Reuters Corpus Topic Classification

Heli Huhtilainen heli.huhtilainen@helsinki.fi
Jaakko Vilenius jaakko.vilenius@helsinki.fi

Abstract

This is a report for the Helsinki University Introduction to Deep Learning course project. The project is about topic classification on the Reuters corpus that contains news articles and is labeled with the topics of the articles. The task is multi-label classification, so there can be several topics associated with each document. The deep learning model that is used in this task has been implemented using Huggingface BERT transformer trying different text truncations and loss function tuning. The developed models are evaluated using precision, recall and F1-score. The final model 13 achieves 0.843 F1-score with the validation data set.

1 Introduction

We chose text classification as our project topic because this spring we have also other courses that handle natural language processing. We thought that it is good to get more practical experience on the subject. We decided to try out transformers as this was a technique that might produce good or even state-of-the-art results in this kind of problem, and it was also something that we were curious to try out for the first time. We found some good instructions on how to use a BERT transformer for this kind of task in the post by Patel, and some of the code used in this project is from this source. The base model used this task is a Huggingface transformer from Wolf et al. (2020).

The project scripts and notebooks can be found here:

`https://github.com/apndx/ReutersDocLabeler`

2 Preparing the data

The training data and testing data were provided as multiple zipped XML files. Each XML file contained a single Reuters' news item with associated topic codes among other data. We extracted the content from the `headline` and `text` tags and concatenated them as our input. We also extracted the corresponding topic codes from `codes` tags having `class = "bip:topics:1.0"` within `metadata` tags. In addition the id of the news item was also extracted.

In order to minimize the input size we dropped all stop words using `stopwords` from `nltk.corpus` and lemmatized the remainder with `WordNetLemmatizer` from `nltk.stem`.

In the source data associated topic codes were assigned to each news item. The source files also contained a list of all possible topic codes. We used the list of all the topic codes to create 'multiple-hot' vectors for each row of our input data. The length of the vector was set to the length of the topic list with `1`'s indicating that a corresponding topic code was associated with the news item while other elements of the vector were assigned a `0`.

The input data was then stored on disk for further processing. For easier data exploration, a list of the topic codes of each news item were also stored on each row of the file.

The same process was used for preparing the test data. Naturally, since no topic codes were provided in the test data, all vectors were all zeros and the topic lists were empty.

3 Exploratory data analysis

After the data preparation step, we had a data frame with the input text and the target topics codes as both in multiple-hot vector form and an array. There were 299 773 documents and 126 different topics, which of 103 were used in the dataset. An overview of the dataframe can be found in figure 1.

The data presented some challenges. As the targets were arrays, there were a total of 8326 dif-

	text	target	codes	length
0	169/04 service commission implementation amend...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[G15, GCAT]	61
1	poultrymeat egg committee application 0206 192...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[G15, GCAT]	137
2	sulphur river ..... shell (lower petro-canada...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[M14, M143, MCAT]	51
3	sulphur 162.00 ..... 25.76 171.00 27.19 ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[M14, M143, MCAT]	53
4	6423.70 bank 300 111.75 lo -0.70 `` worth thin...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[M11, MCAT]	115
...	...	...	...	...
299768	paris m6 le first-half 31 director soft franc ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[C32, CCAT]	176
299769	paris m6 service 12.8 audience 's franc month ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[C152]	96
299770	soon including service commission wanfang tech...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[GCAT, GSCI]	74
299771	planning partnership regulatory 31 ufa without...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[C32, CCAT]	166
299772	mediamanager service told director feed would ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[C22, CCAT]	162

299773 rows × 4 columns

Figure 1: Overview of the dataset

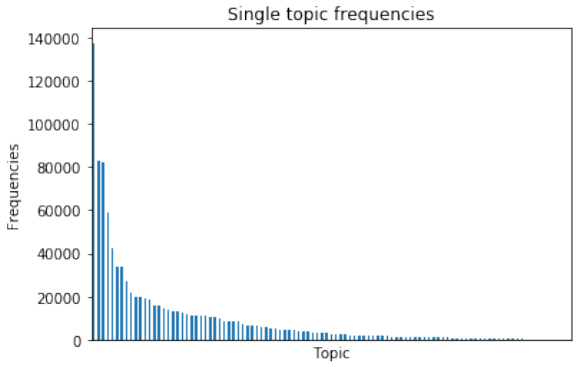

Figure 2: Some topics were common, but most were quite rare.

Figure 3: This word cloud visualization gives an idea of the relative amounts of topics in the full dataset.

ferent topic combinations, of which only five had over 10 000 examples, but 7279 combinations had less than ten examples. Over half (4596) of the combinations had only one example. The distribution of topic combinations is in figure 4. The single topics showed the same kind of polarization: some topics were a lot more commonly represented than others, as figures 2 and 3 show.

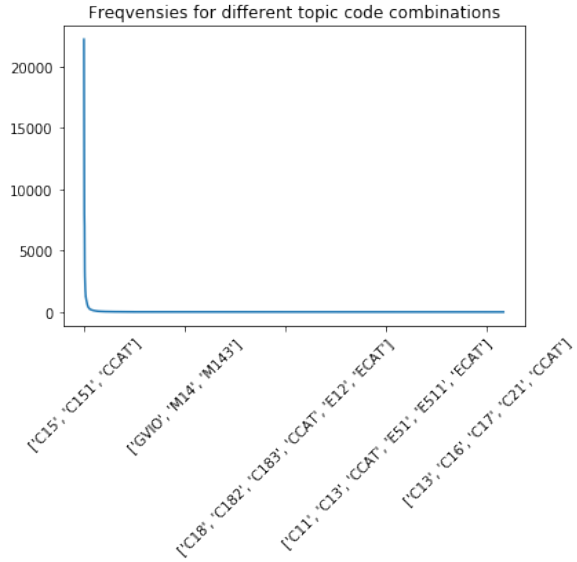

Figure 4: Some topic combinations are common, most are rare. There are 8326 different combinations in total, x-axis shows some examples.

There was also some variation in the input text length. After the data was prepared, the average length of an input was 113 words, with stdev of 74. Longest input was 2758 and the shortest only 6 words long. As figure 5 shows, most of the inputs

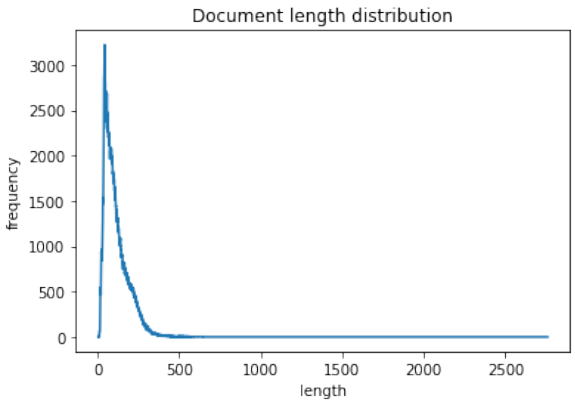

Figure 5: The document length distribution shows that most documents are under 500 words long.

are under 500 words long.

4 Preprocessing the data

Even when the goal was for the model to learn to recognize single topics, the splitting the data with stratify demanded special treatment for the target rows that had only a single example of a certain topic combination. These rows were all kept in the training examples. Stratifying was done to make the train and development set divided in a balanced manner. The data was divided using `train-test-split` provided by `sklearn`. In model development, the train/dev/test split was 80/10/10, and for the final model 13 it was 90/10.

Huggingface `BertTokenizer` was used to encode inputs, token types, attention masks, and labels. These were then transformed into tensors. Dataloader batch iterators were then made for train, development and test. Dataloader objects proved to be handy, as they could be saved and loaded when training, validation or testing started.

5 Model development

5.1 Hyperparameters

We used pre-trained `bert-base-uncased` as our model base, and the idea was to fine-tune it to classify the Reuters corpus topics by training it with the corpus data. In training, we used `AdamW` as the optimizer with a 0.00001 learning rate. As the loss function, we used `BCEWithLogitsLoss`, which was mentioned by Patel as suitable for multilabel classification. In the beginning, additional hyperparameters were not added, as it seemed like a good idea to first set a baseline for the model performance. The used hyperparameters are in table 1.

Model	Input Length	Pos Weight	Split	Epochs
9	256	not used	80/10/10	4
10	512	not used	80/10/10	4
11	512	1	80/10/10	4
12	512	0.25	80/10/10	5
13	512	not used	90/10	4

Table 1: Comparing model hyperparameters.

5.2 Training

To be able to locally debug the code and see if the model training would work as it should, we made a mini dataset where the input was truncated to 64 words, and that had 19584 rows for training. It soon became clear that even with this small set local development with only cpu was too slow and memory-intensive. We changed to use the CSC puhti server, where we did the rest of the training, validation, and testing.

We did some initial tests with batch size 48 but soon decided to use size 32 as it did not need as much memory. For memory saving reasons we first tried the model with input that had been truncated to 256 words, but later changed to use 512 word truncation to see if this would improve the model.

Figure 6 shows precision, recall, f-score and loss during the training of model 13. There seems to be just marginal improvement in loss after epoch 2.

Many of the challenges confronted in training the models related to the large data and model sizes. For example the train-dataloader for model 13 is 3.6 GB, and the finished model file is 438 MB. The models were quite heavy to train, the training took a lot of memory and time. The training was done with 1 node of CSC puhti GPU. For model 9 (input 256) it took 53 min per epoch and model 10 (input 512) it took 113 min. For the final model 13 (512 input and larger train data) it took 126 min per epoch.

6 Validation

The model validation results are in table 2. The models 9 and 10 showed some good overall scores in both precision, recall and F1. When looking at the results closer, it was found that model 9 had 17 totally unlearned topics, and model 10 had 13 (figure 8). We decided to keep the longer 512 input length in the next models as it had slightly better

Figure 6: Left: Cumulative precision, recall and F-score for the whole training run of model 13. Right: Batch loss for each epoch of the training run of model 13

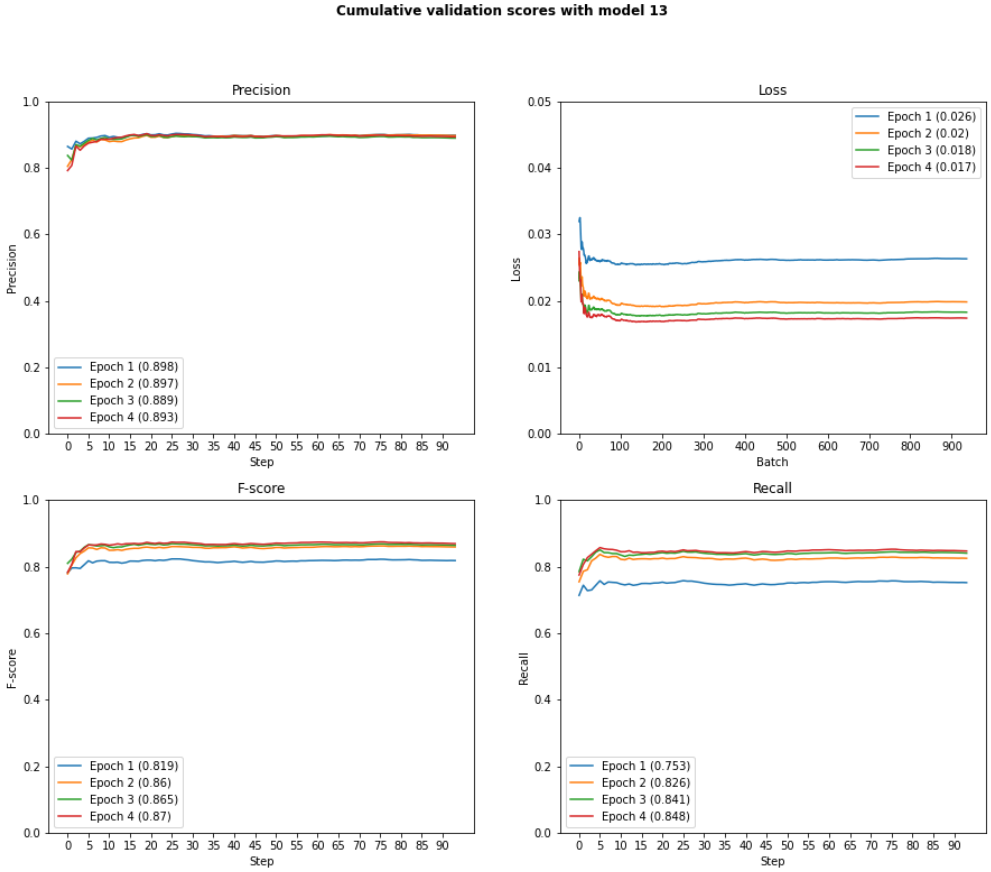

Figure 7: Cumulative precision, recall and F-score and batch loss for each epoch of the validation of model 13

Model	Precision	Recall	F1	Loss
9	0.875	0.816	0.844	0.018
10	0.894	0.816	0.853	0.018
11	0.370	0.951	0.533	0.099
12	0.611	0.883	0.722	0.041
13	0.883	0.806	0.843	0.017

Table 2: Comparing model validation results.

results and fewer unlearned topics. Also, to improve recall and learn more topics we tested adding pos weights to the loss function. This was used in model 11 and 12, and resulted in good improvement in recall but even bigger drop in precision. Also F1 suffered from this addition. In the final model 13 we used 90 / 10 data split to use more of the data for training. This did not have a big effect on the results.

For most of the models we trained for 4 epochs. This seemed to be enough in most cases. When pos weights were introduced, we added one more epoch for model 12, as it seemed that the training was slower with this feature. For this model the extra epoch was beneficial.

For the sake of comparison we stored the model parameters after each epoch and did validation with each version. Figure 7 displays a comparison of results with each epoch version of model 13.

7 Testing

Part of the project was to test the model with unseen test data where the targets were hidden. Each team would submit their results so they could be compared. After we received the test set 33 144 examples, we did the same preparing and preprocessing steps to it as to the model developing data set. In the model validation, only the scores were calculated, so we wrote a test script to get the lists of predicted topics for the test examples.

8 Error analysis

8.1 Unlearned topics

The models had some unlearned topics, and figure 8 shows the topics model 10 did not learn. We analysed a bit the most common of them. Models 9 and 10 both failed to learn `human interest` topic. In all of the data there were 975 examples of this topic, and in the validation data there were 88. None of these validation examples were labeled as belonging to this topic.

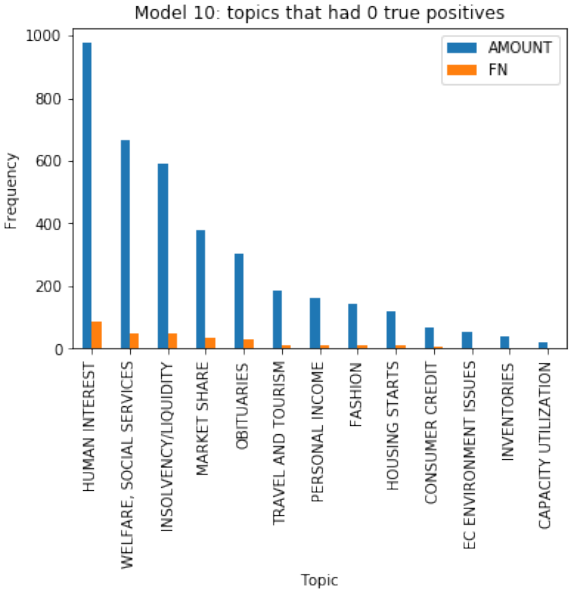

Figure 8: These topics have examples in the train and validation, but remain unlearned. The amount bar tells the count of topics in the full data and false negatives tell the validation example count.

Figure 9: Example of unlearned human interest topic input, also belongs to government/social topic.

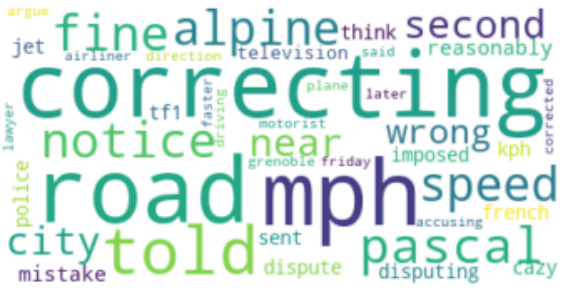

Figure 10: Example of unlearned human interest topic input, also belongs to government/social topic.

Figure 11: Example of unlearned human interest topic input, also belongs to government/social and international relations topic.

Figures 9, 10 and 11 have word clouds made of some examples of the inputs from the validation set. All of these examples share the `human interest` and `government/social` topics. Figure 11 also has a third topic, `international relations`. Just by looking at these examples, it is hard to say why they were not recognised as `human interest`. It could be that the category is quite vague, and at least on first glimpse these examples do not seem to share many common words. Maybe further analysis would reveal some patterns, though it might be hard or even impossible to find out how exactly how the model has learned to classify topics, and why it has not succeeded in this case.

8.2 Distribution of the predicted results

We analyzed which were the top 25 topic codes in the training data and in our prediction results. Figure 12 shows the relative frequencies of these topics. Relative frequency here means the number of times the topic code was found divided b the sample size. No frequency is shown for a topic code if it was not among the top 25 topic codes of that dataset.

E.g., `C13` was among the top 25 topic codes in the training set with a frequency of 0.044 but it was not among the top 25 topic codes of the prediction results, therefore, there is no frequency shown. That doesn't mean the the topic code was not at all in the predicted result, it just wasn't in the top 25. An opposite example is topic code `M143`, which was among the top 25 predicted topic codes but not in the top 25 topic codes of the training set. Overall it seems, at least for the most common topic codes, that the distribution of the prediction results is quite similar to the distribution of the training dataset, when it comes to the relative frequency of the topic codes.

9 Conclusions

In this project a multilabel topic classification was done for Reuters corpus using pretrained Huggingface BERT transformer model as a base and training it with the corpus data. The results were quite good with quite a small amount of tuning, as the final model 13 got 0.843 F1.

If there had been more time, it would have been nice to try some more hyperparameter tuning for the model, or maybe try to replicate some of the rarer examples to get more data to learn the harder topics.

The multi-label aspect made this task especially interesting, and it made analysing the results also harder as there were many dimensions. It was also good to get more familiar with evaluating results with precision, recall and F1. All in all it has been a good learning experience of how to handle the logistics of this kind of a project.

References

Ronak Patel. Transformers for multi-label classification made simple.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

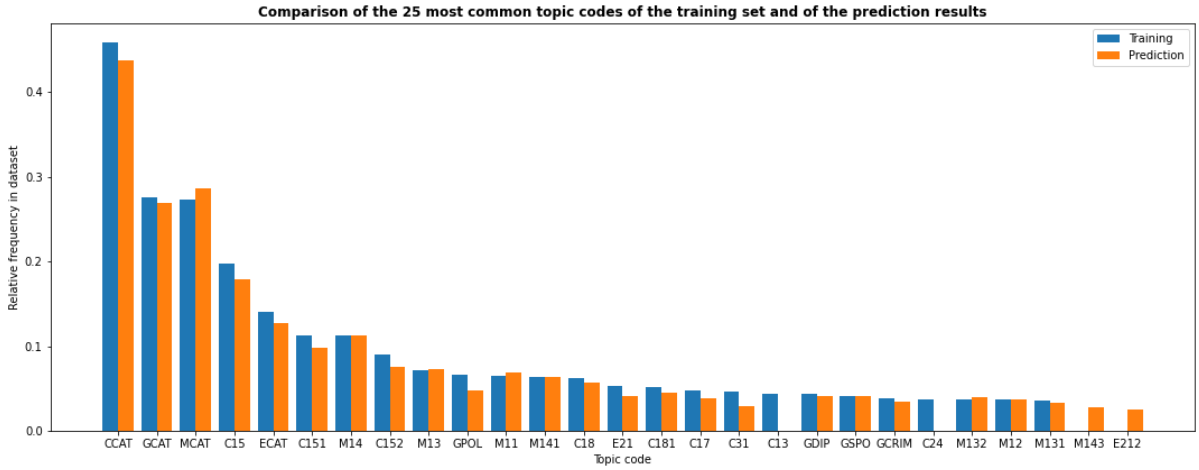

Figure 12: Comparison of the top 25 topic labels of the training data and of the prediction results.