# IMapbook: Automating Analysis of Group Discussions

**Patrik Kojanec**
pk0404@student.uni-lj.si

**Marko Rus**
mr5613@student.uni-lj.si

## 1 Introduction

Nature Language Processing has seen a huge rise in popularity in recent years. It is now broadly studied topic with many successful applications. In this project we touch subfield Text Classification and apply its methods to the data from IMapbook (Gill and Smith, 2013), a web-based technology that allows reading material to be intermingled with interactive games and discussions. Some portion of discussions from this platform were manually annotated, each reply was given more categories based on the information in the reply. Our goal is to take this data and try to build a classifier which would predict these categories. Such classifier could then be used to automate analysis of discussions at this platform, recommend the time for the teacher's intervention and more.

### 1.1 Related work

The domain of our problem is short-text classification, which is closely related to social media. Unlike the common text classification problems, where the documents are usually long and written in formal language, it deals with texts of few sentences, written in informal language. The amount of context information carried in the texts is usually very low, thus classification and information retrieval become challenging tasks to perform efficiently (Song et al., 2014). Furthermore, the low co-occurrence of words induced by the shortness of the texts often results problematic for machine learning algorithms, which rely on word frequency.
With the arise of social media this branch of text classification became a well researched problem, and people tried different approaches to overcome its constraints. In a survey in 2014, (Song et al., 2014) pointed out the main methods of short text classification, which mostly relate on semantic analysis, since it pays more attention to the concept, inner structure semantic level, and the correlation of texts to obtain the logic structure, which is more expressive and objective. Currently, the most widely used vector representations of words (or embeddings), that proved to capture well the semantic information are GloVE (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013).
Although standard machine learning approaches often resulted problematic with short text, Sriram et al. (Sriram et al., 2010) showed that their model with hand-crafted features, related to user's tweets[1], efficiently filtered irrelevant tweets from the users, thus suggesting that by adding extra sources of context information increases the performance. Similarly, this concept was also recently shown by Yang et al. (Yang et al., 2018). Furthermore, they have also shown that Support Vector Machines performed almost equally well in classification when using word embeddings or TF-IDF, but they were outperformed by deep neural networks.

## 2 Dataset

The dataset is provided by IMapBook and includes the discussions between students and teachers on the topics of the book they are reading. The dataset includes approximately 3500 Slovene messages, from 9 different schools and on 7 different books, which were also translated to English. Students in each school were divided in "book clubs", where the conversations occurred.
The data was manually annotated, with three main tags:

- *Book Relevance*: Whether the content of the message is relevant to the topic of the book discussion.

[1] Short text message on the Twitter platform (www.twitter.com).

- *Type*: Whether the message is a question (Q), answer (A) or a statement (S). In original data mixture of these classes also appear (QA and AQ), but because of their low frequency (together they appear only three times in entire dataset), we changed QA occurrences to Q and AQ to A.

- *Category*: Whether the message is a simple chat message (C), related to the book discussion (D), moderating the discussion (M), wondering about users' identities (I), referring to a task, switching it or referring to a particular position in the application (S), or other cases (O).

The *Category* category can be further on split in sub-categories; *chats* may be in the form of greetings (G), related to the book (B), they could be encouraging (E), talk about feelings (F), contain cursing (C) or others (O), *Discussion* messages could be questions (Q), answers (A), answers to users, still related to the discussion topic (AA) or encouraging the discussion (E); *identity* messages can be answers(A), questions (Q) or their combination (QA).
The dataset is suitable for both binary and multi-class classification, whether the target variable is the relevance or the category of the message respectively.

## 3 Methods

In this section we present the methods that will be used to perform three different message classification tasks:

1. Book relevance classification (binary)

2. Type of message classification (3-class)

3. Broad category classification (6-class)

Input data to all classifiers (except for Majority Classifier) are questions and answers in Slovenian.

### 3.1 Baseline

As a baseline model we decided to use Majority Classifier. In each task it classifies every instance as the most representative class in training set.

### 3.2 Hand-Crafted Feature Models

The first group of models that we present is based on a hand-crafted feature set. These features were then used as an input to different classification algorithms, that we list in Section 3.2.2. We describe features extraction in the next section.

#### 3.2.1 Features Extraction

The aim of the features was to simply and intuitively capture the relevance to the question, while filtering gibberish and inappropriate messages. Thus, the following set of features was designed:

- Number of tokens in a message.

- Number of mistakes in a message; this was computed by matching words with the words in a lexicon (Dobrovoljc et al., 2019).

- Maximal length of the token in the message.

- Number of characters in a message.

- Number of question marks in a message.

- Number of exclamation points in a message.

- Number of commas in a message.

- Number of periods in a message.

- Number of capital letters in a message.

- Number of capital letters within the interior of the words in a message.

- Number of peculiar characters in a message.

- Number of numbers within the interior of the words in a message.

- *Levenshtein distance:* Number of all pairs of words from the question and the message, whose Levenshtein distance is less than half the length of the longest of the two words.

- Number of interrogative words in a message.

- Number of "kdo" in a message.

In the case of *Levenshtein distance* feature, the messages were initially tokenized and stop-words (Bučar, 2017) were removed, while for other cases regular expressions were used to extract the features.

All features were designed while looking at the data, having some sense in how the feature could increase the classification success. For instance, many messages had "kdo" word in it, asking for identity of somebody. Those messages have the same class. But nevertheless we observed only small portion of the data, so that chosen features would not be overfitted.

### 3.2.2 Classification Algorithms

We decided to feed the features to four different classification algorithms to see how they perform. We chose a naîve bayesian (NB), random forest (RF), support vector machine (SVM) and a logistic regression (LR) classifiers. We used the implementations from scikit-learn library (Pedregosa et al., 2011).

When selecting the parameters we observed train and test accuracy and paid closed attention to detecting overfitting. For NB we left the default parameters. For the SVM we used the RBF kernel and set the parameter *gamma* to "auto" and *C* to 5, while for the LR we decided to use "lbfgs" optimizer with maximum 1000 iterations. In the case of LR the input data was standardized to ensure equal class importances. For the RF we set the number of estimators to 150, while *min_samples_leaf* to 3 and *min_samples_split* to 10. This way we managed to reduce the overfitting to the training data. We kept the same parameters for all the tasks.

## 4 Evaluation

We considered more metrics for evaluation of models. Precision, recall, F1, AUC ROC metrics are defined in principle only for binary problems, at multiclass problems they loose some interpretability, so we decided not to use them. By changing the models to return probabilities, we were able to apply log loss as a metric. But the problem was that some probabilities of correct class were equal or near 0, which induced instability to the score.

So we decided to use the classification accuracy as an evaluation metric. We used 5-fold cross validation on the whole dataset, where our performance estimator was the average result of the five test sets. We also measured uncertainty, to have an opinion about variance of the estimator.

## 5 Results

Accuracies for all models and targets are shown in Figure 1. We notice that all feature-based classifiers outperform the Majority Classifier. Furthermore, as expecting, the classification accuracy drops with increasing number of target classes. The best performing classification algorithm on this dataset seems to be Random Forest, which outperformed the others in both "CategoryBroad" and "Type" classification tasks. Its performance on the "Book Relevance" task was also on average higher than the rest, however SVM and LR obtained comparable results. Surprisingly, LR with multi-class extension performed quite well also on multi-class classification tasks.

Initially, RF yielded very high performance on the training set, reaching a 95% accuracy. However, the performance on the test set was lower, showing signs of overfitting. Thus, with a more careful selection of the parameters, we dropped the training accuracy for about 10% and reached the current test performance.

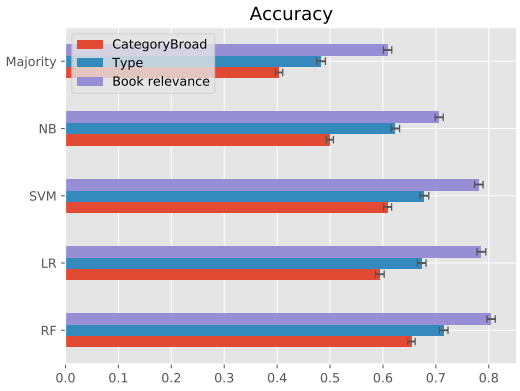


Figure 1: **Accuracies.** Classification accuracies of models on three different classification tasks described in Section 2. Random Forest achieves 80.4%, 71.5% and 65.4% accuracies on "Book relevance", "Type" and "CategoryBroad" targets respectively.

### 5.1 Features Importance

RF is often used as a features selection tool, as it ranks the importance of the features. In Figure 2 we show the importance of each feature in the decision process of the RF model.

As we notice, each classification task focuses on different features, however there are some common ones that are discriminatory for all three tasks, i.e. last five in the plot. As expected, *Lev.*

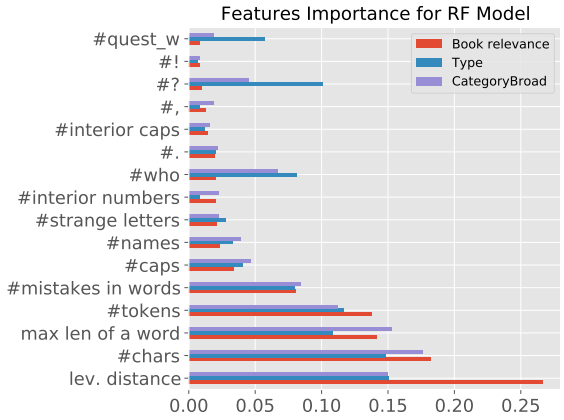


Figure 2: **Features Importance.** *Lev. distance* between answer and question, general length of message and number of mistakes show as important features.

*distance* works particularly well on the "Book relevance" problem, since it performs a naive kind of matching of the text messages with the questions. However, it results also as the most discriminatory feature for "Type" classification and third for "CategoryBroad" classification.

It is not surprising that some features are particularly relevant to some classification tasks, since they were designed for that purpose. It is also known that good features increase performance. Here we showed that some features are particularly suitable for some specific tasks, while others behave well over different classification problems. One future improvement that could be done is trying to define some other features that would boost the performance, removing the irrelevant ones.

## 6 Future Perspective

Random forest seems to be a very powerful classification algorithm that fits well on the given data. However, features given to it were only simple properties of the messages, extracted by hand.

What we would like to try is to create embeddings not by hand, but with some more sophisticated model, as Neural Networks. Many pretrained Neural Networks exist (BERT (Devlin et al., 2018)), which can be used to extract contextual embeddings.

## References


Jože Bučar. 2017. Automatically sentiment annotated slovenian news corpus AutoSentiNews 1.0. Slovenian language resource repository CLARIN.SI.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kaja Dobrovoljc, Simon Krek, Peter Holozan, Tomaž Erjavec, Miro Romih, Špela Arhar Holdt, Jaka Čibej, Luka Krsnik, and Marko Robnik-Šikonja. 2019. Morphological lexicon sloleks 2.0. Slovenian language resource repository CLARIN.SI.

Grandon Gill and Glenn Gordon Smith. 2013. Imapbook: Engaging young readers with games. *Journal of Information Technology Education: Discussion Cases*, 2(1).

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christoper Manning. 2014. Glove: Global vectors for word representation. volume 14, pages 1532–1543.

Ge Song, Yunming Ye, Xiaolin Du, Xiaohui Huang, and Shifu Bie. 2014. Short text classification: A survey. *Journal of Multimedia*, 9.

Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. 2010. Short text classification in twitter to improve information filtering. *Proceedings of the 33rd international ACM SIGIR conference on Research\r\nand development in information retrieval*, pages 841–842.

Xiao Yang, Craig Macdonald, and Iadh Ounis. 2018. Using word embeddings in Twitter election classification. *Information Retrieval Journal*, 21(2-3):183–207.