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

IMapBook Collaborative Discussions Classification

Jan Ivanovič, and Grega Dvoršak

Abstract

Keywords
Natural language processing, Text classification, IMapBook

Advisors: Slavko Žitnik

Introduction

This report presents our practical work for the Natural Language Processing course at the Faculty of Computer and Information Science at University of Ljubljana. Our task was to classify chat messages from the IMapBook platform into several predefined categories such as content discussion, greeting, general comment etc. The IMapBook platform's target users are elementary school pupils, who use the platform to discuss and answer questions related to a book they were required to read. The platform can also be used to study the pupils' learning process and other aspects of their conduct while answering and discussing said questions.

To solve the text classification task, our goal was to first use more traditional approaches such as [1], [2] and [3], and then compare the results to some methods closer to state-of-the-art such as [4], [5] and [6]. The dataset contains the discussion messages which are to be classified, along with plenty additional information, such as message senders, groups and timestamps, grades for the submitted answers and more, so an important task was also to decide which of these pieces of information would be useful for classification and to use them efficiently. We report on the methods used, the results and our findings in the following sections.

Related Work

To get a better understanding of text classification and its approaches a number of other articles with similar goals were reviewed. We reviewed articles where the authors used traditional classification techniques and articles about using neural networks for text classification.

The first article [1] presents a novel technique for the classification of sentences as Dialogue Acts, based on structural information contained in function words. It focuses on classifying questions or non-questions as a generally useful task in agent-based systems. The proposed technique extracts salient features by replacing function words with numeric tokens and replacing each content word with a standard numeric wildcard token. The Decision Tree, which is a well-established classification technique, has been chosen for this work. Experiments provide evidence of potential for highly effective classification, with a significant achievement on a challenging dataset, before any optimisation of feature extraction has taken place.

Dialogue-based intelligent tutoring systems use speech act classifiers to categorize student input into answers, questions, and other speech acts. Previous work has primarily focused on question classification. In this paper [2] the authors present a complimentary speech act classifier that focuses primarily on non-questions, which was developed using machine learning techniques. Their results show that an effective speech act classifier can be developed directly from labeled data using decision trees.

The third article [3] illustrates the text classification process using machine learning techniques. It covers different topics and steps which we need to follow when doing text classification, with the goal to present different techniques and their results.

Focusing more on neural network based approaches, the next article [4] presents a deep learning architecture suitable for text classification where extreme multi-class and multi-label problems are considered. The authors use a hierarchical label set and define a methodology called Hierarchical Label Set Expansion (HLSE) to regularize the data labels and then analyze the impact of different word embedding models. The results are said to prove the usefulness of the HLSE methodology and provide insight to some combinations of word embedding models.

The next paper focuses on text classification for medical

data using a neural network approach [5]. It is a special case of text classification, as it uses specific records and literature, thus causing a high dimensionality and data sparsity problem. To solve these problems, the authors propose a unified neural network method which extracts features from sentences via a convolutional layer with a bidirectional gated recurrent unit (BIGRU) and an attention mechanism to generate the sentence and document representations. The final step is a classifier of medical text categories. The method is said to be effective.

The final paper [6] uses a graph convolutional neural network for the text classification task. The authors build a single text graph for a corpus based on word co-occurrence and document-word relations which is used for learning. The network uses one-hot representations for words and documents and then jointly learns the embeddings for both. Results show, that the main advantage of this approach when compared to other state-of-the-art methods is that it performs better, when the amount of training data is reduced, suggesting robustness.

Methods

As previously mentioned, we classify chat messages taken from a platform used by elementary school pupils to discuss questions about books that they are required to read. The classes for this classification problem are: Content Discussion, Logistics, Greeting, Instruction Question, Assignment Question, General Comment, Incomplete/typo, Feedback, Discussion Wrap-up, Outside Material, Opening Statement, General Question, Content Question, Emoticon/Non-verbal, Assignment Instructions, Response. To solve the classification task, we use supervised machine learning methods and compare their performance to determine which approach gives the best results.

0.1 Preprocessing

As the text is highly dimensional unstructured data, it has to be cleaned and prepared before it can be analyzed. This helps achieve more efficient results as our classification methods do not have to deal with irrelevant data. It is also important not to remove important text features which are determined experimentally as the features' importance differs for each problem and dataset. Preprocessing the data involves many tasks which are chosen according to the representation of text we wish to use. In our case we aim to use the bag of words features and pre-trained word2vec embeddings. The usual tasks performed are tokenization of words and removal of unnecessary data such as empty entries, numbers, punctuation etc. In order to use the bag of words representation we also perform stop word removal and make a word list dictionary with a counter for each word. The bag of words representation is explained further in the following sections.

0.2 Feature Extraction Methods

As is well known, machine learning algorithms prefer well defined fixed-length inputs and outputs. and cannot work with raw text directly, so the text must be converted into numerical structures such as vectors of numbers. This process , called feature extraction helps in reducing the set of terms that are to be used in classification, which helps improve efficiency and accuracy. The feature extraction methods we used in our implementation are described below.

0.2.1 Bag Of Words

The bag-of-words model, or BoW for short, is a simplifying representation used in natural language processing. A bag-of-words representation describes the occurrence of words within a document [7]. It involves a dictionary of known words and a measure of the presence of known words. The approach is very simple and flexible, and can be used in many ways for extracting features from documents. First, we define a fixed length vector where each entry corresponds to a word in our pre-defined dictionary of all words. The size of the vector equals the size of the dictionary. Then, for representing a piece of text using this vector, we count how many times each word of our dictionary appears in the text and mark it as the corresponding vector entry. A representation of this is shown in Figure 1.

Statement 1: Invest in Tesla stock
Statement 2: Tesla stock is rising

Dictionary

	TESLA	IN	INVEST	IS	STOCK	RISING
Statement 1:	1	1	1	0	1	0
Statement 2:	1	0	0	1	1	1

Figure 1. Bag of Words Example

The bag-of-words model is relatively simple to implement and offers a lot of flexibility for tuning it on specific datasets. However, its shortcomings are that the dictionary requires careful design, as to manage the size, which impacts the sparsity of the document representations. Another problem is that discarding the word order of sentences ignores the context, and may in turn cause ambiguity of word meanings in the document and other problems related to semantics. An important step related to the bag of words representation is the previously mentioned preprocessing step, where we could also use n-grams or tf-idf values instead of counts.

0.2.2 Word2vec

The word2vec representation is a type of word embedding which is one of the most commonly used representations of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. When using Word embeddings, individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often included in the field of deep learning [8]. The Word2vec allows transforming words into vectors of numbers. Word vectors represent

abstract features which describe the word similarities and relationships. A property of Word2Vec is that operations on its vectors approximately keep the characteristics of the words, so joining vectors from words of a sentence produce a vector that is likely to represent the general topic of the sentence. A good example of that is shown in Figure 2 below, where an equation of vectorised words *king*, *man* and *woman* returns the approximate vectorised representation of the word *queen*.

$$king - man + woman \approx queen \tag{1}$$

Figure 2. Word2vec words relationships

0.3 Learning Methods

In this section, we present the algorithms which perform the classification task using the above mentioned features.

0.3.1 Naive Bayes

Naive Bayes is an example of a simple solution which still achieves relatively good results. In spite of the great advances of machine learning in recent years, it is useful because of its simplicity and speed. It is also relatively accurate, and reliable and has been successfully used for many purposes, including natural language processing problems [9]. It is a supervised machine learning algorithm that uses probability and Bayes' Theorem to predict classes. It calculates the probability of each class for a given text segment, and outputs the class with the highest probability. The probabilities are calculated using the Bayes' Theorem, which denotes the probability of a feature, based on prior knowledge of conditions that might be related to that feature. The theorem is shown in equation 2, where P(A|B) and P(B|A) are conditional probabilities and P(A) and P(B) are the prior known probabilities.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{2}$$

There are two ways in which Naive Bayes can be used, the Multinomial model and the Bernoulli model. In the former, the documents are the classes which are treated in the estimation as a separate "language". The latter is suitable for discrete data and it is designed for Binary features and works with occurrence counts. In our implementation we make use of the Bernoulli Naive Bayes.

0.3.2 Random Forest

Random Forest is a supervised learning algorithm which combines decision trees that work together and form stronger classification predictors [10]. The main goal of a random forest tree is to combine several base level predictors using a learning algorithm which forms an effective and robust single predictor as shown in figure 3.

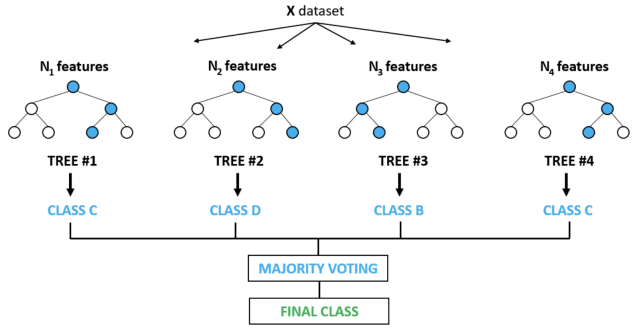

Figure 3. Diagram of a random decision forest

Random Forest adds additional randomness to the model. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This ensures wider diversity that generally contributes to a better model. Only a random subset of the features is taken into consideration by the algorithm when splitting a node in a tree. Trees can be made more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds like a standard decision tree would do [10]. In the implementation the forest classifiers are fitted with two arrays, one with training data and the other with the target values of the testing data.

While being a collection of decision trees, there are some differences. A decision tree will formulate a set of rules, which will be used to make the predictions. For example, to predict whether a person will click on an online advertisement, one might collect the ads the person clicked on in the past and some features that describe his/her decision [10]. Rules that help predict whether the advertisement will be clicked or not will be generated. On the other hand, the Random Forest algorithm randomly selects observations and features to build several decision trees and then averages the results. Another difference is that "deep" decision trees might suffer from overfitting. Most of the time, Random Forest prevents this by creating random subsets of the features and building smaller trees using those subsets. Afterwards, it combines the subtrees. It's important to note this may be inconsistent time and also may make the computation slower, depending on how many trees the random forest builds.

0.4 Implementation summary

The first step of the classification process is preprocessing. Because classes such as Incomplete/typo and Emoticon/Non-verbal may contain some unconventional words or tokens, we

pay extra attention as to preserve the original text on one hand and prepare it for feature extraction on the other. Some words and symbols that are useful for our task may be removed in commonly used approaches as a preprocessing step. An example of this would be if we remove incorrectly written words or specific punctuation which are useful for classes like *Incomplete/typo* and *Emoticon/Non-verbal*. Taking this into account, our preprocessing procedure removes URLs, usernames, unavailable text, numbers, empty entries and special characters like the slash signs, periods etc.

Next is the feature extraction. In our implementation, we extract features based on counting such as number of words, uppercase words, exclamation points, question words etc. We also derive a word list with counts of each word in the data, which is used when generating the bag of words representations and we also use word2vec features which are from a model pre-trained on 2 billion tweets with 200 dimensions [11]. The features are applied to a Naive Bayes and a Random Forest classifier. Each classifier returns predictions for the chat messages used as the input. The results are presented in the following section. A diagram shown in figure 4 shows the necessary steps taken by the classifiers to obtain the output.

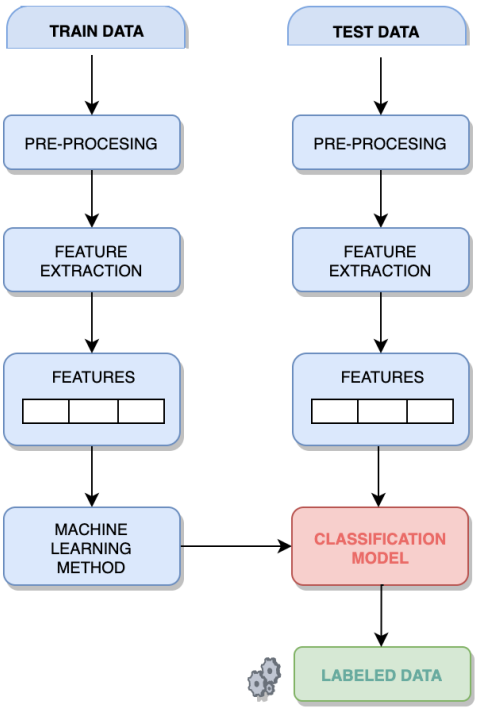

Figure 4. Implementation Diagram for IMapBook Classification

Results

This section presents the results of the above mentioned algorithms tested on the IMapBook dataset. To get the performance and quality of the implemented machine learning methods we split the dataset with a 7:3 train:test ratio. To get the general performance of the algorithms, 8-fold cross-validation is introduced. The results for the Naive Bayes and Random Forest classifiers are listed below.

Table 1

	F1	Precision	Recall
Content Discussion	0.	0.	0.
Greeting	0.	0.	0.
Logistics	0.690	0.526	1.
Instruction Question	0.	0.	0.
Assignment Question	0.	0.	0.
General Comment	0.	0.	0.
Incomplete/typo	0.	0.	0.
Feedback Discussion	0.	0.	0.
Wrap-up	0.	0.	0.
Outside Material	0.	0.	0.
Opening Statement	0.	0.	0.
General Question	0.	0.	0.
Content Question	0.083	1.	0.043
Emoticon/Non-verbal	0.	0.	0.
Assignment Instructions	0.	0.	0.
Response	0.492	0.484	0.5
Accuracy	**0.522**		

0.5 Naive Bayes

We first test the performance of the algorithm without k-fold cross-validation. The results are shown in Table 1. The algorithm got a 52.2% accuracy. Which is expected as the Naive Bayes is a simple method and a larger number of classes is used.

After testing the performance with 8-fold cross-validation the accuracy is slightly higher compared to the first calculation. The average accuracy equals 53.6% The cross-validation scores are shown in table 2. These results are to be expected from a simple classifier such as Naive Bayes given the sixteen classes.

Table 2

	1	*2*	*3*	*4*
Accuracy	50.6%	52.9%	51.2%	53.6%
	5	*6*	*7*	*8*
Accuracy	56.0%	53.6%	59.5%	51.2%
Avg. Accuracy	**53.6%**			

0.6 Random Forest

As with Naive Bayes we firstly test the performance of the Random Forest algorithm without k-fold cross-validation. The classification accuracy equals 61.2%, which is a significant improvement to the results of Naive Bayes. More information is available in table 3.

Table 3

	F1	Precision	Recall
Content Discussion	0.	0.	0.
Greeting	0.	0.	0.
Logistics	0.75	0.6	1.
Instruction Question	0.	0.	0.
Assignment Question	0.67	1.	0.5
General Comment	0.857	0.75	1.
Incomplete/typo	0.	0.	0.
Feedback Discussion	0.	0.	0.
Wrap-up	0.	0.	0.
Outside Material	0.5	1.	0.33
Opening Statement	0.	0.	0.
General Question	0.	0.	0.
Content Question	0.323	0.625	0.217
Emoticon/Non-verbal	0.	0.	0.
Assignment Instructions	0.8	1.	0.67
Response	0.657	0.595	0.733
Accuracy	**0.611**		

The results from the 8-fold cross-validation are shown to average around 60% as sown in table 4). The precise average value of the accuracy reaches 58.3%. These results are satisfactory for the given problem, however it can be observed that some classes are not represented at all in the output for both algorithms. The cause most likely lies with the fact that the problematic classes aren't represented frequently enough in the testing set itself and so there is too few learning data for the model to classify these cases correctly.

Table 4

	1	*2*	*3*	*4*
Accuracy	55.3%	50.6%	58.3%	60.7%
	5	*6*	*7*	*8*
Accuracy	59.5%	61.9%	63.1%	57.1%
Avg. Accuracy	**58.3%**			

Discussion

In this project, we implemented several classification algorithms and tested their performance on the IMapBook dataset, where we classify chat messages posted by elementary school pupils on the topic of a book the pupils were required to read. We implemented the Naive Bayes classifier and the Random Forest classifier to solve this task. Naive Bayes gives results close to 50% due to there being a larger number of classes which makes the computations of the probabilities for the Naive Bayes more complex. The Random Forest classifier performs much better as the results are close to 60% in accuracy. A problem which occurred in the classification is that some classes received zero predictions even though they are present in the training and testing sets. This is most likely a consequence of a lack of data and too few training examples, as those classes tend to be underrepresented, so this is a possible area of improvement for the future work. Another thing to consider would be to compare these results to a neural network approach, which is likely to give better results as deep neural network are used as state of the art methods in many areas including natural language processing.

References

[1] Zuhair Bandar James O'Shea and Keeley Crockett. A machine learning approach to speech act classification using function words. page 82–91, 2010.

[2] Andrew Olney Travis Rasor and Sidney D'Mello. Student speech act classification using machine learning. 2011.

[3] Emmanouil Ikonomakis, Sotiris Kotsiantis, and V. Tampakas. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4:966–974, 08 2005.

[4] Francesco Gargiulo, Stefano Silvestri, Mario Ciampi, and Giuseppe De Pietro. Deep neural network for hierarchical extreme multi-label text classification. *Applied Soft Computing*, 79:125–138, 2019.

[5] Li Qing, Weng Linhong, and Ding Xuehai. A novel neural network-based method for medical text classification. *Future Internet*, 11(12), 2019.

[6] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7370–7377, Jul. 2019.

[7] Jason Brownlee. A gentle introduction to the bag-of-words model. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model, October 2017. [Accesed on: 23.12.2020].

[8] Jason Brownlee. What are word embeddings for text? Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model/, October 2017. [Accesed on: 23.12.2020].

[9] MonkeyLearn. A practical explanation of a naive bayes classifier. Available: https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/. [Accesed on: 23.12.2020].

[10] Niklas Donges. A complete guide to the random forest algorithm. Available: https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/, June 2019. [Accesed on: 23.12.2020].

[11] Jason Brownlee. Twitter pre-trained word vectors. Available: https://zenodo.org/record/3237458#.X-J34elKidY, June 2019. [Accesed on: 23.12.2020].