Multi-label Text Classification

Martin Čapek

Faculty of Information Technology, Czech Technical University in Prague

December 29, 2020

I. Introduction

In this semestral work I will create neural network for assigning multiple lables for articels. I will show two different text embedding methods.

In Multi-Class classification there are more than two classes and instance can have multiple multiple classes, e.g. article can be labeled with [*USA, Election, Donal Trump*] from tens or hundreds of labels.

Multi-label classification has many real world applications from categorising news articles to emails or assigning multiple genres to a movie.

II. Input data and preprocessing

I will be using data from BBC [1]. It is a huge database of news and the reason I've chosen this, is because the articles are multilabeled. In order to get the content, I've wrote an external program to crawl through the website and download the necessary data for the task.

There were a lot of labels, and some was similar, so I unified them by name. Examples:

- Impeachment of Donald Trump → Donald Trump
- Donald Trump's border wall → Donald Trump
- Gina Miller → Gin
- Ruth Bader Ginsburg → Gin

There are some mistakes, where unification doesn't make sence. We will select significant labels (labels, that have at least 100 articles), and then look, if some mistakes will be there.

We can see significant labels, with appearances in articles in *Table 1*. Only label *Unite* was incorectly unified and will be removed.

We can also see from *Table 1*, that our dataset is imbalanced. We want our prediction to be accurate in all these label, so we will be using *class_weight*[2] in our NN.

US election 2020	100
Social media	102
Social distancing	102
Cardiff	117
Retailing	129
Welsh government	130
India	132
Manchester	139
Self-isolation	140
Personal finance	153
Boris Johnson	170
NHS	228
UK economy	233
Donald Trump	243
Police	247
Brexit	277
China	296
Companies	313
United States	333
Unite	373
Coronavirus	1898

TABLE I
LABELS

[1] https://www.bbc.com/news

[2] https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit#supporting_sample_weight_class_weight

III. Methods

I used two text embeddings, GloVe[1] and BERT[2].

A. GloVe

GloVe is method for mapping words to vectors. I downloaded smallest pre-trained dataset from this site. The dataset comes with different dimensions of vector, I used 100 dimensional one.

I used embedding layer with weight matrix computed from GloVe, see *Fig. 1*. Then I used bidirectional LSTM. I choose bidirectional because we have all the information from the text, so it should be better than normal LSTM.

B. BERT

BERT is transformer-based language model. I decided to use pretrained BERT model [3], because training a model would take a lot of time.

C. Classifier

For both methods I used accuracy as metric, Adam optimizer and binary crossentropy loss function.

At the end of NN I used the sigmoid activation function. This will predict the probability for each class independently and select ones passed threshold.

IV. Results

	train	test	validation
BERT	0.9504	0.9566	0.9543
GloVe	0.9470	0.9373	0.9349

TABLE II
ACCURACY

	train	test	validation
BERT	0.0829	0.2033	0.0967
GloVe	0.0797	0.1744	0.1776

TABLE III
LOSS

BERT has slightly better results than GloVe. However there are both good results given small training data.

V. Conclusion

There are parameters to play with, which can achieve better model's results. In GloVe we could try GRU layer instead of LSTM. In BERT we could try different pretrained model ...

This article classifier uses only 20 labels (see *Table 1*), which is insufficient. For adding new labels we should gather larger dataset.

[3] https://huggingface.co/bert-base-uncased

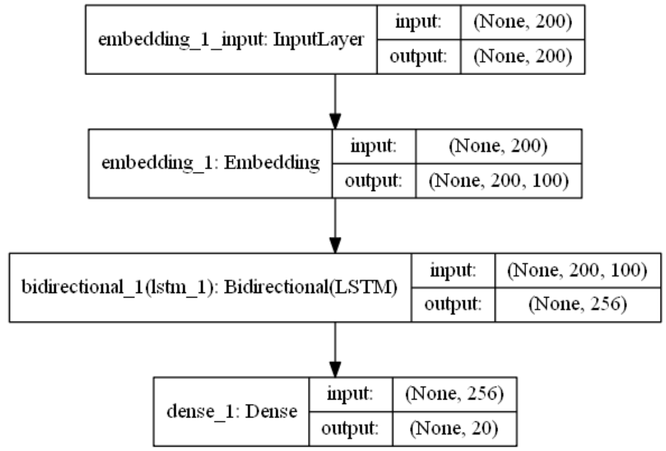

Fig. 1. Architecture for GloVe

References

[1] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].