# IMapBook Text Classification

**Žiga Babnik, Miha Štravs, Kristian Šenk**
University of Ljubljana
Faculty of computer and information science
Večna pot 113, SI-1000 Ljubljana
zb1996@student.uni-lj.si, ms8816@student.uni-lj.si,
ks3803@student.uni-lj.si

## Abstract

In this paper we present a text classification approach, for the given data set which includes book discussions of students from several schools organized into different clubs. From the given data we first extract knowledge regarding the four target variables *Book Relevance*, *Type*, *Category* and *CategoryBroad*. Using the gained knowledge we construct a classification pipeline for each of the three given target variables. By using a simple neural network and Elmo word embbedings we managed to achieve decent results for prediction of *CategoryBroad*, *BookRelevance* and *Type* target variables with balanced precision and recall off all classes.

## 1 Introduction

We were given discussions of three Slovenian stories by primary school students. Our task was to devise a text classification approach to predict the *Book Relevance*, *Entry Type*, *Category* and *CategoryBroad* target variables of the data set.

The problem with interpreting the human language is that it is not a set of rules or binary data that can be fed into the system, from which we could understand the context of a conversation. With new algorithms and powerful processors, we have made a significant advancement in Natural Language Processing (NLP). NLP has been around for quite some time for languages like English, but for Slovene, it was popularized in recent years. Slovene is one of the harder languages to learn, firstly since it contains not only singular and plural word forms but also dual word forms and secondly, it contains declension which can cause many problems even for humans. One of the main challenges in the case of the given data set will be the use of slang.

In section [2] we briefly present the field of text classification and papers most relevant to our approach as well as a text classification survey paper.

In section [3] we present the data used in the paper in detail. Furthermore we present some interesting statistics calculated from the data, which we later use also for classification.

In section [4] we present our approach in detail. Firstly in section [4.1] we present how the data is preprocessed, then in section [4.3] the three different types of word representations used in our approach, lastly in section [4.4] we present how we designed the classification for each of the three target variables.

In section [5] we present the results of our approach. We use two approaches, in section [5.1] we present the results of using logistic regression and in section [5.2] we present the results of using a neural network.

## 2 Related work

Text classification is a well researched field, the main topics discussed in papers related to text classification are word representations and machine learning models used for prediction.

The field of word representations has in recent years seen many breakthrough technologies emerge. In the paper *Efficient Estimation of Word Representations in Vector Space* (Mikolov et al., 2013) two models for word representations are presented, these models still achieve reasonable results nowadays. First the *Continuous Bag-of-Words model* (CBOW) is introduced, which uses the neighboring words as the input for prediction of the current word, after which the *Skip-gram model* (Skip-gram) is introduced, which uses the current word to predict the neighboring words. A newer approach is presented in the paper *Bag of Tricks for Efficient Text Classification* (Joulin et al., 2016a), where a model named *fasttext* is introduced. The *fasttext* model is similar to the CBOW model, while allowing for very fast train-

ing on large data sets. One of the newer approaches to construct word representations is presented in the paper *Deep contextualized word representations* (Peters et al., 2018), which introduces the *Embeddings from Language Models* (ELMo) representation. ELMo representations are contextual, deep and character based, allowing the representation to take into account syntax, semantics and how the uses of the words vary across linguistic contexts.

Related work is presented much more thoroughly in survey paper *Text Classification Algorithms: A Survey* (Kowsari et al., 2019), where various word representations and machine learning models, such as the ones presented in the previous paragraph, are used and compared using many evaluation methods.

## 3 Data

We were given data collected from an online discussion forum, on which primary school students were able to discuss and comment on three different books, that they previously read. Each discussion entry contains various annotations about the date and place of the entry, user who submitted the entry and the actual class that represents the target variable for the given text classification.

### 3.1 Basic data statistics

For the fist classification task, where the aim was to find entries relevant to the book discussion two classes were given, the values of which are shown in Table 1, while the distribution is shown in Table 2.

| Book relevance | |
|---|---|
| Class | Meaning |
| YES | Value is relevant to the book. |
| NO | Value is not relevant to the book. |

Table 1: *Book Relevance* target variable explanation

| Class | Count |
|---|---|
| No | 2155 |
| Yes | 1384 |

Table 2: *Book Relevance* target variable class distribution

For the second classification task, where the aim was to predict the type of entry, we were given three labels. The explanation of each class is shown in Table 3, while the distribution is shown in Table 4.

| Entry type | |
|---|---|
| Class | Meaning |
| QUESTION | Entry is a question. |
| ANSWER | Entry is answer to any question. |
| STATEMENT | Entry is sentence that is not question or answer. |

Table 3: *Entry type* target variable explanation

| Class | Count |
|---|---|
| Statement | 1710 |
| Answer | 1155 |
| Question | 672 |

Table 4: *Entry type* target variable class distribution

For the final classification task, where the aim was to predict the category of the entry, five classes were given, where each class was further divided into sub-classes. The explanation of the classes and sub-classes is shown in Table 5, while the distribution is shown in Table 6.

### 3.2 Word count

Simple statistics can be used in text classification to boost results. Word counting is one of the simpler statistics, where we simply count the occurrences of words per class of the given target variable. We wish to apply word counting to the *Category* variable to see if detecting certain words in the text is indicative of a particular class of the target variable.

We first performed tokenization on the text, to extract individual words from the text. The word counting was then performed by first counting word occurrences for the whole text, as well as per each class. The word counts for each class were then normalized using the global word count. Words that occur less than 20 times, were not considered, as well as words that included only one letter. As the word counts were normalized the results are presented using probabilities that a certain word represents a given class of the *Category* target variable. To ensure the results are easier to read we display only the top ten words for each of the classes.

Figure 1 shows the ten most probable words that appear under the *CO* class. The two words that

| Category | | |
|---|---|---|
| Class | Sub-class | Meaning |
| CHATTING | CG | Greeting |
| | CB | Chatting about books |
| | CE | Encouraging others to join the chat. |
| | CF | Chatting about how they feel. |
| | CO | Chatting about other thing. |
| | CC | Being mean |
| SWITCHING | S | Talking about where in the system they currently are. |
| DISCUSSION | DQ | Discussion question. |
| | DE | Posing a question that directly encourages further discussion. |
| | DA | Answering the discussion question directly. |
| | DAA | Answering a question or commenting on the answer of someone else. |
| MODERATING | ME | Encouraging the students. |
| | MQ | Asking questions relevant to the discussion question. |
| | MA | answering the discussion question directly |
| | DAA | Answering the students questions |
| IDENTITY | IQ | Identity question |
| | IA | Identity answer |
| | IQA | Combination of previous tags |
| OTHER | O | anything that does not make any sense (typos ...). |

Table 5: *Category* and *CategoryBroad* target variable explanation

| Class | Count |
|---|---|
| Chatting | 1430 |
| Discussion | 1197 |
| Identity | 416 |
| Other | 282 |
| Moderating | 177 |
| Switching | 38 |

Table 6: *Category* target variable class distribution

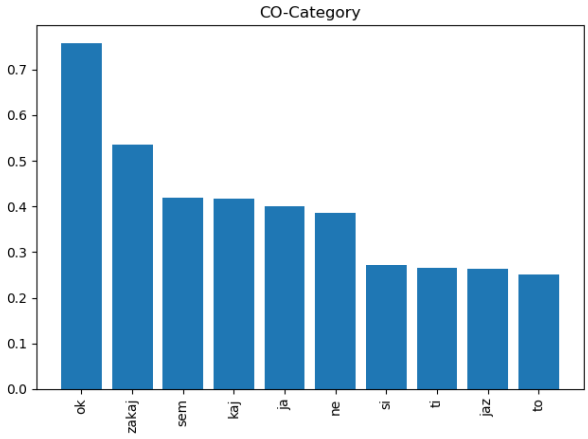


Figure 1: Ten most probable words to appear in the *CO* class of the *Category* target variable, ordered by decreasing probability

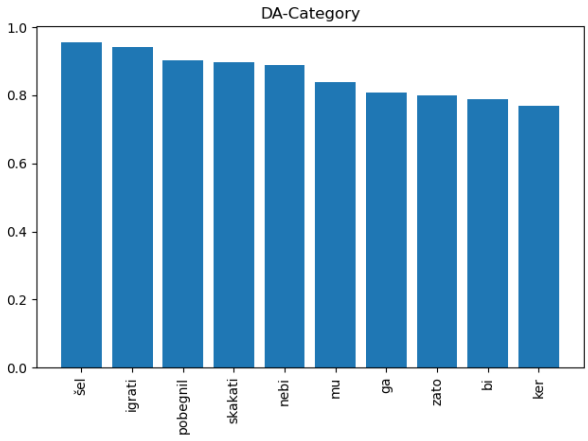


Figure 2: Ten most probable words to appear in the *DA* class of the *Category* target variable, ordered by decreasing probability

signify with a high probability that a message is classified as *CO* are `ok` and `zakaj`.

Figure 2 shows the ten most probable words that appear under the *DA* class. All the top ten words seem to be a good indicator that the message belongs to the *DA* class.

Figure 3 shows only six most probable words that appear under the *IQ* class. Since we remove words that appear less than 20 times overall the result can contain less than ten words. The word `kdo` seems to signify with high probability that the message is classified as *IQ*.

From these results, it is clear that using simple word counting can add knowledge to our model,

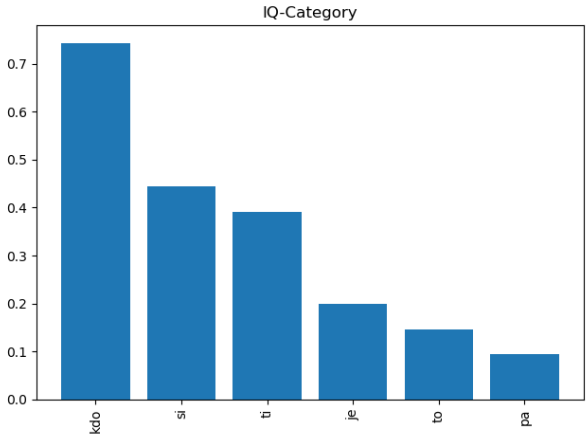


Figure 3: Six most probable words to appear in the *IQ* class of the *Category* target variable, ordered by decreasing probability

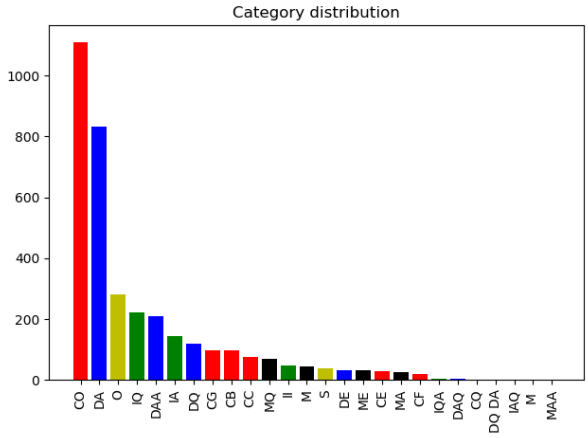


Figure 4: Descending order of distribution of classes for the *Category* target variable, where each *Category Broad* variable is assigned a unique color

since some words indicate with high probability the class of the *Category* target variable of a message.

### 3.3 Target variable dependence

Since the data includes four target variables, we are interested if they are independent. If it turns out there exists some dependence we could use a prediction of one variables to improve the prediction of a different dependent variable. Since *Book Relevance* and *Type* are quite simple in regards to number of tags, we would like to see if they could assist in predicting the *Category* target variable.

Firstly we present the prior distribution of the *Category* target variable in Figure 4. We see that the distribution is unbalanced, with the *CO* and *DA* class being most frequently represented, while some other classes have close to zero appearances. From this we could construct a classifier that only guesses the top four or five classes and achieves fairly good results.

We observed how the *Category* distribution changed when we first split the data into all combinations of *Tag* and *Book Relevance* classes. Figure 5 shows the results. While quite a few combinations of classes prove to be useless since they are underrepresented, some provide useful information for classifying the *Category* target variable. Among the most informative are *NO+S* meaning the message is not relevant to the book and is a statement, the messages of this kind are most likely to represent the *CO* class, *Yes+A* meaning the message is relevant to the book and represents an answer, the messages of this kind are most likely to represent the *DA* class. Both of these results make sense when talking about the meaning of the *Category* target variable, they most likely represent.

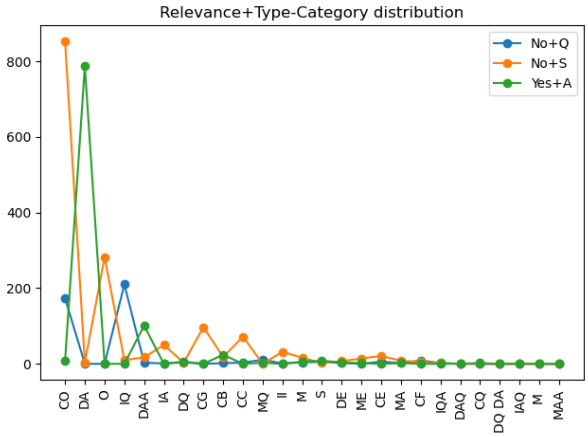


Figure 5: Descending order of distribution of classes of the *Category* target variable, where the data was first split by all *Book Relevance* and *Type* class combinations after which the splits containing less than 10% of all data were removed

From these results it is clear that the *Book Relevance* and *Type* target variables contain information about the *Category* target variable.

### 3.4 Category target variable class sequence dependence

The data is presented as a sequence of messages each with different target variable classes. As such we are interested if there are any class sequence patterns which could assist us in classification. First we divided the data by *Book Clubs* and *Topics*

and than sorted all messages by their time stamps. From the sorted messages we were able to extract pairs of classes, where the first class in the pair is followed by the second class. By constructing a Markov chain where all classes represent nodes and all pairs represent links between these nodes, we can visualize the class sequence dependencies. Furthermore we can count the number of all pair occurrences and encode them as link weights, normalizing them allows us to represent the probability of the second class following the first class. For the final step we remove all links that occur less than 1% of times compared to all links in the Markov chain, and all isolated nodes, this way the final network will be more clear and present only the most important information.

Figure 6 shows the Markov chain of sequences of the *Category* target variable, where we take into account only one previous class. From the results, we can determine that most classes are followed by that same class, while the probabilities of switching between classes is somewhat lower. To take into account more information we construct a network where we take into account two previous classes.

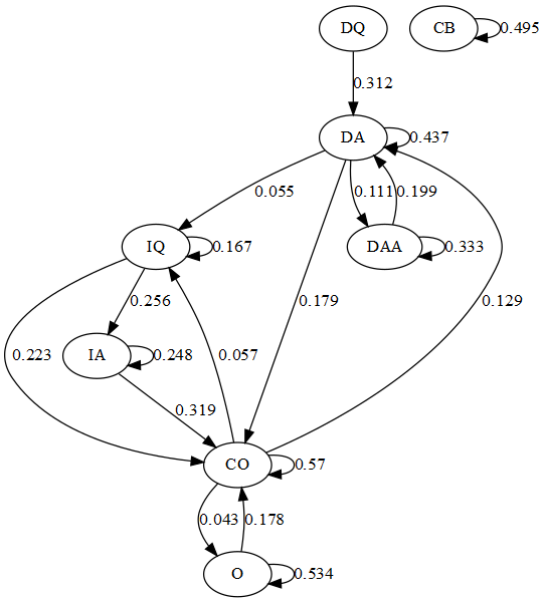


Figure 6: Markov chain of the *Category* target variable sequences, where we take into account only one previous class, with normalized weights and with links that occur less than 1% of the times removed

Figure 7 shows a Markov chain constructed when we take into account two previous classes. From the results we can see that if we encounter two same classes in a row the probability of encountering that same class a third time is quite high.

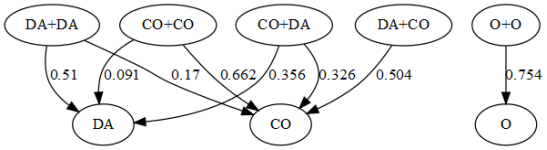


Figure 7: Markov chain of the *Category* target variable sequences, where we take into account two previous label, with normalized weights and with links that occur less than 1% of the times removed

## 4 Methods

In the following section we present our text classification approach. We discuss each step in detail, starting with data preprocessing, where we discuss how the data was transformed to better suite our pipeline. We then present how we detected gibberish text, and later present all the vector representations used in the approach. Finally we propose our classification approach for the four given classes *Book Relevance*, *Entry Type*, *Category* and *CategoryBroad*.

### 4.1 Data preprocessing

We first applied some basic preprocessing steps on the given data, based on the type of vectorization done in the next step. For bag-of-words we first transformed the data to strings and applied gibberish detection, that we present later. For Elmo and Wiki word embeddings we used tokenization, that extract individual words from sentences, add tags and multiply words that define interrogative sentences, such as *who* and *why*. Both steps were performed using the *reldi-tokeniser* and *reldi-tagger* described in the paper *Multilingual Text Annotation of Slovenian, Croatian and Serbian with WebLicht* (Ljubešić et al.). Since we are dealing with really short texts containing informal language we did not remove any stop words.

Now we have transformed our data into vectors using the bag-of-words approach and word embeddings. Tags were filtered so that only a select few remained, they were then vectorized using the bag-of-words approach and appended to the message vector. Topics were also appended using the bag-of-words approach. Lastly for *Category* classification book relevance and gibberish word detection were used, both of which were predicted and appended to the message vector. For

the case of book relevance, if the message was not relevant $-1$ was appended otherwise, if the message was relevant 1 was appended. For the case of gibberish detection $-1$ was appended if the message contained gibberish words, otherwise 1 was appended.

### 4.2 Gibberish detection

We created a gibberish detector, which looked for sentences with one word, very long words, many duplicated letters in a word, emojis and numbers in the middle of the string. We then trained a logistic regression model based on previous predictions and applied it to better predict the *Category* target variable of the sentence.

### 4.3 Vector representation

As mentioned in the paper *Bag of Tricks for Efficient Text Classification* (Joulin et al., 2016b) the use of word vector embeddings can boost performance on problems with small amount of data. We represented each sentence by summation of individual word vectors of a sentence, which were represented in the form of bag-of-words (BOW), word2vec embeddings and Elmo embeddings.

We used contextual embeddings proposed in the paper *High-Quality ELMo Embeddings for Seven Less-Resourced Languages* (Ulčar and Robnik-Šikonja, 2019), since context holds valuable information for our classification tasks. For comparison we also used fast text embeddings trained on Wikipedia pages found on their web-page. Since the data includes informal texts, mistakes might be a common occurrence. For this reason we used the pymagnitude library presented in the paper *Magnitude: A fast, efficient universal vector embedding utility package* (Patel et al., 2018) for querying the embeddings, since the library produces similar vectors even in the case of typos. It also includes lazy loading which is more RAM friendly.

### 4.4 Classification

Once the messages were represented as vectors we devised the final classification step of our approach. The vector representations presented previously were fed as input into a logistic regression model and a simple neural network model. Since some classes were underrepresented compared to others oversampling was used. To see the effects of oversampling a model with and without oversampling was trained. Square root of the class precision from the training data was then used as a weight when calculating class probability for that model. Weighted probabilities were summed and the class with the largest probability was then chosen as the target variable prediction.

## 5 Results

In the following section we present results gained using the proposed approach. For the given target variables *Book Relevance*, *Entry Type*, *Category* and *CategoryBroad* we have chosen two benchmarks. F1 micro, which shows us which embeddings give an overall better precision and recall and F1 macro, which favors models with better representation of smaller classes.

### 5.1 Logistic regression classification

The first classifier used was logistic regression. We decided to use the sklearn implementation with a fixed random state for test repeatability and a standard Limited-memory BFGS (L-BFGS) solver. As seen in Table 7 word vector embbedings did give us a small boost. The most notable difference was with F1 macro score for the *Category* and *CategoryBroad* target variables, which tells us that embbedings helped the model to adapt more to the less represented classes. Between different types of words embeddings there were no notable differences with one occasionally outperforming the other by a small margin.

| Word vector model | micro | macro |
|---|---|---|
| | Type | |
| Bag of words | 0.76 | 0.75 |
| **Fast text wiki** | **0.78** | **0.78** |
| High-Quality Elmo | 0.77 | 0.77 |
| | Book relevance | |
| Bag of words | 0.85 | 0.84 |
| **Fast text wiki** | **0.86** | **0.85** |
| High-Quality Elmo | 0.85 | **0.85** |
| | Category | |
| Bag of words | 0.60 | 0.33 |
| Fast text wiki | 0.61 | **0.37** |
| High-Quality Elmo | **0.63** | 0.36 |
| | Broad Category | |
| Bag of words | 0.69 | 0.49 |
| Fast text wiki | **0.73** | 0.60 |
| **High-Quality Elmo** | **0.73** | **0.61** |

Table 7: F1 micro and macro scores for all given target variables using a logistic regression model

### 5.2 Neural network classification

For the neural network we also used the sklearn implementation. The network had one hidden layer with one hundred nodes and the ReLu activation function. While we did get mostly similar results compared to the logistic regression model there was a notable increase in performance when using the Elmo word vectors. We can see the results in Table 8.

| Word vector model | micro | macro |
|---|---|---|
| | Type | |
| Bag of words | 0.75 | 0.73 |
| Fast text wiki | 0.78 | 0.78 |
| **High-Quality Elmo** | **0.79** | **0.79** |
| | Book relevance | |
| Bag of words | 0.85 | **0.85** |
| **Fast text wiki** | **0.86** | **0.85** |
| **High-Quality Elmo** | **0.86** | **0.85** |
| | Category | |
| Bag of words | 0.60 | 0.34 |
| Fast text wiki | 0.64 | **0.41** |
| **High-Quality Elmo** | **0.65** | **0.41** |
| | Broad Category | |
| Bag of words | 0.69 | 0.54 |
| Fast text wiki | 0.72 | 0.60 |
| **High-Quality Elmo** | **0.77** | **0.65** |

Table 8: F1 micro and macro scores for all given target variables using a neural network model

### 5.3 Class representation

F1 scores were calculated for each data point we tried to predict. Scores were calculated for results gained using the neural network model and Elmo word embeddings.

Figure 8 presents results for the *Category* target variable. We managed to get good F1 score for *DA* and *IQ* classes and somewhat lower scores for *O*, *CG* and *M* classes. Considering that we had more than twenty classes to predict the results are not acceptable and the model could not be used for wide *Category* target variable prediction.

Figure 9 present results for the *CategoryBroad* target variable. We see that our model achieved decent F1 scores for all classes, of around 80%. The *S* class achieved a very poor F1 score, the reason for this is most likely due to low representation of the class in the data.

Figure 10 present results for the *Entry Type* target variable. We achieved a similar F1 scores for

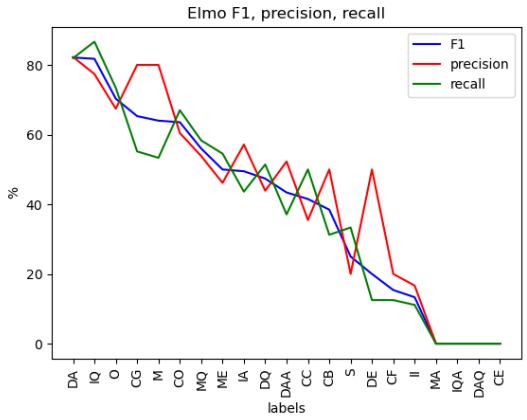


Figure 8: F1, precision and recall for the *Category* target variable using a neural network model and Elmo word embeddings

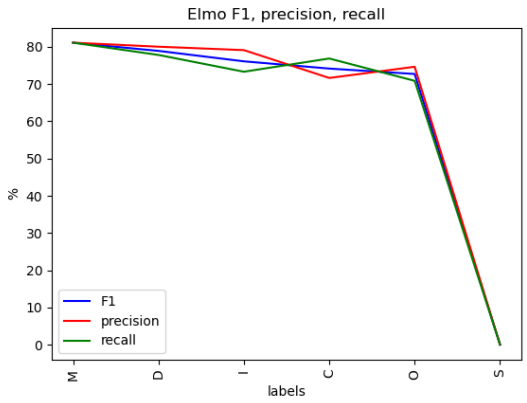


Figure 9: F1, precision and recall for the *CategoryBroad* target variable using a neural network model and Elmo word embeddings

all classes, with *Q* having a higher recall since messages in the form of questions were easier to differentiate from statements and answers.

Figure 11 presents results for the *Book Relevance* target variable. Both classes achieved decent scores with the *No* class being slightly better, probably due to having more representation in the data.

## 6 Discussion

In this paper we presented a text classification approach, that tries to classify short texts from the given data set, containing discussions of three books. Our approach was centered around using three different word vector embbedings. With Elmo embbedings and a neural network we managed to predict the *Entry Type*, *Book Relevance*

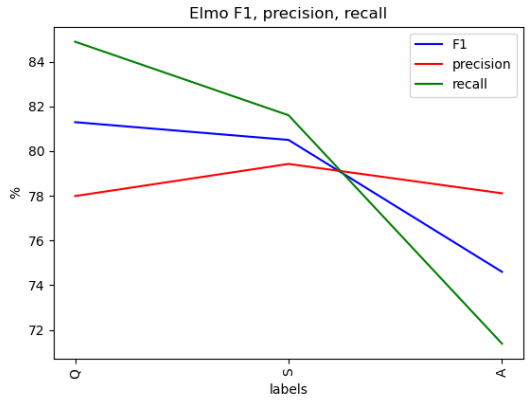


Figure 10: F1, precision and recall for the *Entry Type* target variable using a neural network model and Elmo word embeddings

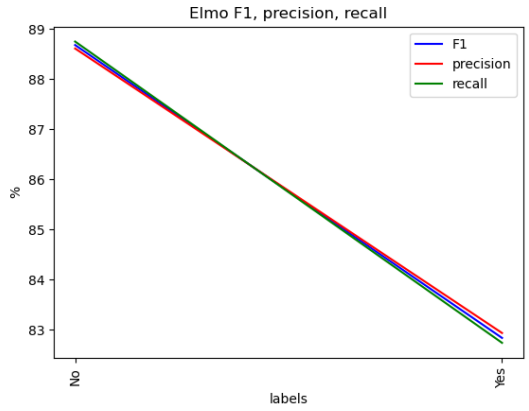


Figure 11: F1, precision and recall for the *Book Relevance* target variable using a neural network model and Elmo word embeddings

and *CategoryBroad* target variables with decent results, which had balanced scores for all classes. We managed to solve uneven class representation to a certain degree by oversampling underrepresented classes. This did not improve the prediction of the *S* class of the *CategoryBroad* target variable and quite a few of the classes from the *Category* target variable. The given problem could be solved if we would find some unique properties for messages of less represented classes of the *Category* target variable. Perhaps as noted in section 3.4 using class predictions of previous messages might help, but that would require decent classification of previous messages to prove useful.

## References


Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016a. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016b. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information*, 10(4):150.

Nikola Ljubešić, Tomaž Erjavec, Darja Fišer, Erhard Hinrichs, Marie Hinrichs, Cyprian Laskowski, Filip Petkovski, and Wei Qui. Multilingual text annotation of slovenian, croatian and serbian with weblicht.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ajay Patel, Alexander Sands, Chris Callison-Burch, and Marianna Apidianaki. 2018. Magnitude: A fast, efficient universal vector embedding utility package. *arXiv preprint arXiv:1810.11190*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Matej Ulčar and Marko Robnik-Šikonja. 2019. High quality elmo embeddings for seven less-resourced languages. *arXiv preprint arXiv:1911.10049*.