FRI Natural language processing course 2021
Project report

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

1. 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.

2. 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.

3. Dataset and Preprocessing

3.1 Dataset

For this research a pre-labeled chat messages dataset from the IMapBook platform is used. The dataset contains 711 unique chat messages, where the two most important columns for computation are:

- Category – pre-labeled category of a chat message
- Chat Message – raw and unfiltered message text

Overall there are 16 uniquely predefined categories which a message can be classified into. Before doing the main feature extraction processes it is always important to take a closer look and analyse the data, as we get a better understanding of what kind of data we are dealing with. Some trimming and additional editing of the category text is needed to correctly distribute our chat messages. Figure 1 represents six of the most populated categories. The Content Discussion category has the highest number of samples. We can see quite a big drop after that. The remaining categories contain even less samples than the Feedback category which is the least populated category on the graph. We can observe that the data is not evenly distributed and because of this we can assume less accurate results as the training process might be inefficient for some categories.

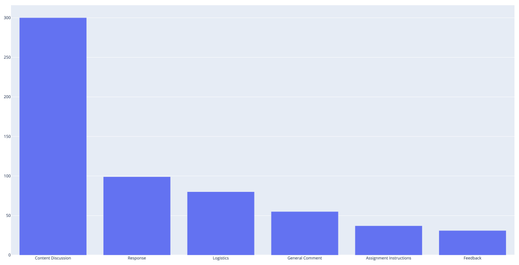


Figure 1. Dataset Distribution

Following further analysis, We prepared the dataset so we could clearly see which of the students were the most active. This is presented in Figure 2. The two most active students were *edf-16* and *edf-15*. They both delivered 104 messages of the course of the discussion which is almost twice as much as the third most active student.

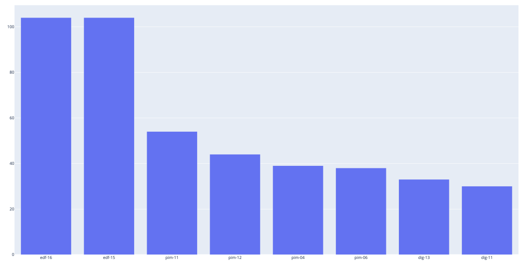


Figure 2. Most Active Students

To determine which of the students contributed more, chat categories needed to be taken into account. We analysed the data even further and shaped it in a way we so can see to which category did most of the students messages classify. The category that probably holds the most impact is the Content Discussion category as it represents discourse on the assigned material and follow-up questions. However the Response category is also important as it represents an answer or reaction to another's question or statement. These two categories are closely connected to the *Discussion only data* and the final responses in the dataset. Figure 3 shows in which categories student *edf-16* was most active and Figure 4 represents the same but for student *edf-15*. Student *edf-16* wrote 30 chat messages that were classified in the Response category and 21 messages which were classified in the Content Discussion Category. On the other hand student *edf-15* had 46 messages in the Discussion Category and 11 in the Response category. We can assume that they both had a great impact on the discussion topic as they both scored high on the most important categories.

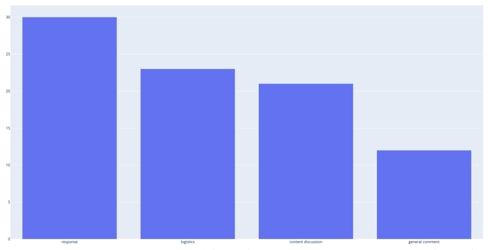


Figure 3. Student edf-16 message categorisation

3.2 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

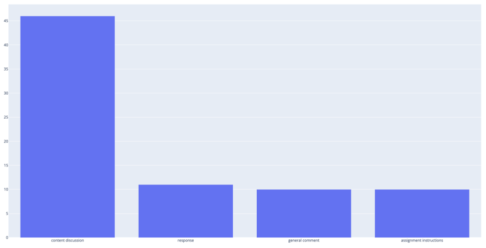


Figure 4. Student edf-15 message categorisation

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. We divide the data preprocessing into three steps:

- Cleansing
- Text Processing
- Wordlist

3.2.1 Cleansing

Data gathered from outside sources usually includes some noisy text and our dataset is no exception. This is why cleansing of such text has become a necessary process in the world of natural language processing. We want to remove any extra text that we think holds no additional value that can be further analyzed. This may be some URLs, special characters and also numbers.

3.2.2 Text Processing

After we manage to produce clean data more preprocessing has to be done. We must further transform our text so we can benefit the most from it. This step includes Tokenization and Stemming of our data inputs:

- Tokenization – the process of breaking complex data like sentences into simple units called tokens. We split a sentence into a list of words.
- Stemming – a normalization technique where list of tokenized words are converted into shorten root words to remove redundancy.

This process returns a vectorised sentence. An example is shown in the Figure 5 below.

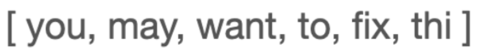

[you, may, want, to, fix, thi]

Figure 5. Text Processing

3.2.3 Wordlist

The last step we need to implement, to successfully make a bag-of-words representation of the data is build a wordlist (dictionary). The wordlist is made simply by counting the occurrences of every unique word across the dataset. However, before building an efficient wordlist we must take care of one more thing - stop words. Stop words usually represent the most common words in a language, which are filtered out before or after processing of natural language data (text) as they generally don't affect the sentiment of the text. To get a list of stop words we would want to exclude from text, we use the Natural Language Toolkit. However the list also includes some words that could help us determine a class. These kinds of words are mean to be whitelisted.

3.2.4 Extensions of features

When people are expressing themselves, asking questions or writing an opinion it is really common that they will use some special characters like exclamation marks, question marks, emojis, etc. The usage of this special characters can be to some extent connected to specific types of classes of our data. For example a question mark is generally to be used when seeking information. This can help us classify a statement that contains a question mark to one of the question type classes. Similar can be said for the use of emojis. Because of this, additional features are added that might help to better separate and classify our data, consequently giving better performance. Extraction of those features must be done before any of the above preprocessing actions happen.

4. Methods

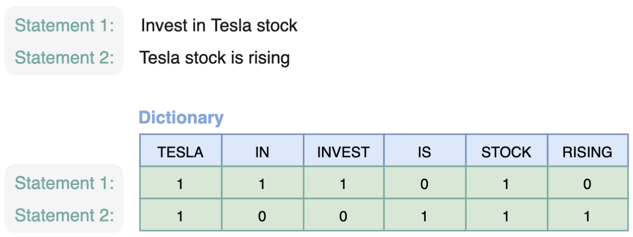
As previously mentioned, we classify chat messages taken from a platform used by elementary school pupils to discuss questions about books that they are required to read. The classes for this classification problem are: Content Discussion, Logistics, Greeting, Instruction Question, Assignment Question, 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.

4.1 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.

4.1.1 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 6.



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 6. 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.

4.1.2 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 7 below, where an equation of vectorised words *king*, *man* and *woman* returns the approximate vectorised representation of the word *queen*.

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

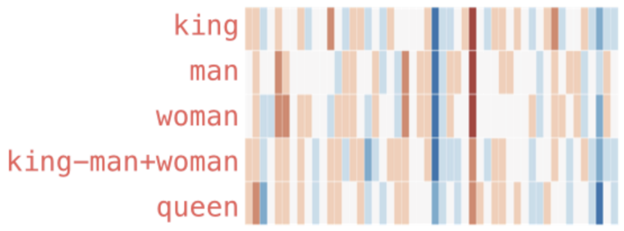


Figure 7. Word2vec words relationships

4.2 Learning Methods

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

4.2.1 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.

4.2.2 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 8.

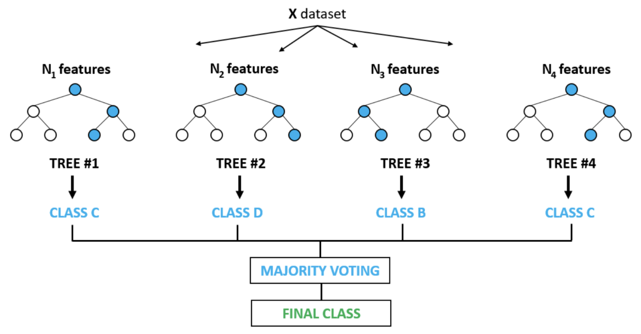


Figure 8. Diagram of a random decision forest

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.

4.2.3 BERT Neural Network Models

Bert models [11] are a subgroup of neural networks that are gaining popularity in natural language processing. Neural networks are a part of deep learning methods which are quite different from the previously explained traditional methods. a network is a complex model which is fed a large amount of (labeled) data and it learns to predict the correct classes by adjusting the weights. BERT networks provide a new architecture which can be used in many natural language processing tasks and provides significant improvements compared to other previously successful deep learning methods. The models are said to be difficult to implement correctly, so we decided to use a pre-trained model which solves a similar task and trained the model providing our own data. The architecture of a BERT model in simple terms follows the Encoder-Decoder structure, where the encoder maps an input sequence of symbol representations to a sequence of continuous representations. The decoder then generates an output sequence of symbols one element at a time. At each step the model is auto-regressive, consuming the previously generated symbols as additional input when generating the next. More information can be found in the implementation paper [11].

4.3 Implementation Summary for Traditional Methods

A diagram shown in figure 9 shows the necessary steps taken by the classifiers to obtain the output. 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.

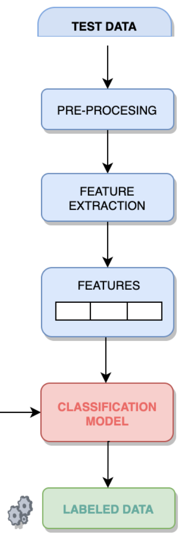


Figure 9. Implementation Diagram for IMapBook Classification

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 [12]. 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 9 shows the necessary steps taken by the classifiers to obtain the output.

We also use custom features which we designed after studying the dataset and should contribute to better performance. The custom features are added after the bag of words is created. Most of them are still count-based and are added as an extension to the bag of words representation, where the counted elements are not word tokens but rather number of question marks, exclamation points, ellipses (three dots), etc. The more important custom features are also number of emoticons which should help with detecting the emoticon class, and quotes, where we count quotations from the book material. Some of these counters can turn out to be zero or one, so we could use a binary feature, but because the characters may appear more than once, we cover those cases by counting all occurrences instead. We test the performance of the custom features by running the classifiers with only the classic bag of words representation and comparing the results with the ones where the custom features are present.

4.4 BERT Implementation

For the implementation of the BERT model, we did not perform the same preprocessing steps as with the traditional methods. The data was loaded as is in the dataset and then the target classes were transformed into a number format to be able to train the network. We split the data into training testing and validation sets next. There were two possible approaches available. One was to use a trained BERT network and try to get a good score using the existing architecture, and the other was to use BERT embeddings in a custom model, where more convolutional and other types of layers could be added. Both methods were tested and as it turned out, the first approach gave more promising initial results, so the custom model training was discontinued.

One of the main tasks once the data was prepared for the learning process was to find a suitable pre-trained model which would have the correct architecture (non-binary classification) and thus converge to good results faster. We used the Hugging Face portal for this task and found a pre-trained model which classifies a business description into one of 62 industry tags [13]. It was originally trained on 7000 samples of Business Descriptions and associated labels of companies in India. The model is a variant of BERT called DistilBERT, which is more lightweight, uses less resources, but still gives results comparable to other BERT variants. Some other models were tested, but this model was the most promising to train when taking into account the validation accuracy and loss function behaviour after the first few epochs of training.

Once the model was chosen, we used a BERT tokenizer and transformed the data to a compatible input format for the model. The main features created at this step were input ids, attention masks and token type ids. The data was organized into batches and the training process was next. Because the loss function was decreasing slowly, instead of training a huge number of epochs, we saved the weights at the end of each run into a file and then loaded them into the model at the start of the next run. This prevented potential errors such as our machine being out of memory. After about five runs, the accuracy already surpassed that of the traditional methods, proving the hypothesis that deep learning methods perform better. Another mechanism used was the Reduce in Plateau method, which reduces the learning rate if the results stagnate for several epochs in a row. Details about the results are described in the following section.

5. 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.

5.1 Meassurments

To calculate the efficiency and performance of our algorithms we use the following measurements:

- *Accuracy* – measures how many messages were predicted correctly out of all the messages in the datasets.
- *Recall* – measures how many messages were predicted correctly as belonging to a given category out of all the messages that should have been predicted as belonging to the category.
- *Precision* – measures how many messages were predicted correctly belonging to a certain category out of all the messages that are predicted either correctly or incorrectly.
- *F1 score* – the measure of calculating the weighted average of precision and recall. F1 score ranges from 0 to 1 and F1 score is considered perfect when it is 1 which means that the model as low false positives and low false negatives.

5.2 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 when using all custom features. Which is expected as the Naive Bayes is a simple method and a larger number of classes is used. Classes with the value of 0.0 are not included in the table. These classes are *Content*

Discussion, Greeting, Instruction Question, Assignment Question, General Comment, Incomplete/typo, Feedback Discussion, Wrap-up, Outside Material, Opening Statement, General Question, Emoticon/Non-verbal, Assignment Instructions

	F1	Precision	Recall
Logistics	0.690	0.526	1.
Content Question	0.083	1.	0.043
Response	0.492	0.484	0.5
Accuracy	**0.522**		

Table 1. Naive Bayes Accuracy

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.

	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%**			

Table 2. 8-fold Cross-validation Accuracy

For comparison, we tested the same method when we do not use custom features. The naive Bayes results are very similar to those in table 1. We observe a slight change in the cross validation phase. The average accuracy after cross validation equals 53.7%, which is a slight increase, but the results show that the custom features do not bring an improvement for the Naive Bayes method. This suggests that we find other features for this method.

5.3 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. However even when using Random Forest our results contain classes with the value of 0.0. These classes are not so frequent as before but still present. These are the following - *Content Discussion, Greeting, Instruction Question, Incomplete/typo, Feedback Discussion, Wrap-up, Opening Statement, General Question, Emoticon/Non-verbal*

The results from the 8-fold cross-validation are shown to average around 60% as shown 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.

	F1	Precision	Recall
Logistics	0.75	0.6	1.
Assignment Question	0.67	1.	0.5
General Comment	0.857	0.75	1.
Outside Material	0.5	1.	0.33
Content Question	0.323	0.625	0.217
Assignment Instructions	0.8	1.	0.67
Response	0.657	0.595	0.733
Accuracy	**0.611**		

Table 3. Random Forest Accuracy

	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%**			

Table 4. 8-fold Cross-validation Accuracy

Again for comparison, we test the Random Forest method without using custom features. This time the results without using custom features are slightly worse. Table 5 shows the results for the Random Forest without using custom features.

	F1	Precision	Recall
Logistics	0.75	0.60	1.
Assignment Question	0.67	1.	0.5
General Comment	0.857	0.75	1.
Outside Material	0.5	1.	0.33
Content Question	0.313	0.556	0.217
Assignment Instructions	0.8	1.	0.67
Response	0.646	0.60	0.70
Accuracy	**0.601**		

Table 5. Random Forest Accuracy without custom features

We can observe some changes in the Content Question and Response fields. Overall the accuracy when using custom features is better for one percent which is a minor improvement, but noticeable, so we can assume the custom features do improve results to some degree. We should however consider generating other features which improve the results even more. After cross validation, the results are very close for both when we use custom features and not use them, so our custom features do not seem to give us a clear advantage.

5.4 BERT

The described DistilBERT model was first trained using 100 epochs for several runs. The results far surpassed the ones from Naive Bayes and Random Forest. When training in this way, we only output the F score and accuracy. Figure 10 shows the last 100-epoch run's loss functions for the training and validation set. The accuracy score at the end of training was equal to **0.9014** and the F score was equal to **0.8649**. This shows that the model outperforms the other presented methods by a significant margin. When testing this method, only the overall scores were computed compared to the traditional methods, where we computed scores for each class. This is because the scores for the traditional methods were lower and we needed to determine the cause.

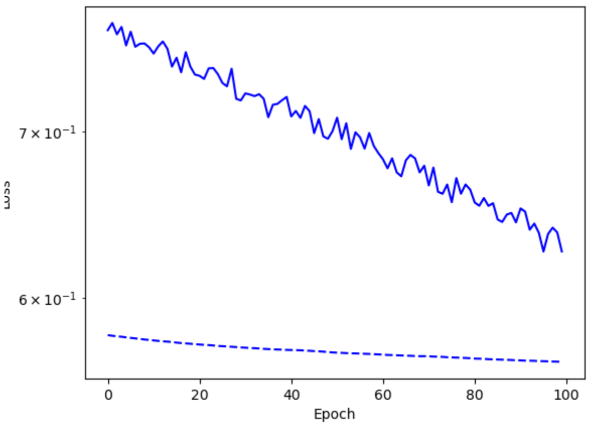


Figure 10. Loss functions for the final training run with 100 epochs

Next, we tested the trained model in a shorter run of 10 epochs to be able to perform a demonstration, so other users who use this code do not have to train the network for a large number of epochs and thus save some time. We also computed precision and recall scores along with the accuracy and F score. Table 6

	F1	Precision	Recall	Accuracy
Score	0.854	0.838	0.887	0.887

Table 6. BERT scores for the trained model

6. 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 with traditional machine learning methods and a variant of a BERT transformer neural network to solve this task with deep learning methods. 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. The deep learning approach clearly outperforms the traditional methods as the accuracy reaches up to about 90%. This shows that with the right model, the transformer methods are clearly better suited to solve this task.

For the traditional methods, we tested some custom features which should help us determine questions from other classes and determine emoticons. These features were shown to have only a minor impact on the performance, so an improvement would be to include some other features. We also included some quotes from the book data as features but they too do not contribute to better performance and the implementation of them should be reviewed. A possible solution would be to include features which place emphasis on the input messages which were typed by the two most active users. The two most active users were determined when studying the dataset to see which user contributed most to the final answer. This would give a better representation if the data and provide additional information for the classifiers. Overall, since the deep learning method gives good results, the task of classifying the IMapBook dataset is solved. The trained network could be used for other similar tasks in speech act text classification, we could also use it to filter out certain categories of chat messages automatically or create an application which helps the pupils with questions about the books, for instance in the form of a chat bot. An interesting experiment would also be to gather similar data and try to classify them in a different language to see how good the network performs in that case. There are plenty of opportunities to use the findings and classifiers from this project for either practical applications or further study.

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] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

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

[13] Sampath Kethineedi. Bert industry classification model. Available: https://huggingface.co/sampathkethineedi/industry-classification, May 2021. [Accesed on: 20.05.2021].