Natural Language Processing 2020/21
Project Report

University *of Ljubljana*
Faculty *of Computer and Information Science*

IMapBook Collaborative Discussion Classification

Uroš Polanc

Abstract

Automatic text classification has been considered as a vital method to manage and process a vast amount of documents in digital form, which is widespread and continuously increasing. That brought a wave of new tools for student education, such as the IMapBook platform. It gives students a way to communicate and discuss the books they are reading. In our assignment, we will try to classify those discussions based on the book it's relevant to, and the type of message. We tested multiple classifiers and models, as well as different feature extractors, such as TF-IDF, CountVec, and our handcrafted features. The results have shown that out of the classifiers Logistic Regression performs the best with a peak accuracy of 82%, meanwhile overall the best performance was achieved by BERT at a 92% accuracy. For the extractors, CountVec performed best in most cases, but when it came to text classification based on message types our handcrafted features achieved better results.

Keywords
IMapBook, Text Classification, TF-IDF, Handcrafted Features, BERT, Logistic Regression

Advisors : Slavko Žitnik

Introduction

With the rapid development of the internet and big data, digital information is increasing at a high rate. With this technologies and applications such as IMapBook [1], a web-based application that allows for discussions on reading materials as well as interactive games, also started emerging.

In our assignment, we looked at some IMapBook collaborative discussions on different books. For that, we need to look at some vital information, that we extracted from the discussions and the book texts. We will create three different message classifications, based on:

- Book ID (3 classes),
- CodePreliminary (9 or 16 classes),
- Topic (3 classes).

From the message types, we can say that not all are important to the discussions of the books, and only the content discussion and content question are directly related to the books. With this, we were also provided with a (1) dataset that contains all message types and a (2) dataset that contains only massages important to the discussion. With this, we will be able to further compare how much noise impacts the classifiers and prediction. An important note here is that there are 16 different classes of CodePreliminary in the first dataset, while only 9 in the second, due to filtering out certain classes.

By checking the manual classifications in the dataset, we noticed some were misspelled and some were incorrectly written. Therefore, we applied some fixes to the manual classifications. The errors such as excess whitespace, lower and upper case, etc... were removed. Classifications where there were multiple classes, were simplified into one class, to simplify the classification process. The final fixed distribution is not intraclass uniform as can be seen in figures 1 and 2.

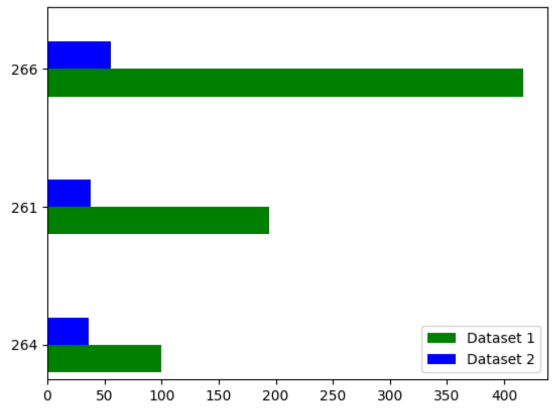


Figure 1. The distribution of book IDs in the dataset.

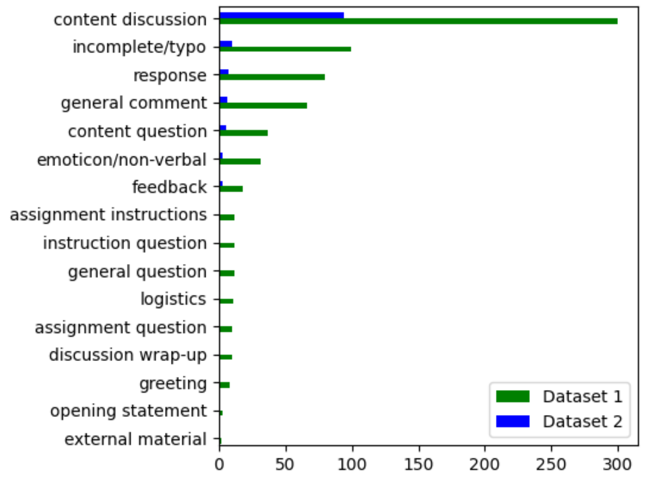


Figure 2. The distribution of message types in the dataset.

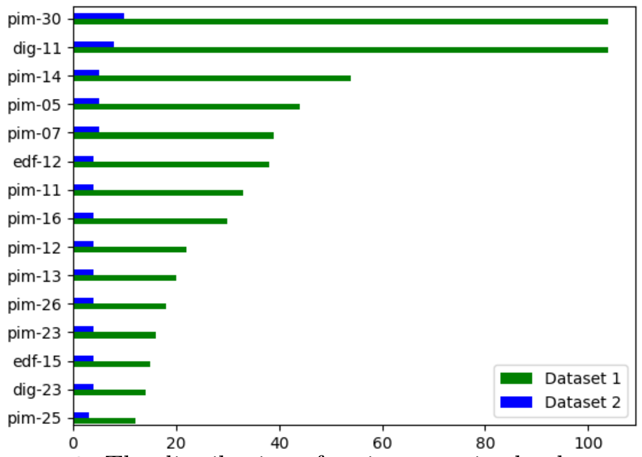
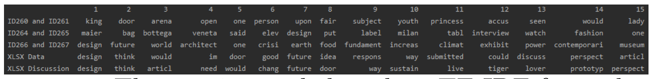


Figure 3. The distribution of active users in the dataset.

From all the graphs above we can see that in dataset 2, where there are messages concerning the discussion, we get a more uniform distribution of users as well as books. While the distribution of message types is dominated by the content discussion.

To better understand the contents we are trying to classify, we checked the top unigrams of the book texts. We also calculated the TF-IDF of the books as well as the collaborative discussions (as a whole) and displayed the top 15 stemmed words for each (see figure 4). This gives us a quick insight into the most important words of each book.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ID260 and ID261	king	door	arena	open	one	person	upon	fair	subject	youth	princess	accus	seen	would	lady
ID264 and ID265	maier	bag	bottega	veneta	said	elev	design	put	label	milan	tabl	interview	watch	fashion	one
ID266 and ID267	design	future	world	architect	one	crisi	earth	food	fundament	increas	climat	exhibit	power	contemporari	museum
XLSX Data	design	think	would	im	door	good	future	idea	respons	way	submitted	could	discuss	perspect	articl
XLSX Discussion	design	think	articl	need	would	chang	future	door	way	sustain	live	tiger	lover	prototyp	perspect

Figure 4. The top 15 words based on TF-IDF for each set.

From this we can say that the texts are contextually different, aside from a few words such as ’one’, the top frequent words are more or less unique to their books.

Related works

Our work is closely related to short text classification, and with the explosive growth of social media, online communication, short text classification has become an important topic in recent years. Examples of short text can be found in multiple places, such as tweets, chat messages, search queries, product descriptions, or online reviews. Although the applications are wide-ranging, short text classification can be a hard and challenging task.

Short texts present several challenges associated with the lack of semantic and syntaxes as well as data sparseness. To overcome these obstacles, several solutions have been proposed to improve text representation, in other words, to enrich the semantics by using implicit or explicit information, specific classification algorithms.

In the case of text representation, many authors have used different techniques to translate the text into a number format. The most common method is Bag of Words (BoW), where each text is represented as a vector, in which each component represents whether a word in the whole corpus vocabulary appears in the short text or not. This approach does not take into account the word order and produces sparse vectors that difficult the use of classification algorithms. To overcome this issue, Neural Language Models (NLM) [2] have been used to learn word representations in a semantic vector space. Some of the most common word embeddings models are the Word2vec [3] and Glove [4], which have been used as input for several classification approaches. Authors such as Chavaltada et al. [5], Pandey et al. [6], Wang and Manning [7] used BoW representation in classification techniques such as Naive Bayes, Logistic Regression and Support Vector Machines to build a text classifier, whereas Huang et al. [8] and Zhu et al. [9] used pre-trained word embeddings with Logistic Regression and K-Nearest Neighbor techniques, respectively.

Methods

For feature extraction, we used two known methods TF-IDF and CountVec, as well as our handcrafted features. For classifiers, we decided to test Naive Bayes (NB), Random Forest (RF), Logistic Regression (LR), K-Nearest Neighbors (KNN), as well as Majority Voting (MV). Now, for Majority Voting we decided that it will use LR, NB, and RF classifiers as the voters.

Text preprocessing

As we mentioned when we used TF-IDF and CountVec extractors, we preprocessed the text. This includes the following steps:

- remove punctuation,
- remove special characters,
- remove single characters,
- remove digits,
- remove any excess whitespace characters,
- transform to lowercase,
- remove stopwords,
- stem any non named entity word.

Handcrafted features

Unlike with the other two handcrafted features, we will not preprocess the text beforehand, meaning we will keep it as it is.

For our features we simply processed each message and created a vector of numbers, where the numbers in order represent:

- number of sentences,
- number of words,
- number of non-white characters,
- number of unique words,
- number of digits,
- number of stop words,
- number of interrogative words,
- the maximum word length,
- the average word length,
- the number of exclamation marks '!',
- the number of question marks '?',
- the number of commas ',',
- the number of dots '.',
- the number of dashes '-',
- the number of closing brackets ')',
- the number of opening brackets '(',
- the number of colons ':',
- the number of capital letters,
- the number of non-ascii characters.

Results

We then tested classifiers such as Random Forest (RF), Naive Bayes (NB), Logistic Regression (LR), K-Nearest Neighbors (KNN), Majority Voting (MV) with extractors such as CountVec, TF-IDF, and our handcrafted features. To this, we also added a pre-trained BERT model, with no external preprocessing.

Table 1. $CodePreliminary$ with $tfidf$ on dataset 1.

Model	Extractor	F1 Score	Accuracy
LR	**countvec**	**0.52 ± 0.07**	**0.58 ± 0.06**
NB	countvec	0.45 ± 0.08	0.52 ± 0.07
LR	handcrafted	0.44 ± 0.07	0.51 ± 0.06
RF	handcrafted	0.42 ± 0.07	0.51 ± 0.06
MV	handcrafted	0.42 ± 0.07	0.51 ± 0.06
LR	tfidf	0.39 ± 0.07	0.51 ± 0.06
RF	tfidf	0.39 ± 0.06	0.50 ± 0.05
MV	tfidf	0.39 ± 0.06	0.50 ± 0.05
KNN	handcrafted	0.39 ± 0.07	0.48 ± 0.07
NB	tfidf	0.34 ± 0.07	0.47 ± 0.06
RF	countvec	0.34 ± 0.07	0.47 ± 0.06
MV	countvec	0.34 ± 0.07	0.47 ± 0.06
NB	handcrafted	0.42 ± 0.06	0.46 ± 0.05
BERT	none	0.41 ± 0.21	0.42 ± 0.21
KNN	countvec	0.15 ± 0.07	0.13 ± 0.05
KNN	tfidf	0.07 ± 0.03	0.07 ± 0.03

Table 2. $Topic$ with $tfidf$ on dataset 1.

Model	Extractor	F1 Score	Accuracy
BERT	**none**	**0.93 ± 0.07**	**0.92 ± 0.08**
LR	countvec	0.77 ± 0.07	0.79 ± 0.05
NB	countvec	0.76 ± 0.06	0.77 ± 0.06
NB	tfidf	0.70 ± 0.06	0.75 ± 0.05
LR	tfidf	0.70 ± 0.05	0.74 ± 0.04
RF	tfidf	0.66 ± 0.07	0.71 ± 0.06
MV	tfidf	0.66 ± 0.07	0.71 ± 0.06
RF	countvec	0.64 ± 0.06	0.71 ± 0.05
MV	countvec	0.64 ± 0.06	0.71 ± 0.05
RF	handcrafted	0.57 ± 0.04	0.63 ± 0.04
MV	handcrafted	0.57 ± 0.04	0.63 ± 0.04
LR	handcrafted	0.55 ± 0.06	0.61 ± 0.06
KNN	countvec	0.50 ± 0.06	0.61 ± 0.05
KNN	tfidf	0.45 ± 0.05	0.59 ± 0.04
NB	handcrafted	0.56 ± 0.05	0.58 ± 0.04
KNN	handcrafted	0.50 ± 0.04	0.56 ± 0.04

Table 3. *BookID* with *tfidf* on dataset 1.

Model	Extractor	F1 Score	Accuracy
BERT	**none**	**0.92 ± 0.07**	**0.92 ± 0.07**
LR	countvec	0.77 ± 0.07	0.79 ± 0.05
NB	countvec	0.76 ± 0.06	0.77 ± 0.06
NB	tfidf	0.70 ± 0.06	0.75 ± 0.05
LR	tfidf	0.70 ± 0.05	0.74 ± 0.04
RF	tfidf	0.66 ± 0.07	0.71 ± 0.06
MV	tfidf	0.66 ± 0.07	0.71 ± 0.06
RF	countvec	0.64 ± 0.06	0.71 ± 0.05
MV	countvec	0.64 ± 0.06	0.71 ± 0.05
RF	handcrafted	0.57 ± 0.04	0.63 ± 0.04
MV	handcrafted	0.57 ± 0.04	0.63 ± 0.04
KNN	countvec	0.51 ± 0.06	0.62 ± 0.05
LR	handcrafted	0.55 ± 0.06	0.61 ± 0.05
KNN	tfidf	0.46 ± 0.04	0.59 ± 0.04
NB	handcrafted	0.56 ± 0.05	0.58 ± 0.04
KNN	handcrafted	0.50 ± 0.05	0.54 ± 0.05

Table 5. *Topic* with *tfidf* on dataset 2.

Model	Extractor	F1 Score	Accuracy
LR	**countvec**	**0.82 ± 0.11**	**0.82 ± 0.10**
NB	countvec	0.81 ± 0.11	0.82 ± 0.11
LR	tfidf	0.80 ± 0.15	0.81 ± 0.13
BERT	none	0.79 ± 0.09	0.79 ± 0.07
NB	tfidf	0.77 ± 0.14	0.78 ± 0.13
RF	tfidf	0.71 ± 0.14	0.72 ± 0.14
MV	tfidf	0.71 ± 0.14	0.72 ± 0.14
RF	countvec	0.71 ± 0.14	0.72 ± 0.14
MV	countvec	0.71 ± 0.14	0.72 ± 0.14
KNN	tfidf	0.69 ± 0.18	0.70 ± 0.17
LR	handcrafted	0.55 ± 0.15	0.58 ± 0.13
RF	handcrafted	0.55 ± 0.14	0.57 ± 0.12
MV	handcrafted	0.55 ± 0.14	0.57 ± 0.12
NB	handcrafted	0.52 ± 0.15	0.52 ± 0.13
KNN	handcrafted	0.45 ± 0.16	0.46 ± 0.18
KNN	countvec	0.39 ± 0.17	0.45 ± 0.17

Table 4. *CodePreliminary* with *tfidf* on dataset 2.

Model	Extractor	F1 Score	Accuracy
NB	**handcrafted**	**0.81 ± 0.11**	**0.82 ± 0.12**
LR	countvec	0.79 ± 0.08	0.82 ± 0.10
LR	handcrafted	0.82 ± 0.09	0.82 ± 0.09
RF	handcrafted	0.79 ± 0.08	0.82 ± 0.07
MV	handcrafted	0.79 ± 0.08	0.82 ± 0.07
KNN	handcrafted	0.76 ± 0.10	0.80 ± 0.10
KNN	tfidf	0.67 ± 0.10	0.77 ± 0.08
RF	tfidf	0.61 ± 0.11	0.72 ± 0.09
NB	tfidf	0.61 ± 0.11	0.72 ± 0.09
MV	tfidf	0.61 ± 0.11	0.72 ± 0.09
LR	tfidf	0.61 ± 0.11	0.72 ± 0.09
RF	countvec	0.61 ± 0.11	0.72 ± 0.09
MV	countvec	0.61 ± 0.11	0.72 ± 0.09
NB	countvec	0.58 ± 0.09	0.62 ± 0.10
KNN	countvec	0.34 ± 0.15	0.29 ± 0.11
BERT	none	0.07 ± 0.17	0.12 ± 0.17

Table 6. *BookID* with *tfidf* on dataset 2.

Model	Extractor	F1 Score	Accuracy
NB	**countvec**	**0.81 ± 0.11**	**0.82 ± 0.11**
LR	countvec	0.82 ± 0.11	0.82 ± 0.10
BERT	none	0.81 ± 0.11	0.82 ± 0.10
LR	tfidf	0.80 ± 0.15	0.81 ± 0.13
NB	tfidf	0.77 ± 0.14	0.78 ± 0.13
RF	tfidf	0.71 ± 0.14	0.72 ± 0.14
MV	tfidf	0.71 ± 0.14	0.72 ± 0.14
RF	countvec	0.71 ± 0.14	0.72 ± 0.14
MV	countvec	0.71 ± 0.14	0.72 ± 0.14
KNN	tfidf	0.67 ± 0.18	0.68 ± 0.18
LR	handcrafted	0.55 ± 0.15	0.58 ± 0.13
RF	handcrafted	0.55 ± 0.14	0.57 ± 0.12
MV	handcrafted	0.55 ± 0.14	0.57 ± 0.12
NB	handcrafted	0.52 ± 0.15	0.52 ± 0.13
KNN	handcrafted	0.44 ± 0.15	0.44 ± 0.17
KNN	countvec	0.37 ± 0.16	0.44 ± 0.15

Conclusion

In our assignment, we compared multiple classifiers and models in IMapBooks discussion classification.

When comparing the different extractors we concluded that the CountVec performed the best, and then depending on the classification target handcrafted and TF-IDF. When we classified CodePreliminary the handcrafted features performed better than the TF-IDF, but performed worse when we classified Book ID and Topic. This is because our handcrafted features only take into account the structure of the messages, which would make more sense for CodePreliminary.

Overall BERT performed great with an accuracy of 92% on the first dataset. An interesting observation is that for some reason BERT performed very poorly when predicting the CodePreliminary. Unfortunately, we ran out of time to do a more in-depth analysis there. It is important to also note that BERT's accuracy dropped in the second dataset, while the accuracy of other classifiers increased. This is due to a small dataset, which is not appropriate to use for learning deeper models. It still performed decently on Book ID and Topic, but due to the time complexity, it took to predict, simpler classifiers, which take almost no time are more appropriate here.

The results also showed that most of the time Logistic Regression performed best out of the classifiers. An important result was also that KNN performed significantly better on the dataset containing only messages about the discussion.

All this means that proper filtering is important when using text classification, as any unwanted or not relevant data will result in significant losses to the accuracy. Also if we use deeper networks, we need enough data to train it, otherwise, the results will not be good, due to overfitting or not learning at all.

References

[1] Grandon Gill and Glenn Smith. Imapbook: Engaging young readers with games. *Journal of Information Technology Education: Discussion Cases*, 2:10, 01 2013.

[2] Y. Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. volume 3, pages 932–938, 01 2000.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[4] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[5] Chanawee Chavaltada, Kitsuchart Pasupa, and David Hardoon. A comparative study of machine learning techniques for automatic product categorisation. pages 10–17, 05 2017.

[6] Nitish Pandey, Abir Hudait, Debarshi Kumar Sanyal, and Amitava Sen. Automated classification of issue reports from a software issue tracker. In Pankaj Kumar Sa, Manmath Narayan Sahoo, M. Murugappan, Yulei Wu, and Banshidhar Majhi, editors, *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, pages 423–430, Singapore, 2018. Springer Singapore.

[7] Sida Wang and Christopher Manning. Baselines and bigrams: Simple, good sentiment and topic classification. pages 90–94, 07 2012.

[8] Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[9] Lei Zhu, Guijun Wang, and Xianchun Zou. Improved information gain feature selection method for chinese text classification based on word embedding. In *Proceedings of the 6th International Conference on Software and Computer Applications*, ICSCA '17, page 72–76, New York, NY, USA, 2017. Association for Computing Machinery.