FRI Natural language processing course 2021
Project report

University of Ljubljana
Faculty of Computer and Information Science

Text classification using IMapBook dataset

Tara Patricija Bosil, Kristijan Šuler and Miha Arh

Abstract

This paper presents our implementation, results and discussion about text classification using IMapBook dataset. We used some traditional feature extraction methods with addition of our custom feature extractor. Then we use generated feature vector with some basic machine learning algorithms like Naive Bayes, SVM and logistic regression. We also implement some more advanced methods like neural networks, deep neural networks and BERT transformer. We analyze and compare different combinations of datasets, features and methods.

Keywords
IMapBook, text classification, machine learning

Advisors: Slavko Žitnik

Introduction

In the era of digital documents, automatic text classification has been an important topic for researchers and also its use in production applications.

The IMapBook concept is a web-based application, designed to improve reading comprehension among elementary and middle school students and adults. It has an integrated e-reader and games which can be played when reader completes reading certain stage. This improves reader comprehension [1]. In our case readers first read a book and are later formed into groups where they talk to each other based on specific topic which is usually question and at the end they need to provide a final answer from that topic. They talk using chat inside IMapBook application and this is later used for grading criteria. Based on these messages we want to estimate the relevance of each message and predict the appropriate class.

In section `Related work` we will look over different researches and methods used in NLP and data prepossessing. Then in section `Data` we examine IMapBook data gathered from students chat and what are the classes that will be used for classification. In section `Methods` we will describe what methods will be used to classify messages and how they work. Later in section `Results` we will present results with following `Discussion` and `Conclusion`.

Related work

Text classification is a very popular and widespread field, so a lot of different researches have already been done. It is mainly focused on text preprocessing, feature extraction and use of different machine learning and deep learning methods. In this section we present some related works and used methods.

One of the most popular machine learning methods for text classification is Naive Bayes and support vector machines (SVM). Yu et. al. [2] compare the performance of both algorithms on two literary text classification tasks, the classification of Dickinson's poems and the sentimentalism classification. For the preprocessing they use namely stemming, stopword removal and statistical feature selection. Both algorithms achieved high accuracy for sentimental task, but for poem classification Naive Bayes performs better than SVM.

Recently, deep learning methods have become increasingly popular and they have achieved state-of-the-art results across many domains, including natural language processing. Conneau et. al. [3] propose new deep learning architecture, called very deep convolutional neural network VD-CNN. This architecture uses many layers with small convolutions and 3 pooling operations. They shown that with increasing the model depth up to 29 convolutional layers the performance drastically improves.

Zhou et. al. [4] implement a model called C-LSTM, which consist of two main components, convolutional neural network (CNN) and long short-term memory recurrent neural network (LSTM). The model uses CNN to extract higher-level sequences of word features and then this is fed to LSTM model to learn long-term dependencies. With this new model they have achieved very promising results.

Data

Data used in this report was gathered from students chatting about a specific book within IMapBook system. Before these conversations took place students had to read one of the three books given (The Lady or the Tiger, Design for the Future When the Future is Bleak or Just have less). After that small book clubs were formed where each group had to discuss about the given book and at the end provide a collaborative answer to the given question.

After the project was finished they ended up with approximately 800 chat messages and final responses. Data was then annotated and formatted into three tabs, Crew data, Discussion only and Crew final responses.

IMapBook dataset explanation

As mentioned there are three tabs in the IMapBook dataset. ”Crew data” tab includes the whole chat history of all users where each message was annotated. Data contained here was divided into 11 columns which tell us what book corresponds to a particular message, what was the question, which book club contains this message, which user wrote this message, time of posting, if it represents a final answer and final response number. Messages in this tab belongs to 29 different terms. These terms are then later in the preprocess part merged so that we have only 17 initial terms.

The second tab ”Discussion only data” contains similar columns to ”Crew data”, but here after discussing the book users didn’t have to provide a collaborative answer. Messages in this tab belongs to 9 different terms, where 72% of all the messages belongs to the Content Discussion term. All other 8 terms are significantly under-represented. The last tab is ”CREW final responses” which contains data about collaborative responses between users. In addition to that, it also includes grades which were submitted by graders and are based on how good that response describes the given book.

Since the is not a lot of data we had to find a different solution to make our learning and training dataset bigger. This was accomplished by joining together ”Crew data” and ”Discussion only data” which in total gave us 840 messages.

In our prediction we were mostly interested in columns ”Message” and ”CodePreliminary” (we will infer to this as Term) from combined dataset. Term is the column we wanted our model to predict.

Additional analysis

Before we started preprocessing data, we wanted to know some additional information about the provided dataset. We used a combined version of ”Crew data” and ”Discussion only data” to perform analysis seen in Table 1.

Information type	**Result**
Total number of messages	840 messages
Most active user	edf-15 (108 messages)
Average messages per user	14.24
Most active book club	Book Club One
Number of terms	17 terms
Most common message term	Content Discussion
Longest message	57 tokens
Average message length	11.08 tokens
Most discussed book	Design for the Future When the Future Is Bleak (471 messages)

Table 1. Additional information about IMapBook dataset (information is based on combined Crew data and Discussion only data).

Book analysis

After analyzing messages from students we were also interested in some information about the three books they were given. For this we have used similar techniques for analysis as with message analysis. We did this analysis since we used book content in our term prediction and by that we got some additional information-

Table 2 shows our results of books analyzed. Book1 corresponds to the book titled The Lady or the Tiger, book2 to book Just Have Less and book3 to Design for the Future When the Future Is Bleak. We can see that book1 has a lot less people mentioned than the other two, but overall all three books have similar proprieties

Information	**Book1**	**Book2**	**Book3**
Length	**15287**	10161	16220
Number of people mentioned	5	**66**	64
Number of tokens	**2769**	1906	2690
Number of nouns	611	554	**896**
Number of verbs	**477**	326	393
Number of adjectives	216	164	**239**
Number of adverbs	**149**	82	148
Number of pronouns	**259**	190	119

Table 2. Information about the three provided books.

Methods

Text classification is divided into three main parts data preprocessing, feature extraction and use of different machine learning and deep learning methods to predict the message term called ”CodePreliminary” on four different data sets.

- CREW data (17 terms)
- Discussion only data (9 terms)
- joined CREW data and Discussion only (17 terms)
- joined CREW data and Discussion only with joined terms (11 terms)

On all these four datasets we use three different set of features, TF-IDF, BoW and custom features, where we also consider the connection between the messages and the book content.

Data Preprocessing

The first and very important part of text classification is data preprocessing. As mentioned before we needed a bigger dataset for our experiments and we achieved that by combining ”Crew data” tab with messages from ”Discussion only data”. We can do this because the goal of our experiment is to predict the class ”CodePreliminary” based on ”Message” and these two columns are located in both files. After we combine these two files, our dataset contains 840 values. Then we do some prepossessing steps to prepare the data for later steps.

At the beginning our raw ”Crew data” contains 29 different terms, while only 17 of these terms are described in the provided annotation ”CodeBook” file. Because of this we looked at the differences between the terms and we found out that some of these terms had grammatical errors and some typos. For example, we joined *Instruction Question* with *Instruction Question*, *General Comment (Narrative?)* with *General Comment*, *Observation* to *General Comment*, *Outside Material* to *External Material* and at the end when we joined all such cases the final data only had these 17 terms.

Then we create a new dataset in which we join ”Crew Data” and ”Discussion only data”. ”Discussion only data” contains 9 of the initial 17 terms, so when we joined this dataset with the ”Crew data” we still have all 17 initial terms.

We also wanted to make a prediction on a dataset with joined terms. We join the terms with only a few representatives into more general ones. For example, we join *Content Question*, *Instruction Question*, *General Question* and *Assignment Instruction* into a term *Question*. Then we join *General Discussion* and *Content Discussion* to *Discussion*. And at the end we combine also *Greeting* and *Opening Statement* to a term *Greeting*. With this we get 11 terms, on which we make a prediction.

Further we also reviewed the individual messages and found out that they contain unnecessary typos. To correct these misspellings we use python library *contractions*. The messages also contain some strange characters, like . . . , “, —, ’, which we replace with corresponding UTF-8 characters.

After fixing misspellings in the text we set the text to lower case, remove all stop words and punctuations except the emoticons. We keep emoticons because one of the terms is *Emoticon/Non-verbal* and we do not want to remove them from the text. Then at the final step we use tokenization and lemmatization.

Feature Extraction

After data preprocessing was complete we had to extract features for our models. We transformed tokens into features using term frequency-inverse document frequency (TF-IDF), bag-of-words (bow) and our custom feature extractor.

Bag of words

The first technique we implemented was Bag of words where the idea is to count token appearance in all messages and representing it by a matrix. This sparse matrix consists of n rows and m columns where n equals to the number of messages and m equals to number of different ngrams or tokens. The problem with this approach is that we loose word order, hence the name Bag of words. Since there already exists some implementation of this technique we used the one from Python library sklearn. This method CountVectorizer expects different parameters that we can tweak.

First we set stop words to English to remove them from tokens. Next we set the parameters for ngram_range to (1,3) which means that we well be dealing with unigrams, bigrams and trigrams. Finaly we set the min_df to 3 which means that we will only look at tokens that appear in at least 3 documents.

TF-IDF

Second technique we used was TF-IDF which is a measure of originality of a word, by comparing the number of times a word appears in a message with the number of messages the word appears in. Like with the bag of words we need to provide some parameters to TfidfVectorizer.

For now we have set the same parameters as for Bag of words since we again want to try different ngrams and we only want to take into account words that appear at least three times.

Custom feature extractor

The last version of features was generated using BoW, TF-IDF and our own implementation of features. We wanted to gather as much information as possible so our feature vector was constructed with following properties:

- length of each message,
- longest word in message,
- shortest word in message,
- number of words in message,
- does message contain a question mark,
- number of question marks,
- does message contain a exclamation mark,
- number of exclamation marks,
- does message contain uppercase characters,
- does message contain numbers,
- number of emoticons,
- is message a url link,
- number of nouns,
- number of verbs,

- number of adjectives,
- number of adverbs,
- number of pronouns,
- is a person from book mentioned in message,
- is a word from message also present in one of the books and
- similarity between message and the whole book using cosine similarity measure.

The majority of this properties were constructed using raw messages and some were constructed using lemmatized tokens. With the properties related to the books we used IOB tags to specify what types of words we were interested in. We were also comparing all messages to each of the three books to get additional properties. Firstly we tokenized the whole book, then we lemmatized it and at the end removed all the stop words. This was done so that we minimized the chance of a random word from a message, that has no meaning (like stop words), appearing in the book multiple times and those increasing the cosine similarity.

As we present in `Results`, using custom features improves the quality of class prediction.

Algorithms

The field of text classification is very widespread and popular so there are many different machine learning and deep learning classification algorithms. We use and evaluate some classical classification algorithms, such as Support vector machines (SVM), Naive Bayes (NB) and logistic regression (LR). For NB model we use default parameters, while for the SVM we use rbf kernel, gamma of 0.001 and we set parameter C to 1000. On the other hand for LR we use lbfgs optimizer with maximum 1000 iterations. For the implementation of these models we use *scikit-learn* library.

We also implement and evaluate some neural networks, such as Multi layer perceptron (MLP), deep dense neural network (Deep NN) and basic neural network (NN). For the simple neural network we use sgd optimizer, while for the MLP and deep neural network we use Adam optimizer. This models we implemented using *Keras* library which is based on *Tensorflow*.

The deep neural network is shown in Figure 1. MLP and a basic neural network use a similar structure with the exception that the basic NN misses some layers.

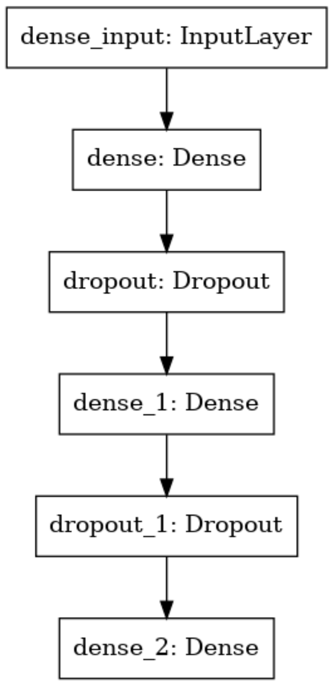

Figure 1. Visualization of deep neural network. This is a visualization of deep neural network with dense layers. The layers have *relu* activation. The model uses *adam* optimization method. Between dense layers are dropout layers with dropout frequency of 71%

BERT

Another model we use is a pre-trained case sensitive BERT model that we fine-tuned for our classification tasks. We defined multiple CNN layers with 50 filters, relu activation and dropout rate of 10%. We trained the BERT model on seven epochs on training data that consisted of 75% of our dataset, while the rest 25% is used for testing.

Baseline models

For the evaluation it is very important to compare the results of more advanced models with some baseline models. It is expected that more advanced models achieve much better results than the baselines. We implement two different baseline models, the majority and random classifier. The majority model always predicts the most common class in the training data and the random model predicts the classes randomly.

Results

We evaluate our models with calculating accuracy, precision, recall and F1-score. We use weighted F1-score because we have multi class classification problem. F1-score is a weighted harmonic mean of the precision and recall and it is normalized between 0 and 1. To compare the performance of the models, we compare the F1-score and models with higher score.

We use two different evaluation techniques. On classical classification models, such as SVM, NB and LR we use 10-fold and leave one out cross validation (LOOCV). When

	F1-score		
Model	**Crew**	**Crew and Discussions**	**Joined**
Popularity	0.41	0.33	0.29
Random	0.07	0.11	0.15
Naive Bayes	0.46	0.49	0.54
SVM	0.55	0.60	0.65
LR	0.58	**0.63**	0.65
NN	0.26	0.30	0.31
MLP	**0.60**	0.59	**0.67**
Deep NN	0.38	**0.63**	**0.67**
BERT	0.57	0.57	0.57

Table 3. Results on different datasets with custom features. Where *Crew* has only data from Crew dataset, *Crew and Discussions* is Crew data joined with discussions and the last *Joined* is the same as second column with less classes as some classes had very little instances and we joined them in larger with similar or same topic.

we compare the results between 10-fold and LOOCV, the results are almost the same. Because of that we decide to use 10-fold cross validation, because LOOCV is much more computationally expensive.

And for the neural networks we use simple hold-out evaluation. Firstly we split the data to train and test set, where we use 75% of data for the train set and the rest 25% for the test set. We further split the train data into train and validation set, where we use 10% of the train data as a validation data.

Firstly we test our classification models on four different datasets based on custom features. With Discussion only data we get very good results, for example logistic regression achieves F1-score of 0.85. This is because the Discussion only dataset contains only 9 terms and 72% of all cases belong to the term Content Discussion. Because of that we compare the model results on three other datasets, Crew data with 17 different terms, joined Crew and Discussion only data with also 17 different terms and joined Crew and Discussion only data with joined 11 terms, where the distribution between the classes is slightly better. We can see the results of all classification algorithms in Table 3. We notice that all classifiers outperforms baseline models, which is expected. The best performing classification model on a dataset with all initial 17 terms is logistic regression and deep neural network, both achieves F1-score of 0.63. And the best performing model on a dataset with joined terms are MLP and deep neural network, both these two models achieves F1-score of 0.67. From the table we can also see that the models with joined terms achieves better scores than other two datasets, which is expected because here we we join less representative terms together.

From the above table we can see that we achieve the best results with the Joined dataset. Because of that we also compare the results of the classification models according to the use of different features, TF-IDF, BoW and custom features. In the table ?? we can see the results based on different features. The best F1-score of 0.67 we achieved with custom features with deep neural network.

Model	**Accuracy**	**Precision**	Recall	**F1-score**
Popularity	0.48	0.23	0.48	0.31
Random	0.08	0.20	0.08	0.09
BERT	0.60	0.56	0.60	0.57
TF-IDF				
Naive Bayes	0.5	0.36	0.51	0.36
SVM	0.67	0.65	0.68	0.63
LR	0.68	0.65	0.68	0.64
NN	0.47	0.22	0.47	0.29
MLP	0.51	0.37	0.51	0.40
Deep NN	0.50	0.29	0.49	0.37
BOW				
Naive Bayes	0.59	0.52	0.59	0.50
SVM	0.56	0.47	0.56	0.48
LR	0.57	0.64	0.57	0.58
NN	0.47	0.22	0.47	0.30
MLP	0.65	0.60	0.65	0.61
Deep NN	0.60	0.54	0.60	0.53
Custom features				
Naive Bayes	0.606	0.52	0.61	0.54
SVM	0.67	0.67	0.65	0.65
LR	0.64	0.69	0.64	0.65
NN	0.48	0.29	0.48	0.32
MLP	0.67	0.65	0.67	0.65
Deep NN	**0.69**	**0.69**	**0.69**	**0.67**

Discussion

Our results were quite good if we take into account that the dataset didn't contain many messages. Since our goal was to predict what was the content type of a particular message we tested our implementations on raw datasets and also different versions of combined datasets. The combined versions gave us better results those we could say that it was a right move. We also analyzed three different feature extraction techniques and again improved our classification by using our custom features. We get the best classification results with the Multi layer perceptron and deep neural network. Both these two models achieves F1-score of 0.67. Similar results are also achieved with Support Vector Machine and Logistic regression, both achieves slightly worse F1-score of 0.65. We found out that the results depend on different features as well on how many different terms we have and how they are distributed. We could also improved our results by using some other algorithms or if we will have larger dataset, but due to a small dataset we achieved quite good results.

Conclusion

In this paper we presented implementation and analysis of text classification using IMapBook dataset. We went through the whole process from data cleaning, data reprocessing, feature extraction, model training and class prediction. We used multiple techniques in each part so that we got a variety of results that we compared. In addition to BoW and TF-IDF features we constructed our own a added the other two to it. We found out that combining these features works best and improves our model. We then divided our data into train, test and validation data for deep learning methods, and for the other methods we used 10-fold cross validation technique on test and train data. We compared the results between all models and found out that deep neural network works best, but other methods aren't far behind (with exception of popularity and random classifier).

There are multiple direction for further improving out work. For example we could try to improve out custom feature extractor by adding additional properties or by analyzing it more deeply. There are also other methods for classification we could experiment with. In general our implementations achieve pretty good results on such a small dataset.

References

[1] Imapbook. https://www.imapbook.com/blog/. (Accessed on 03/26/2021).

[2] Bei Yu. An evaluation of text classification methods for literary study. *Literary and Linguistic Computing*, 23(3):327–343, 2008.

[3] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[4] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.