Introduction to Python and Natural Language Technologies
BMEVIAUAV35
2020/21/2

Movie genre classification based on plot (Multilabel classification)

Team name:
Whatever, just choose something

Team members:
Bidlek Márton (GPUVN8)
Csabai Máté Mihály (F8SM1M)

1. Description of the task

1.1 Motivation

We wanted to have a task, which is common enough to have appropriate datasets on the internet, because mining data is very draining and resourceful, on top of that neither of us had the tools nor the knowledge to do that. There were project ideas given in a list and links to external data science pages with a vast amount of datasets. In addition to these, the fact that we share a common interest in movies and tv shows helped us arrive at a consensus of doing "determining the genres of movies based on their plot". Other ideas we had also involved movies and tv shows as datasets, however the complexity of some of these made them really unappealing for two begginers like us.

1.2 Description

Our chosen task is determining the genres of movies based on their plot, which is a multilabel classification task. This is one of the most common tasks of NLP, where a text or document is given and we have to categorize it to one or more categories which are given in some form. The datasets were from a data mining site called kaggle.com, and we used two separate models for the task, both of us tried implementing one and we compared results and conclusions afterwards. Csabai Máté implemented the LSTM based model, Bidlek Márton the BERT based one. In short, we made a classifier comparison exercise on the same dataset. Both the LSTM and Bert models were implemented by the NLP lectures, laboratories and open source codes.

2. Results

2.1 Preprocessing of the data

The data preprocessing was the most time-consuming part of the job. We tried multiple datasets from kaggle.com, and we chose the least dirty one. Our dataset had 22579 lines with 2 columns. A text and a genre (label) column. The elements of the sequences were words.

The text column contains the short plot of the films in string format. First of all we deleted all the punctuation and stopwords from the data and applied a lowercase function to these strings. To convert our strings to number format, we used a tokenizer.
The genre types were also strings, so we had to dummify it.

After these procedures, we splitted our data to train, validation, and test datasets.

2.2 Deciding on models

For the purpose of comparing the models, we worked with the same parameters wherever we could, such as epoch number, batch size, etc.
In this documentation, we get these results with the epoch number = 5, batch size = 10, training with 4000 samples, and test with 2000. We chose these numbers because the pretrained BERT model was too resource hungry, and bigger numbers weren't possible on our computers. Even using a GPU, the batch size of 15 caused memory overflow.

2.2.1. LSTM

The type of this model is a Sequential model. In the Sequential model every layer has one input and one output tensor. The first layer of the model is an Embedding layer, which turns indexes into dense vector [1]. The model also contains layers like: SpatialDropout1D with the rate of 0.2, LSTM (Long Short-Term Memory) and Dense with Softmax activation function. The type loss function is CategoricalCrossentropy, which computes the crossentropy loss between the labels and predictions [2]. We also used the Adam optimizer and Early Stopping which monitored the validation loss value. This model was implemented by this documentation [3].

2.2.2. BERT

BERT is a famous NLP architecture which has inspired many recent NLP architectures, training approaches and language models, such as Google's TransformerXL, RoBERTa, etc.[4] We created the neural network BERTClass with transformers.BertModel which is a bidirectional transformer pretrained using a combination of masked language modeling objective and next sentence prediction on a large corpus. The advantage of this model is that it can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks. The loss function we used is a combination of Binary Cross Entropy which is implemented as BCELogits Loss in PyTorch. This model was based on this documentation [5].

2.3. Conclusion

	LSTM		BERT	
Epoch	Accuracy	F-score (Macro)	Accuracy	F-score (Macro)
After Last Epoch	0.5951	0.5840	0.7753	0.4205

After the last (5th) epoch we got these results. The BERT has a better accuracy after a small amount of epochs which can be the direct result of it being pretrained, further tests are required with larger computing units. Accuracy wise, the LSTM was the winner after the 5 epochs.

3. Resources

[1] Embedding layer ,https://keras.io/api/layers/core_layers/embedding/, Last download: 2021.05.09.

[2] CategoricalCrossentropy class ,https://keras.io/api/losses/probabilistic_losses/#categoricalcrossentropy-class, Last download: 2021.05.09.

[3] Multi-Class Text Classification with LSTM, Susan Li, 2019. 04.10. https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17, Last download: 2021.05.09.

[4] Demystifying BERT: A Comprehensive Guide to the Groundbreaking NLP Framework, Mohd Sanad Zaki Rizvi, 2019.09.25. https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/, Last dowload: 2021.05.09.

[5] Fine Tuning Transformer for MultiLabel Text Classification, https://colab.research.google.com/github/abhimishra91/transformers-tutorials/blob/master/transformers_multi_label_classification.ipynb, Last download: 2021.05.09.