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

Text classifications using IMapBook dataset

Tara Patricija Bosil, Kristijan Šuler and Miha Arh

Abstract

TODO

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 focus on text preprocessing, feature extraction and using 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.

IMapBook dataset explanation

Crew data 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.

Discussion only data contains similar columns to Crew data, but here users didn't have to provide a collaborative response.

Final book club responses are presented in Crew final responses where we can also find grades for each response.

In our prediction we will be mostly interested in "Messages" and "CodePreliminary" (we will infer to this as Term) from crew data and discussion only data. Term is exactly what our models had to predict.

Additional analysis

Before we started preprocessing data we were interested in additional information about 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).

Methods

Text classification is mainly focus on three parts, data preprocessing, feature extraction and using different machine learning and deep learning methods.

Data Preprocessing

The first and very important part of text classification is data preprocessing. Firstly we combine the crew data and discussion only data. We can do this because the goal of our task 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 the training.

At the beginnig our raw data contains 24 different terms, while only 17 of these terms are described in the codebook file. Because of that we decide to correct the misspellings and join some terms together, so the final data will have only these 17 terms. Also there are some unnecessary typos in the messages, so we use *contractions* library to correct these misspellings. We also change the messages to lowercase, remove all stop words and punctuation except the emoticons. Then we use *nltk* library for tokenization and lemmatization. After these preprocessing steps the final data contains 840 messages and 17 different terms.

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. Before feature extraction we also had to split data into train and test set, where train set included 75% of messages and test set included 25% of messages.

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

We tried to implement our own feature extractor using mostly token based features. Then we want to merge it with Bag of words and TF-IDF.

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 machine learning algorithms, such as Support vector machines (SVM), Naive Bayes (NB) and logistic regression (LR). 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). This models we implemented using *Keras* library which is based on *Tensorflow*.

For the evaluation of these more advanced methods it is very important to compare the results with some baseline models. The more advanced models should outperform baselines. If not you know you have to do some more work and improve the models. We use two different baseline models, the popularity and the random classifier. The popularity model always predicts the most common class in the training data and the random model predicts the classes randomly.

Neural networks

We implemented different and tested different types of neural networks. The deep neural network is shown in Figure 1, Muli layer perceptron is shown in Figure 2 and simple one layered neural network is shown in Figure 5.

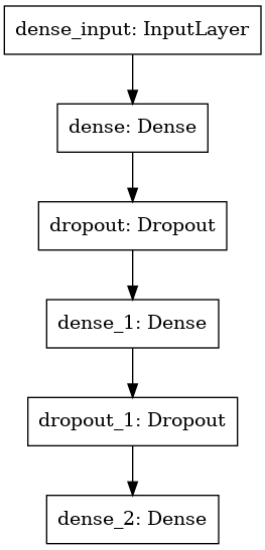

Figure 1. Visualization of deep neural network. This is a visualization of deep neural network with dense layers. The layers have *relu* activation. The model uses *adam* optimization method. Between dense layers are dropout layers with dropout frequency of 71%

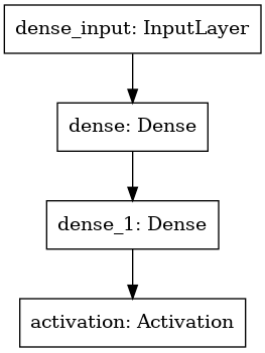

Figure 2. Visualization of Multi layer perceptron. This is a visualization of MLP. The layers have *relu* activation and model uses *adam* optimization method. The first layer have 17 input nodes and second dense layer have 51 nodes and the third 17 with *softmax* activation.

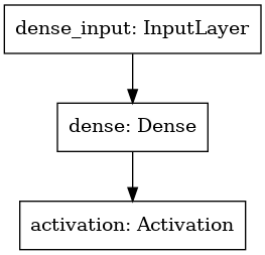

Figure 3. Visualization of neural network. This is a visualization of simple neural network with one hidden dense layer with 17 nodes.

Results

The evaluation consisted of trying different features on baseline models, Support vector machine, Naive Bayes model, Logistic regression and neural networks. All the models are trained and evaluated on the same data. We use 75 % of data for the training set and 25 % of the data for the testing. Then we calculate different metrices, such as accuracy, precision, recall and F1-score. Because we have multi class classification problem we use weighted F1-score.

In Table 2 we can see the results for all used machine learning models based on TF-IDF features. All more advanced models outperforms baselines, which is expected. The best performing classification algorithm is logistic regression which achieves F1-score of 0.631. Support vector machine algorithm also performs good, its scores are comparable with the logistic regression scores.

Model	Accuracy	Precision	Recall	F1-score
Popularity	0.466	0.218	0.467	0.297
Random	0.061	0.233	0.062	0.088
Naive Bayes	0.490	0.442	0.49	0.344
SVM	0.662	0.608	0.662	0.599
LR	0.690	0.682	0.69	0.631
NN	0.467	0.218	0.47	0.29
MLP	0.510	0.35	0.51	0.40
Deep NN	0.495	0.29	0.49	0.37

Table 2. Results for all models using TFIDF features.

And in the Table 3 we can see results for all models using the bag of words features. We notice that all more advanced models achives better scores than the baselines. The best performing algorithm is Multi layer perceptron ehich achieves F1-score of 0.61.

Model	Accuracy	Precision	Recall	F1-score
Popularity	0.466	0.218	0.467	0.297
Random	0.061	0.233	0.062	0.088
Naive Bayes	0.576	0.502	0.576	0.474
SVM	0.438	0.577	0.438	0.452
LR	0.566	0.597	0.567	0.567
NN	0.467	0.218	0.47	0.30
MLP	0.647	0.60	0.65	0.61
Deep NN	0.605	0.54	0.60	0.53

Table 3. Results for all models using BOW features.

Below we can see the performnace of Multi layer perceptron during the training. Figure 4 shows us the accuracy scores on train and test data over all epochs and on Figure 5 we can see loss values on train and test data.

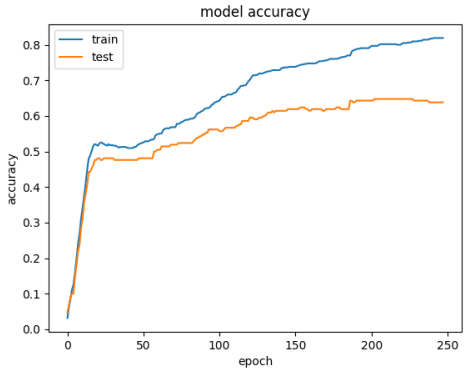

Figure 4. Plot of accuracy for Multi layer perceptron on train and test data over epochs

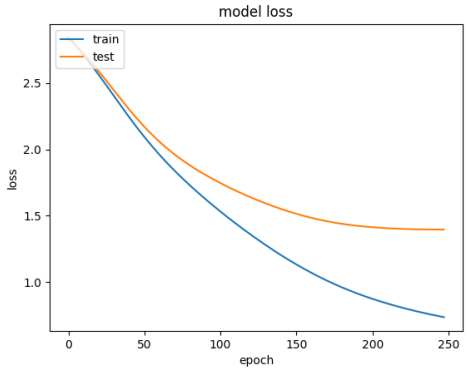

Figure 5. Plot of Multi layer perceptron loss on train and test data over epochs.

Discussion

Conclusion

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.