ь.

ПОТОМУ ЧТО ЭТО МАСТЕР. А ВЫ ДЕЙСТВИТЕЛЬНО БЕ-БЕ-БЕ..

Извини тебе в грязном ходить приятно?

Ты в маём серце казёл.

У вас пост поломался.
```

Figure 5: The example of correct comments

```
Верблюдов-то за что? Дебилы, бл...

Собаке - собачья смерть

Ебать тебя разносит, шизик.

Обосрался, сиди обтекай

Уроды!! у нас в семье 3 поколения там родились
```

Figure 6: The example of toxic comments

When analyzing all the comments, about 1000 texts were found in which there are various Latin letters, despite the fact that the comments should initially be written in Russian. It was found that almost all of these words either consist of one Latin letter, or these words point to a link to a specific site, so it was decided to remove such words and leave only Russian letters. Figure 7 shows an example of such Latin words:

```
['https', 'psv', 'c', 'u', 'docs', 'd', 'c', 'ezgif']
['wago']
['Fia']
['E', 'nt', 'bpj', 'hnf']
['xD']
['NewZanaModel']
['IQ']
['gif']
['p']
['rf']
['Gifx', 'Gifx']
['R']
['svin', 'ja']
```

Figure 7: The example of Latin words

Then a variable was added, such as the length of the words in the comments. With the help of the seaborn library, distributions of the number of comment words depending on the target variable were constructed.

Figure 8 shows just such a graph, thanks to which we can assume that the average number of words is the same and does not depend on whether the text is toxic or not.

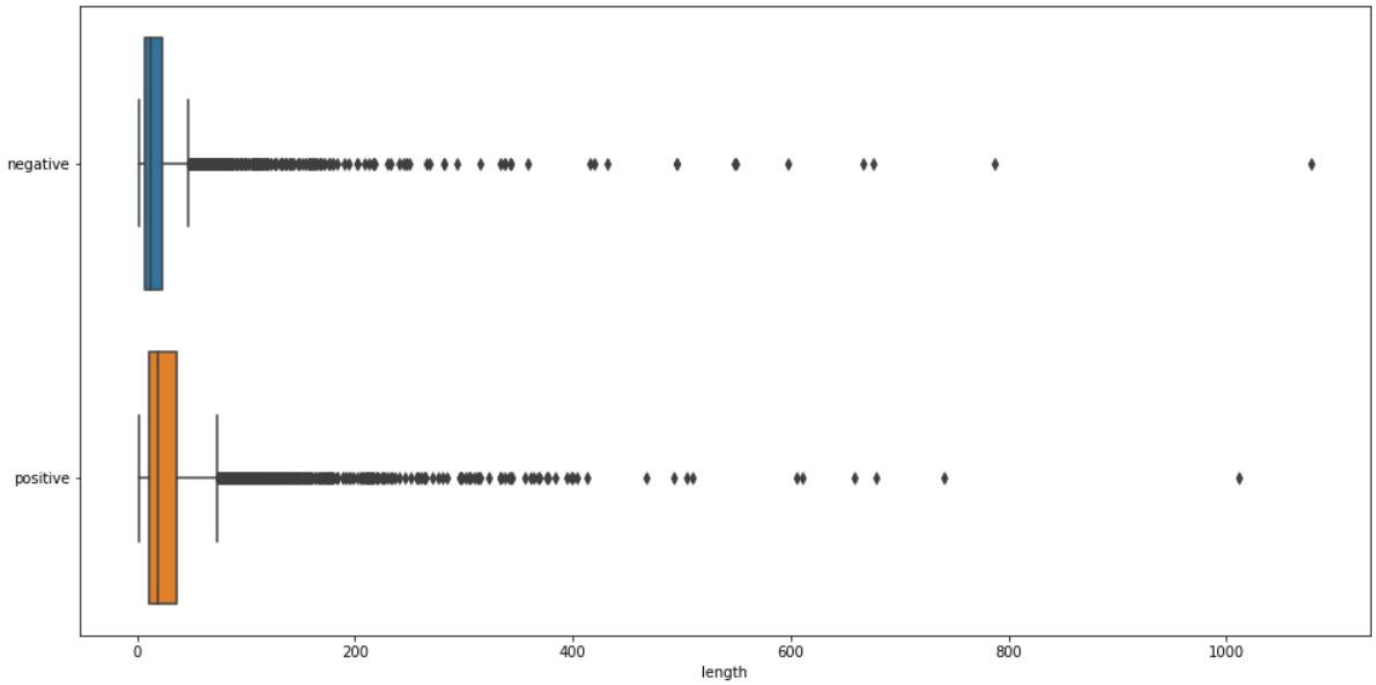

Figure 8: Boxplot for the target feature

4.3 Preprocessing texts

For further analysis, it is necessary to translate all messages to lower case, thereby we are approaching a common understanding of words, for example, the words «Это» and «это» will already be considered as one and the same, and not two different ones.

It was also necessary to remove all stop words in all comments. These are words that come across in almost every sentence, that is, they have a very high frequency, while almost no specific meaning was carried. Such words are particles, various pronouns, conjunctions, interjections and others.

Then various punctuation marks and other text decorations, such as \n, were removed. Also all numbers, double spaces

All comments were subjected to tokenization, that is, division into words. Further, each word separately must be brought to a general form. This is done either with stemming or lemmatization.

Stemming is a kind of normalization of words. Normalization is a technique in which a set of words in a sentence is converted into a sequence in order to reduce search time. Words that have the same meaning but have some differences depending on context or sentence are normalized. For example, the words «хороший» and «хорошая» have the same root, according to which some words should be compared, and unnecessary endings should be deleted.

Lemmatization is an algorithmic process of finding the lemma of a word depending on its meaning. Lemmatization usually refers to the morphological

analysis of words, the purpose of which is the removal of inflectional endings. This helps in returning the base or dictionary form of a word, which is known as a lemma. For example, the words «хочешь», «хотела» and «хочу» have the same lemma «хотеть», to which it is necessary to bring all such words similar in meaning.

Finally, we have the following text processing:

1. Reduction to lower case;
2. Removing of English words;
3. Removing numbers;
4. Removing stop words;
5. Removing of other frequently occurring words;
6. Removing punctuation marks;
7. Tokenization;
8. Stemming and lemmatization.

As a result, Figure 9 shows an example of the text before its pre-processing, and Figure 10 shows an example of the same text, but after pre-processing using stemming, and Figure 11 using lemmatization.

```
Страницу обнови, дебила кусок. https://dfb.rf Это тоже не оскорбление,
а доказанный факт - не дебил про себя во множественном числе писать не будет.
 7841
```

Figure 9: An example of a comment before processing

```
['страниц', 'обнов', 'деб', 'кусок', 'оскорблен',
 'доказа', 'факт', 'деб', 'множествен', 'числ', 'писа']
```

Figure 10: An example of a comment after processing with stemming

```
['страница', 'обновить', 'дебил', 'кусок', 'оскорбление',
 'доказать', 'факт', 'дебил', 'множественный', 'число', 'писать']
```

Figure 11: An example of a comment after processing with lemmatization

Judging by the words already processed, it seems that lemmatization works better in terms of the semantic load of words. But we will test this hypothesis already when we get the first results of quality metrics separately for stemming and lemmatization.

Also, after processing the words, a word cloud was built for various classes of the target variable. It is not difficult to guess that in Figure 12, on the left, there are the most frequently occurring words from toxic comments, and on the right, the most frequent words for acceptable comments.

Figure 12: Word clouds for different values of the target variable

5 Experiments

5.1 Metrics

There are various quality assessment metrics for classification tasks. It is necessary to consider each of them and choose one or more suitable for a particular task [2]

1. Accuracy (accuracy) - one of the simplest metrics. It shows the number of correctly labeled class labels (true positives and true negatives) out of the total data. But this metric has a lot of disadvantages. Firstly, it does not take into account the initial distribution between classes 0 and 1. Secondly, this metric does not take into account the ratio of false positives of the model, which can be critical, for example, in the medical field.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. Precision (precision) - a metric that shows the number of true positive outcomes from the entire set of positive labels. This metric is used when the cost of a false positive is high. For example, when the cost of further testing of a patient is high and we cannot send all patients for such testing.

$$Precision = \frac{TP}{TP + FP}$$

3. Recall (completeness) - a metric that determines the number of true positives among all class labels that have been determined to be positive. This metric is used when the cost of a positive class misrecognition error is high. For example, when diagnosing a deadly disease.

$$Recall = \frac{TP}{TP+FN}$$

4. F1-score (F-score) - a metric that is used when Precision and Recall are equally significant and is calculated as their harmonic mean.

In this problem, most of the solutions reviewed use the F-measure metric. I think that this metric is completely suitable for this topic, since Precision and Recall are equally significant.

$$F = 2\frac{Precision \times Recall}{Precision + Recall}$$

In this problem, most of the solutions reviewed use the F-measure metric. I think that this metric is completely suitable for this topic, since Precision and Recall are equally significant.

5.2 Experiment Setup

Since the data initially did not have a training and test set, it was necessary to split it. 80% of the data was selected for the training sample and the remaining 20% for the test sample, on which the quality will be checked in the future, that is, the metric will be calculated. Also, data separation took place taking into account the fact that the target variable is not balanced.

To train the models, the following function was implemented:

```
def scoring(model, tokenizer, params):
  model_pipeline = Pipeline([
      ('vectorizer', TfidfVectorizer(tokenizer = lambda x: tokenizer(x))),
      ('model', GridSearchCV(model(), params, cv=5, scoring='f1'))])
  model_pipeline.fit(X_train, y_train)
  pred = model_pipeline.predict(X_test)
  return precision_score(pred, y_test), recall_score(pred, y_test), f1_score(pred, y_test)
```

Figure 1: The function that implements the training of models

As you can see, vectorization occurs first using TF-IDF, and then the training itself on the initial data, using the selection of the best parameters for each model separately.

Also, this whole function was run three times on each model. First for just tokenized words, then using stemming, and then using lemmatization.

The following machine learning models were used: logistic regression and the support vector model. As it turned out in the future, it is linear models with a bunch of TF-IDF vectorization that give the best results. But still, more complex models such as decision trees and a random forest model were tested. Even with the selection of the best parameters, the last two models showed much worse results than the linear models. Various variations of the naive Bayes model were also considered, which also showed a good result.

5.3 Baselines

Models such as logistic regression on tokenization data without normalization texts were used as a base model so that there was something to compare the value of the metrics with.

6 Results

Table 1 presents the final results for all types of text normalization and various machine learning models without parameter selection.

Model	F-score		
	Tokenization	Stemming	Lemmatization
Logistic Regression	0.6006	0.7230	0.7325
SVM	0.7480	0.7797	0.7851
Decision Tree	0.6046	0.6607	0.6637
Random Forest	0.6456	0.7039	0.7020
Multinomial NB	0.4655	0.6126	0.6169
Complement NB	0.7300	0.7763	0.7765

Table 1: Metric values for various combinations of text normalization and machine learning models

All models, fitting parameters and metrics on test data are presented in Table

Model	Parameters	F-score		
		Tokenization	Stemming	Лемматизация
Logistic Regression	'C' : [0.1, 0.5, 1, 10]	0.7338	0.7814	0.7850
SVM	'penalty': ['l1', 'l2'], 'C' : range(1, 10, 2)	0.7518	0.7797	0.7851

Decision Tree	'criterion': ['gini', 'entropy'], 'max_depth' : range(5, 200, 40)	0.5071	0.6152	0.6137
Random Forest	'criterion': ['gini', 'entropy'], 'max_depth' : range(50, 250, 50)	0.5023	0.6357	0.6358
Multinomial NB	'alpha' : [0.01, 0.1, 1, 2, 5, 10, 20]	0.7515	0.7844	0.7786
Complement NB	'alpha' : [0.01, 0.1, 1, 2, 5, 10, 20]	0.7933	0.8277	0.8171

Table 2: Classification results for models with fitted parameters

As you can see, the naive Bayes model turned out to be the best result. Although this model seems to be quite simple and easy, but specifically on our task, it showed the highest value of the metric f1 = 0.8277. It can also be seen that in all linear models, text normalization using lemmatization always gives a better result than stemming, that is, a simple trimming of word endings. And non-linear models, such as decision trees and random forest, give much worse metric values.

Conclusion

Within the framework of this project, a complete analysis of the texts was carried out, and some primary patterns were identified. Then all texts were brought to a normalized and tokenized form. After that, all texts were vectorized using the function TF-IDF. Next, various machine learning models were tested and the optimal parameters for them were selected.

References

1. Russian Language Toxic Comments - Kaggle - URL: https://www.kaggle.com/datasets/blackmoon/russian-language-toxic-comments
2. Metrics for classification. URL: https://webiomed.ru/blog/osnovnye-metriki-zadach-klassifikatsii-v-mashinnom-obuchenii/