Document classification with NLP methods

**Abstract**

This documentation covers the work of Győri Mihály, Kristóf Varga and Zsemle Zsolt for BMEVITMAV45 Deep Learning in Practice with Python and LUA. Our elected topic is Document classification with NLP methods. Our implementation includes data cleaning, text preparation with tokenization and encoding with BERT, model training of an LSTM classification model. Training started with the first 5 most common categories, then we tested with 7 and 10 categories, the best implementation predicts 5 categories. We augmented our training process with hyper parameter optimization in order to improve training time, resource usage and results. With our final optimized model we achieved 67.12% precision with 5 categories.

**Introduction**

Unstructured content is all over, such as emails, chat discussions, websites, and social media. All things considered, it’s difficult to extricate esteem from this information unless it’s organized in a certain way. Doing so utilized to be a troublesome and costly prepare, since it required investing time and assets to physically sort the information. Content classifiers with NLP have demonstrated to be a extraordinary alternative to structure textual data in a quick, cost-effective, and versatile way. By utilizing Natural Language Processing (NLP), text classifiers can automatically analyze content and assign categories or labels based on its content. The most common use cases for automatic text classification are sentiment analysis, topic detection and language detection

Our implementation is for topic detection, the base task is to determine topic of news articles. At first, we classify the 5 most common topics, then we aim to widen the array of topics while maintaining the precision of the model.

State-of-the-art implementations with the lowest error score use XLNet‎[1] or BERT-ITPT-FiT‎[2] which are both build on Hugging Face Transformers, but other methods like LSTM and CNN networks also achieved competitive scores.

**Implementation**

**Data preparation**

For our implementation we used AG's corpus of news articles. “*AG is a collection of more than 1 million news articles. News articles have been gathered from more than 2000 news sources by ComeToMyHead in more than 1 year of activity. ComeToMyHead is an academic news search engine which has been running since July 2004*.”‎[1]

The whole corpus consists of 14 unique categories which are shown in Table 1:

Table 1. Distribution of topics in the corpus

After downloading the data, we started our preparation. First, we dropped lines which were faulty or had missing values. We decided not to use source field, it contains the websites the articles from, which would defeat the purpose of this task. Our initial implementation only used the five most common topics: world, entertainment, sports, business and top stories.

As our next step we defined a tokenization and a vocabulary building method. Our elected method uses WordPiece‎[2] which is based on BERT tokenizer form HuggingFace. Our method tokenizes the title and the description and removes all non-alphabetical characters from the text and words which are shorter than three characters. From this cleaned data we built our vocabulary. Tokenized text then encoded with encode\_plus. We set maximum length to 64 with extended padding, in case some of our inputs are shorter and created a vocabulary from the encoded data as well.

**Model training**

Model training started with splitting the data. In order to have sufficient data for training, validation and testing we used 70% of all data as training and the remaining 30% has been split in two equal parts and all labels have been turned into float32 categorical values.

We defined a function which builds the model. Our model uses LSTM which seemed to be the best solution for this type of task based on other implementations we researched‎[3]‎[4]‎[5]. The fundamentals of our model are:

* An embedding layer with the length of our dictionary created before as input dimension
* A conv1D layer with ‘relu’ activation and same size padding
* A dropout layer 1
* An LSTM layer
* A dropout layer 2
* A dense layer 1 with ‘relu’ activation
* Finally, a dense layer 2 with ‘softmax’ activation

The model is compiled with 'categorical\_crossentropy' loss and its aim is to get the best accuracy. A graphical representation can be seen in the Appendix as Figure 2.

**1D Convolution**

The 1D Convolution block represents a layer that can be used to detect features in a vector. It can be configured with number of filters, producing as output a new vector with the number of channels as the number of filters. Every value in the tensor is then fed through an activation function to introduce nonlinearity. The default is to move filters of a set width by 1 element at a time, this movement called stride. The bigger the stride, the smaller the output vector will be. This can be used to reduce the number of parameters and memory used but leads to a loss of information.

**LSTM**

An LSTM has a similar control flow as a recurrent neural network. The differences are the operations within the LSTM’s cells. The core concept of LSTM’s is the cell state, and it’s various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain. The gates can learn what information is relevant to keep or forget during training. Gates contain sigmoid activations which squishes values between 0 and 1. There are three different gates that regulate information flow in an LSTM cell. A forget gate, input gate, and output gate. The forget gate decides what information should be thrown away or kept, input gate updates the cell state, output gate decides what the next hidden state should be.

Our initial implementation scored a 55% accuracy which suggested that there is room for improvement. As this implementation used parameters determined randomly our next step was to fine-tune on an array of possible settings.

**Hyper parameter optimization**

Hyper parameter optimization is down by using Hyperband.‎[7] *Hyperband is an optimized version of random search which uses early-stopping to speed up the hyperparameter tuning process. The main idea is to fit a large number of models for a small number of epochs and to only continue training for the models achieving the highest accuracy on the validation set*.‎[8] We added a set of values for each layer mentioned above have in order to achieve a better accuracy:

* Embedding layer: output dimension is varied between 128 and 256 with a 32-step size
* conv1D layer: filters vary between 2 and 16, step size is 2, kernel size starts from 4 and can change up to 16 by steps of 4.
* Dropout layer 1: dropout rate starts from 0.1 and with 0.1 steps it can be up to 0.7.
* LSTM layer: units’ minimum value is 96 which can go up to 128, step size is 16.
* Dropout layer 2: same as dropout layer 1, dropout rate changes between 0.1 and 0.7 with 0.1 step size.
* Dense layer 1: units’ minimum value is 32, maximum value is 96, step size is 32.
* The model is compiled with a varying learning rate which can be 1e-2 or 1e-3.

**Evaluation**

The best model has achieved a precision of 67.13% accuracy testing on X\_test. We also tested with 7 categories, on that we achieved 64,77% accuracy and with 10 classes, which got 63.83%. The best model could identify ‘sports’ and ‘business’ the most accurately, and ‘top stories’ the worst. In regards of ‘top stories’ it was expected that this category will be the least accurate as it had the fewest available examples in the training set. Moreover, our model could not distinguish ‘top stories’ from the rest of the categories, only 3 samples were correctly labeled in this category. Results regarding the other 4 categories are someone surprising, it seems that most of ‘world’ categories were identified correctly, but a considerable number of examples were classified as ‘entertainment’ or ‘business’. This pattern is similar for ‘entertainment’ where the majority of faulty labels are ‘world’. A visual representation of Graphical user interface, application

Description automatically generatedthe results can be seen in Figure 1.

Figure 1. Confusion matrix of results

**Results**

We selected 10 samples from the original dataset to showcase our models’ predications. It can be seen from these sentences that the worst performing category is ‘top stories’, which influences the model’s overall precision greatly. All other categories were classified correctly. These example sentences can be seen in Table 2.

Table 2. Example sentences with predictions

**Future plans and conclusion**

As it might be seen, the test cases we implemented are promising, but there is still work ahead. By implementing additional models and a more sophisticated data pre-processing methodology we could accomplish a more superior result. Based on the papers we revised to complete this assignment, we are confident that with a bit of extra effort mentioned before our results could be much better. Overall, this task supplied us with a great amount of extra knowledge which could be used in the future.

**References**

1. Yang, Zhilin, et al. "Xlnet: Generalized autoregressive pretraining for language understanding." Advances in neural information processing systems 32 (2019). <https://arxiv.org/abs/1906.08237>
2. Chi Sun and Xipeng Qiu and Yige Xu and Xuanjing Huang ,“How to Fine-Tune BERT for Text Classification?, (2020), arXiv: 1905.05583 <https://arxiv.org/abs/1905.05583>

1. <http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html>
2. Xinying Song and Alex Salcianu and Yang Song and Dave Dopson and Denny Zhou, “Fast WordPiece Tokenization”, 2021, arXiv: 2012.15524v3 <https://arxiv.org/abs/2012.15524v3>
3. Zhou, Chunting, et al. "A C-LSTM neural network for text classification." *arXiv preprint arXiv:1511.08630* (2015). <https://arxiv.org/abs/1511.08630>
4. Y. Luan and S. Lin, "Research on Text Classification Based on CNN and LSTM," 2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), 2019, pp. 352-355, doi: 10.1109/ICAICA.2019.8873454. <https://ieeexplore.ieee.org/abstract/document/8873454>
5. Li, Chenbin, Guohua Zhan, and Zhihua Li. "News text classification based on improved Bi-LSTM-CNN." 2018 9th International Conference on Information Technology in Medicine and Education (ITME). IEEE, 2018. <https://ieeexplore.ieee.org/abstract/document/8589431>
6. Sherstinsky, Alex. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network." *Physica D: Nonlinear Phenomena* 404 (2020): 132306. <https://doi.org/10.1016/j.physd.2019.132306>
7. Li, Lisha, et al. "Hyperband: A novel bandit-based approach to hyperparameter optimization." The Journal of Machine Learning Research 18.1 (2017): 6765-6816. <https://jmlr.org/papers/volume18/16-558/16-558.pdf>
8. <https://www.sicara.ai/blog/hyperparameter-tuning-keras-tuner>

**Diagram

Description automatically generatedAppendix**

Figure 2. Network sturcture