Building a High-Precision Classifier on Unbalanced Sets of Short Text Messages[*]

Andrey S. Mokhov[1] [0000-0002-1979-6411], Vladimir O. Tolcheev[1], Artem A. Borodkin[1]

[1] National research university "Moscow power engineering institute", 111250, Moscow, Russia

asmokhov@mail.ru
tolcheevvo@mail.ru
nowar1@mail.ru

Abstract. This paper considers the building of a classifier in the case of unbalanced training sets, which consist of very short text messages. We use machine learning to automatic classifier construction. High quality of classification is the main requirement for a classifier (we use precision and recall as the criterion of quality). Imbalance in sets means, that the number of texts belonging to one class is significantly lower than those belonging to the other classes. The development of classifiers on the base of unbalanced samples is a well-known complex problem in Text Mining. A lot of theoretical and experimental researches are devoted to searching for a solution. This is especially the case for relatively short text documents, for which the use of known approaches is not always effective. We give a brief review of methods and argue using Gradient Boosting Machine (GBM) for classification short text messages in the case of unbalanced training sets. We diagnose situations in which the classifier fails, analyze them, form a hypothesis on how we can improve text processing, and configure the settings of the classifier. Also, we investigate and compare two schemes of preprocessing - Bag of Words and Word2Vec. A final section is devoted to experimental results. Experiments were carried out on datasets collected in the process of the daily operation of one of the large bank's branches. These real data have a rather significant imbalance. In the case of the unbalanced training set our voting approach (GBM) improves performance considerably.

Keywords: Text Mining, Text Classification, Bag of words, Word2Vec, Gradient Boosting Machine, Preprocessing of text documents, Feature extraction, Precision, Recall.

1 Introduction

With the rapid growth of unstructured information in Internet, classification has become

[*] Copyright 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

one of the key techniques for handling and organizing text data. Methods of classification are used to analysis of documents (scientific articles, news, e-mails, WWW-pages, product reviews, etc.) and dividing them into classes (or categories). This problem arises in various fields of human activity. For example in large corporations (banks, governmental organizations, offices of e-commerce) are frequently required to automatically identify the type of messages sent by customers (or users) and correctly recognize their categories and direct them to the proper department for processing.

In this paper, we are dealing with the problem of automatically dividing service requests that come through the corporate network of the Bank. The purpose is to distribute requests based on the content and send them for execution to the relevant department. These requests may have the following form: "Please replace the cartridge in office #12, printer – HP Color LaserJet CM 1312 MFP, suitable time – from 10 to 12 a.m.”. To be able to solve such a real-world problem it is necessary to develop a classifier, which will efficiently distribute the requests according to the classes (departments) automatically. By doing so we want to exclude an operator from the process of distribution of requests, but maintain the high quality of classification (precision and recall). Thus, the processing of requests should take place without any human participation (more correctly with no or minimum human effort).

It is important to emphasize that the precision and recall of the classifier should not be lower than the precision and recall of the operator having instructions and rules for the direction of the request to the departments. Moreover, it is even possible to increase the accuracy shown by the operator due to eliminating a human impact (errors occurring because of fatigue, negligence, loss of concentration, etc.).

The main approach of creating such classifiers is machine learning – the construction of a classification function (rules) on the examples. To develop such classifiers, we need labeled data (documents and their corresponding categories - labels). The labels are assigned by the experts (supervised learning). A set of labeled data is used to predict the labels of unseen data. In this research, the number of classes is six and all requests can belong to only one category

A variety of methods have been proposed to categorize unstructured information. The most widely used classifiers are Naive Bayes, Logistic Regression, Nearest Neighbor, Decision Trees, Neural Networks, Support Vector Machines, and others [1,2,3]. Choosing the most appropriate algorithm is still an open problem and depends on the different dataset or domain area. A major drawback of the already mentioned algorithms is their relatively low quality of classification. In most cases, these approaches don’t achieve the quality of classification that is provided by the operator.

As described above standard classifiers have a bias towards classes that have more texts and tend to predict the majority class. As a result, we gain misclassification of the minority class (in comparison with the majority class). That is way conventional approaches to classifying texts demonstrate rather bad performance when faced with imbalanced datasets. There are two effective ways how to operate with unbalanced sets [4].

First, we can change samples (data preprocessing). The goal is either to increase the number of documents in the minority class or to reduce the number of documents in the majority class. This is done to obtain approximately the same number of instances for both classes. The most popular techniques are random undersampling (randomly eliminating majority class text examples) and oversampling (increasing the number of text instances in the minority class first through simulation). Our experimental studies show that the disadvantages of these approaches outwit the advantages, and the quality of the classification changes only slightly.

Second, we can improve classifiers to make them appropriate for imbalanced data sets. These studies are conducted as part of the construction of ensemble classifiers to improve the performance of single classifiers by voting (and aggregating their predictions).

Many researchers have investigated the technique of combining the predictions of multiple classifiers to produce a single solution [5,6]. Nowadays the most popular and effective techniques of creating ensembles of classifiers are Random Forests, AdaBoost, and Gradient Boosting Machine – GBM [2,7,8].

In this article, we apply and explore Gradient Boosting Machine for the categorization of requests. The arguments in favor of usage GBM can be formulated in the following way:

- In comparison with other ensemble classifiers, GBM generally performs better than Random Forests and AdaBoost (although there is a price for that: GBM has a few hyper-parameters to tune, while Random Forest and AdaBoost are almost tuning-free). The possibility of tuning gives researchers more options to achieve better results on a particular dataset.
- The Kaggle competitions (the most popular testing ground of machine learning techniques) demonstrate high universality GBM and efficiency for a wide range of real applications (https://www.kaggle.com/).
- The availability of software implementations modifications of the method for solving various practical problems (for example, algorithm XGBoost - eXtreme Gradient Boost – the variant of classical GBM with some additional heuristics [9]).
- GBM meets the requirements for speed of categorization and provides parallelization of computations. So this method is computationally cheap and easy to implement.

This paper is organized as follows. In the first section, we consider pre-processing tasks (extracting text, tokenization, stop-word removal, normalization, named entity recognition) and give descriptions of two models of text representation (Bag of Words and Word2Vec). In the second section, we briefly explain Gradient Boosting Machine, setting its parameters on the simples and compare the quality of classification under using Bag of Words and Word2Vec.

2 Preprocessing and Feature Extraction

For the learner to compute a classification function, it needs to understand the document. For the learner, the document is merely a string of text. Hence, there is a need to represent the document text in a structured manner. The most common technique to represent text is the Bag-Of-Words (BOW) model [1]. In this technique, the text is broken down into words. Each word represents a feature. This process is also referred to as “Tokenization” since the document is broken down into tokens (individual words). A group of features extracted thus forms a feature vector for the document. Note that in such a model, the exact order of word occurrence is ignored. Since this vector becomes too large, there are several ways to prune this vector. Techniques like stop word removal and stemming are commonly applied. Stop word removal involves removing words that add no significant value to the document. However, when dealing with shorter text messages, traditional techniques will not perform as well as they would have performed on larger texts. This is primarily because there are few word occurrences and hence it is difficult to capture the semantics of such messages (the word occurrence is too small, they offer no sufficient knowledge about the text itself). That is why short text messages are harder to classify than the larger corpus of text.

The other model of text representation is Word2Vec [10]

Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space [10].

Word2vec has two main learning algorithms: CBOW (Continuous Bag of Words) and Skip-gram. CBOW is "a continuous bag of words" a model architecture that predicts the current word based on its surrounding context. The Skip-gram architecture works the other way: it uses the current word to predict the surrounding words. The user of Word2vec can switch and choose between the algorithms. The order of context words does not affect the result in any of these algorithms. Obtained at the output of the coordinate representation of the vectors of the words allow us to calculate the "semantic distance" between words. Word2vec technology makes its predictions based on the contextual proximity of these words. Since Word2vec tool is based on neural network training, it is necessary to use large datasets to train it to achieve its most efficient operation. This allows improving the quality of predictions.

To apply models, the original samples first must be processed specially, taking into account the peculiarities of the language to identify the most informative features (Fig.1).

This processing included:

- Removing punctuation marks and service characters using the Re library and regular expressions.
- Reducing words to lowercase.
- Splitting the text into separate tokens using"whitespace".
- Reduction of words to normal form. Individual tokens were normalized using the pymorphy2 library.
- The addition of the negative particle to the next word. The meaning of some requests depends on the negative particle and affects the meaning of the entire request. Therefore, it is important to save the negative essence.
- Removing stop words and template phrases (clichés). It was performed using two dictionaries. Experts compiled the first one manually. It contained a list of words

directly related to the automated system. The second dictionary was the standard dictionary from the NLTK library (Natural Language Toolkit).

- Named entities recognition for further linking.

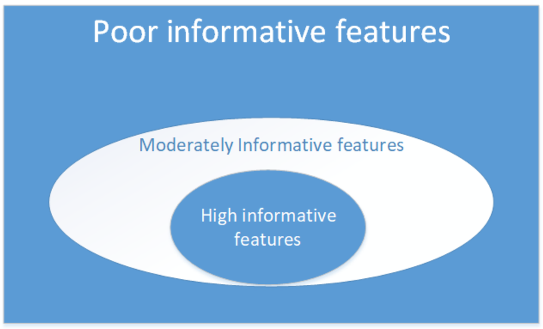
Poor informative features
Moderately Informative features
High informative features

Fig. 1. Informative features

The task of named entity recognition generally involves finding named entities in the text and determining their types, as well as linking a named entity to an object. The latter is important since identical words/phrases can refer to completely different objects. For example, the word “Lena” in Russian can be:

- The name of the person
- The name of the river
- The name of the town
- The name of the highway

The types of named entities in the work included:

- Dates - 07.02.2019, 28 November 2018, 1995-2003;
- Numbers – 1, 3.5, 547899.00;
- Addresses – Novosibirskaya street, 23;
- Time – 9:30, from 13:00 to 14:00;
- Currency units and money amounts - 5000 rubles, $10;
- Names of automated systems - SAP, CRM;
- Positions - Manager, engineer, senior analyst;

To recognize the named entities in the text, both the specificity of their entry and dictionary resources are used: dictionaries of names, geographical names, monetary units, occupations, etc. In this work, regular expressions are used to recognize named entities, in addition to special dictionaries supplemented with the subject of the problem.

It is well known that the quality of classification with GBM is significantly influenced by the dimension of vectors and terms used to describe documents. Therefore, it is important to select the best representation of the documents. In this paper, we conduct a comparative analysis of two approaches to feature extraction BoW and W2V. Based on our experiments, we identify an approach that provides a higher quality of classification with Gradient Boosting Machine. We make our conclusion based on analysis of values of precision and recall (as well as their combination – F1-measure).

3 Text Categorization with Gradient Boosting Machine

Classification is a supervised data mining technique that involves assigning a label to a set of unlabeled input objects. Based on the number of classes present, there are two types of classification: binary classification (classify texts into one of the two classes) and multi-class classification (classify texts into one of the multiple classes). In our work, we consider the more complicated case of multi-class classification. We need to build a classifier that can correctly assign a document to one class based on its content.

As noted earlier for the building of the classifier we use boosting technique which creates a highly accurate prediction rule by combining many weak and inaccurate rules. Each classifier is serially trained with the goal of correctly classifying texts in every round that were incorrectly classified in the previous round. After each round, it gives more focus to texts that are harder to classify. The quantity of focus is measured by a weight, which initially is equal for all instances. The most common representatives of the boosting family are Adaptive Boosting (AdaBoost), Gradient Boosting (GBM), and XGBoost (eXtreme Gradient Boost). In Gradient Boosting Machine the principle idea behind the algorithm is to construct the new base-learners to be maximally correlated with the negative gradient of the loss function, associated with the whole ensemble [11].

Gradient Boosting is an approach to approximate a function $\varphi^*: R^n \to R$ by a function φ of the form

$$\varphi(x) = \sum_{j=1}^{M} \alpha_j h_j(x) \quad (1)$$

where $\alpha_j \in$ R are real-valued weights, h_j: $R^n \to R$ are weak learners and x $\in R^n$ is the input vector [1].

Gradient Boosting can be seen as a generalization of AdaBoost. The latter is designed to minimize the exponential loss with stump weak learners h_j: $R^n \to \{+1, -1\}$, while Gradient Boosting can make use of real-valued weak learners and can minimize other loss functions [12]. Gradient Boosting has shown significant performance improvements in many classification problems concerning classic AdaBoost [13].

Let us discuss in more detail the description of the short text documents (requests for service) and training samples. The request that must be classified consists of two parts: “Information” – a brief description of the problem, written by the user and “Options” that may contain additional information about the request, for example, user’s department, system information, etc. As a result of combining these two parts, we obtain a document for classification.

To perform the experiments we downloaded and processed 9000 user requests. After that corpus was divided into two parts – training set and testing set – in the ratio of 67 and 33 percent respectively. Each class in the dataset has a different volume, which is shown in table 1.

To perform the best classification result we need to adjust the parameters of the Gradient Boosting Machine (http://xgboost.readthedocs.io). For some parameters were used the recommendations of the developers. But for the main parameters that have the greatest impact on the accuracy indicators, a special adjusting was carried out with the GridSearchCV module which performs an exhaustive search over specified parameter values for an estimator using k-fold cross-validation (k = 5). The next table shows parameters with the description, range, and selected value.

Table 1. Dataset volume.

Class name	The volume of the training set	The volume of a test set
C1	511	260
C2	563	295
C3	998	520
C4	3561	1770
C5	163	65

Table 2. Adjusted parameters.

Parameter	Description	Range	Selected value
eta	Step size shrinkage used in the update to prevents overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.	[0.1;0.3]	0.1
max_depth	Maximum depth of a tree, increase this value will make the model more complex/likely to be overfitting. 0 indicates no limit, limit is required for depth-wise grow policy.	[1,10]	3
objective	Specify the learning task and the corresponding learning objective	[binary:logistic, reg: logistic, binary:logitraw, reg:linear]	"binary: logistic"
booster	Which booster to use, can be gbtree, gblinear, or dart. gbtree and dart use a tree-based model while gblinear uses a linear function.	[gbtree, gblinear, dart]	gbtree
eval_metric	Evaluation metrics for validation data, a default metric will be assigned according to objective	The multiclass classification error rate	merror
colsample_bytree	Subsample ratio of columns when constructing each tree	[0,1]	0.7

To perform feature extraction first we break sentences into words. On this step, we had a deal with 469880 words and after preprocessing our training corpus contained 230905 informative words which we gave to the input of BoW and W2V.

To perform word extraction with Word2vec using CBoW algorithm we applied the following parameters:

- Learning algorithm - Hierarchical softmax.
- The minimum number of repetitions of the word in all sentences -min_word_count = 40.
- The dimension of the resulting space - num_features = 300 (default value).
- How many words "around" a particular word should be considered -
- Context = 10 (Normally used value).
- Parameter «sampling» makes the model choose very-frequent words less often in the training process - sampling = 1e-3 (default value).

Evaluation of a classification algorithm performance is measured by the Confusion Matrix which contains information about the actual and the predicted class.

Table 3. Confusion matrix

Actual	**Predicted**	
	Positive class	**Negative class**
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

$$F1 - score = \frac{2(Presicion*Recall)}{(Presicion+Recall)} \tag{4}$$

The results of classification with GBM and two methods of feature extraction are presented in tables 4 and 52. The results are obtained on the testing set.

Table 4. Prediction with GBM and Bag of Words

Class	Precision	Recall	f1-score
C1	0,95	0,86	0,90
C2	0,96	0,91	0,93
C3	0,92	0,93	0,93
C4	0,96	0,98	0,97
C5	0,76	0,79	0,78

Table 5. Prediction with GBM and Word2Vec

Class	Precision	Recall	f1-score
C1	0,92	0,89	0,90
C2	0,91	0,92	0,89
C3	0,89	0,90	0,90
C4	0,95	0,97	0,93
C5	0,88	0,86	0,85

As we can see prediction with GBM using Bag of Words provides a slightly better result than Word2vec but it is still not good enough to replace a human operator. According to the experience of the Bank human-operators perform such task approximately with f-score = 0.92 - so our goal is to perform an average f-score of at least 0.93 [14]. Small classes are rather poorly identified with the model, but the results on the biggest C4 let us expect good results if we will train our model on an extended corpus.

Although that BoW model provides better results on average it also performs the worst result in C5 category. Results of W2V model are smoother than the results of BoW model so we could note that W2V model is more resistant to the size of the dictionary.

4 Conclusions

Currently, the models and methods considered in the article are tested in one of the branches of a large bank in Russia. During testing, we focused on the analysis of the causes of errors. The main conclusion is that most of the errors are due to incorrect filling of requests, in particular, the use of slang, abbreviations that correspond to the vocabulary of modern messengers and social networks. To improve the prediction quality, we plan to achieve through the following:

- as the number of requests increases to expand the training set to achieve the more precise tuning of the parameters for Word2vec, and also make our sample balanced by extending small classes and discarding some objects of large classes;
- try W2V skip-gram algorithm. It is more computationally expensive than CBoW but performs better results in models using large corpora and a high number of dimensions [10];
- perform more detailed customizing of XGBoost model on an extended corpus;
- develop formal rules and guidelines for filling service requests.

At the same time, the conducted testing and discussion of its results with the Bank's specialists allows identifying several similar processes that can be automated based on Text Mining techniques.

References

1. Manning C.D., Raghavan P., Schutze H.: Introduction to Information Retrieval. – Cambridge University Press, (2008)
2. Bobryakov A, Kuryliov M, Mokhov A, Stefantsov A.: Approaches to automation processing of user requests in a multi-level support service using featured models. Proceedings of the 30th DAAAM International Symposium, pp. 0936-0944, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-22-8, ISSN 1726-9679, Vienna, Austria. DOI: 10.2507/30th.daaam.proceedings.130 (2019)
3. Batura T.V. Metodi avtomaticheskoy klassifikacii tekstov [Methods for automatic classification of texts]// Software & Systems. №1(30), pp. 85-89. DOI: 10.15827/0236-235X.117.085-099 (2017)
4. Krawczyk B.: Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence 5 (4), pp 221–232 DOI: 10.1007/s13748-016-0094-0 (2016).
5. Onan A.,Korukoglu S., Bulut H.: Ensemble of keyword extraction methods and classifiers in text classification. In: Expert Systems with Applications. 57 (15), pp. 232-247. DOI: 10.1016/j.eswa.2016.03.045 (2016).
6. Daroczy B., Siklois D., Palovics R., Benczur A.: Text Classification Kernels for Quality Prediction over the C3 Data Set. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1441-1446. DOI: 10.1145/2740908.2778847 (2015).
7. Friedman J.: Greedy boosting approximation: a gradient boosting machine. In: Annals of statistics. 29, 1189–1232, (2001)
8. Friedman J., Hastie T., Tibshirani R.: Additive logistic regression: a statistical view of boosting. Annals of statistics. 28, 337–407, (2000)
9. Chen T., Guestrin C.: XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785-794. DOI: 10.1145/2939672.2939785 (2016).
10. Mikolov T, Chen K, Corrado G, Dean J: Efficient Estimation of Word Representations in Vector Space. In: Proceedings of Workshop at ICLR, (2013) (arXiv:1301.3781).
11. Natekin A and Knoll A.: Gradient boosting machines, a tutorial. In: Front. Neurorobot. 7 (21). DOI: 10.3389/fnbot.2013.00021 (2013).
12. Becker C., Rigamonti R.: Kernel Boost: Supervised Learning of Image Features For Classification. School of Computer and Communication Sciences Swiss Federal Institute of Technology, Lausanne. Technical Report. 2013, p.1-19.
13. Hastie T., Tibshirani R., and Friedman J.: The Elements of Statistical Learning. Springer (2001).
14. Bolshakova E., Vorontsov K., Efremova N. et al. Avtomaticheskaya obrabotka tekstov na estestvennom yazike I analiz dannikh. [Automatic processing of texts in a foreign language and data analysis] // NRU "Higher school of economics", ISBN 978–5–9909752-1-7 (2017) (In Russ.).