IMapbook: Automating Analysis of Group Discussions

Patrik Kojanec
pk0404@student.uni-lj.si
Univerza v Ljubljani,
Večna pot 113,
1000 Ljubljana

Marko Rus
mr5613@student.uni-lj.si
Univerza v Ljubljani,
Večna pot 113,
1000 Ljubljana

Abstract

The constant advancements in technology introduce new approaches and tools for students' education. Such tool is the ImapBook, an interactive platform that allows students to communicate and discuss about the book they are reading. To give the best support, such applications need natural language processing features, that understand the content of communications and automatically intervene when the focus on the topic is lost. Thus, we present different text classification models that classify such texts from students' communications into useful categories. The best performing method is fine-tuned BERT, which outperforms simpler classification methods based on handcrafted features.

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

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

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.
- *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 are exchanged messages. To provide information about wether users are discussing about relevant topic, each message also has information about the question provided to users before the discussion.

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 importance. 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.

3.3 ELMo Embeddings

We handcrafted features by looking at the messages and observed what could potentially discriminate different types of messages. For the next experiment we wanted to know, how good features can we extract automatically, so that such human interaction and understanding of messages wouldn't be necessary.

ELMo (Peters et al., 2018) is model for creating contextual embeddings. We have chosen it as it can also be used to embed entire message. Firstly we put discussion topic into it, and then message, so that message's embedding also contains information about the relevance to the topic.

We have used pretrained ELMo model for Slovene language (Ulčar, 2019).

For classification we tried all models discussed in 3.2.2 and also KNN (Fukunaga and Narendra, 1975) with cosine distance, as it is natural distance to use in ELMo embeddings. Random Forest classifier ended up having the highest performance.

3.3.1 Fixing Typos in Messages

Messages in the input data contain a lot of words, that have typos in them and are not part of the Slovene lexicon (Dobrovoljc et al., 2019). Also, a lot of mistakes come from users deliberately leaving out carrot (e.g. 's' instead of 'š'). That is why we decided to write an algorithm for correcting typos that are away from the correct word for at most Levenshtein distance of two. We also calculated probabilities of the words and removed words with probability less than 10^{-8}.

3.4 BERT Fine-Tuning

An other approach we propose is using a pretrained BERT (Devlin et al., 2018), by fine-tuning it for our classification tasks. We avoided customizing BERT models, because they require notorious amount of data which was not available. We trained our BERT model for Slovenian, Croatian and English languages for sequence classification for three epochs on training data that consisted of 80% of our dataset, while the remaining 20% was left for testing. Out of these 80%, 15% were used for validation. We trained one model for each task, for both Slovenian and English-translated messages.

4 Evaluation

We evaluated the models using F1 evaluation metric. At multi-class problems we used weighting over different classes to compute it.

Because of the complexity of our models, we opted for two different evaluation techniques: on models that are not so computationally expensive to fit, we used 5-fold cross validation on the whole dataset, where our performance estimator was the average result of the five test sets. This technique also points out the variance of our estimator, hence quantifying to some extend the uncertainty of the performances of our models. The second evaluation is a simple hold-out evaluation, where we split train and test sets at 80%, thus losing information about the variability of the performance of our predictor.

5 Results and Discussion

5.1 Baseline Models

Scores for computationally less expensive models (Majority Classifier and different models with handcrafted features) 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 is 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.

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.

5.1.1 Features Importance

RF is often used as a features selection tool, as it ranks the importance of the features. Features used at the top of the tree contribute to the final prediction decision of a larger fraction of the input samples. The expected fraction of the samples they contribute to can thus be used as an estimate of the relative importance of the features. I In Figure 2 we show the importance of each feature in the decision process of the RF model.

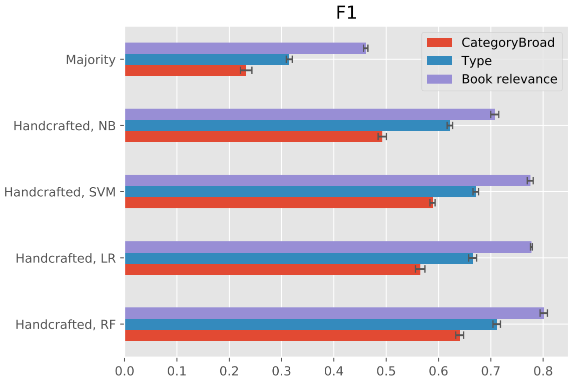

Figure 1: **F1 scores.** Scores of baseline models on three different classification tasks described in Section 2.

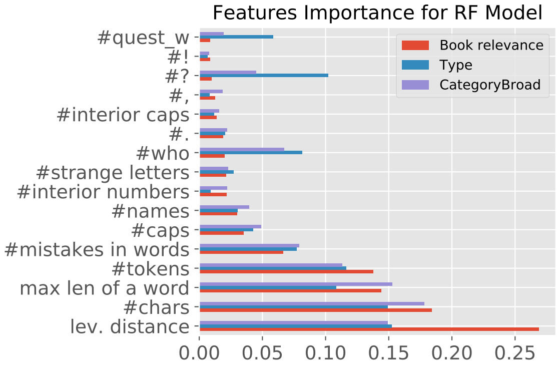

Figure 2: **Features Importance.** *Lev. distance* between answer and question, general length of message, and number of mistakes show as important features.

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. distance* works particularly well on the "Book relevance" problem, since it performs a naïve 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.

5.2 Deep Models

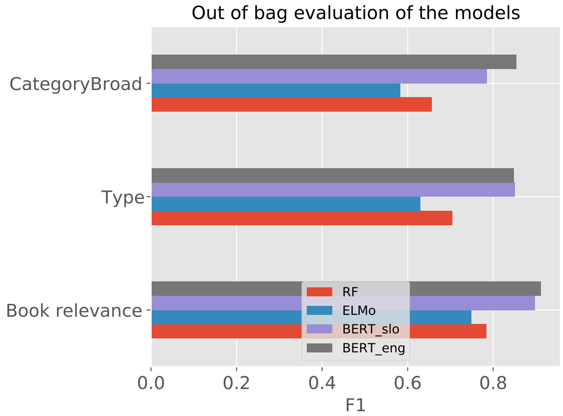

Figure 3: **Hold-out performance evaluation.** Comparing performances on the test set of BERT, Handcrafted Features Model and ELMo model.

F1 scores from hold-out evaluation can be seen in the Figure 3 and in the following table.

	Relevance	Type	Categ.
Handcrafted	0.78	0.70	0.66
ELMo	0.75	0.63	0.58
BERT	0.90	**0.85**	0.78
BERT (Eng)	**0.91**	0.85	**0.85**

Here BERT (Eng) is BERT trained on English translations. Note that these translations were made by human and wouldn't be present in unseen data.

5.2.1 ELMo

We can see that ELMo has worse performance than baseline model with handcrafted features. But it is important to note here, that handcrafted features may be overfitted to the given data. If model was applied to discussions from older children, same features may perform worse. In the other hand, ELMo features are generated automatically and may generalize better.

5.2.2 BERT

When analyzing performance of the BERT models, we can clearly see 15 - 20% increase in performance compared to model with handcrafted features. BERT model that uses English translations is even more successful, especially in the classification of the category, where we can observe nearly 29% increase in performance. This clearly demonstrates dominance of BERT models.

We would like to mention, that here we did not measure uncertainty of the scores. But scores are still comparable, as we evaluated models on the same test set.

5.3 Analyzing Predictions

A lot of messages are asking for identity of somebody, and such messages were mostly successfully classified by all models. Lots of messages contain a lot of gibberish and are as such distinguishable from other messages. Harder to predict are messages that are short and contain only few words. Models performed worse also on messages with a lot of unidentified mistakes in words.

6 Conclusion

The main question that we were answering was to what extent can messages be automatically recognized as of certain type. We showed that you can achieve decent performance with handcrafted features and simple models, as also with fully automatic generation of features. Furthermore, we fine-tuned end-to-end BERT neural network, yielding a significant increase in performance.

6.1 Further Work

Messages typically have a flow, messages closer together are more probable to have the same type. In our approach we discarded this information about the time the message was sent, which resulted in the loss of information. One possible improvement to our models would be to somehow incorporate this information to the models, to identify messages that are a direct reply to some other message. This would improve performance at short messages, which are without such context rarely meaningful.

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.

Keinosuke Fukunaga and Patrenahalli M. Narendra. 1975. A branch and bound algorithm for computing k-nearest neighbors. *IEEE transactions on computers*, 100(7):750–753.

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.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

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.

Matej Ulčar. 2019. ELMo embeddings models for seven languages. Slovenian language resource repository CLARIN.SI.

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