Assignment 2 Report:News Classifier based on bert model

WANG XITING 18206392, ZHAO MEISHANSHAN 18206067, LIU XINYUN 18206386, and SUN YUXUAN 18206360

1 ABSTRACT

In recent years, the Internet has developed very rapidly and communication methods have emerged one after another, which means that people have more and more channels to get news from all over the world. Under this circumstance, it is important to classify these news properly. This is a crucial field of data mining and natural language processing. Our assignment make some surveys about some models of NLP and choose the best one for our task. Our task includes data pre-process, model construction and prediction. We finally got the ideal outcome.

Additional Key Words and Phrases: Natural language processing, news classification,

ACM Reference Format:
Wang Xiting 18206392, Zhao meishanshan 18206067, Liu Xinyun 18206386, and Sun Yuxuan 18206360. 2018. Assignment 2 Report:News Classifier based on bert model. In . ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

2 INTRODUCTION

There are different news classification models using data mining and machine learning algorithms and these models have their own advantages and disadvantages. After reading some literature and trying some models, it is found that in the field of NLP, compared with other models, BERT model has high efficiency in semantic analysis and its effect is the best. Therefore, it is very suitable to be used in the research of news classification. In this project, the whole news dataset will be divided into 2 parts. After the model structure is imported, part of the news samples (train set) will be trained first to establish the model and the prediction pattern can be verified by testing the samples. After these processes are completed, the other part of the news samples (test set) will be predicted which category they belong to through the established model. The final prediction will also be calculated. Part of our code refers to[5].

3 LITERATURES REVIEW

News classification belongs to text classification, which is a classic problem in NLP. In previous studies, people used various neural models to learn text representation, such as convolution model, recursive model and attention mechanism. By using a large number of unlabeled data, pre trained language models have shown a useful role in learning common language representation. These models include Elmo, GPT, Bert and so on. I will introduce four models for and compare them about the advantages and disadvantages

Manuscript submitted to ACM

3.1 Fasttext

First, I want to introduce Fasttext. It is a convenient tool launched by Facebook, including text classification and word vector training. Fasttext's classification implementation is very simple: convert the input into word vector, take the average, and then get the category through linear classifier. The input word vector can be pre trained or initialized randomly and trained together with the classification task.

However, it has some shortcomings. It has a simple structure and is essentially a linear model. Therefore, it is suitable for some short texts. For some samples with long length or linear inseparable, its effect needs to be improved.

3.2 TextCNN

Another common model is TextCNN. It is a model proposed by Yoon Kim in 2014, which pioneered the use of CNN to encode n-gram features. The convolution in the image is two-dimensional, while textCNN uses "one-dimensional convolution", that is, filter_size * embedding_dim has a dimension equal to embedding. In this way, the filter can be extracted_Information about size gram.

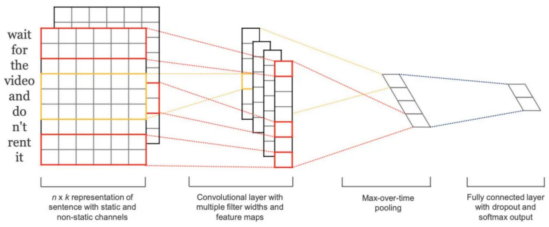

Fig. 1. TextCNN

TextCNN is a strong baseline suitable for medium and short text scenes, but it is not suitable for long text, because the convolution kernel size is usually not set to be large, so it is unable to capture long-distance features. At the same time, Max pooling also has limitations and will lose some useful features. In addition, if you think about it carefully, textcnn is essentially the same as the traditional n-gram word bag model. Most of its good effects come from the introduction of word vector, because it solves the sparsity problem of word bag model[4].

3.3 GPT

GPT is also a common model. It is short for "generative pre training". It is divided into two stages. In the first stage, the language model is used for pre training (unsupervised form), and in the second stage, the downstream tasks are solved through fine tuning mode (supervised mode). Its feature extraction ability is stronger than RNN, so it adopts transformer's feature processor.

GPT uses the transformer's decoder structure and makes some changes to the transformer decoder. The original decoder contains two multi head attention structures, while GPT only retains mask multi head attention. For different downstream tasks, we need to keep up with the network structure of GPT and transform the network structure of tasks into the same as that of GPT. Its advantage is that it can capture more and longer range of information than RNN.

Besides, its computing speed is faster than RNN and easier for parallelization. His disadvantage is that he is a single structure, can not use the context information, can only use the above[3].

4 DATA PRE-PROCESSING

4.1 Set and fix a random seed

If a random seed is given in the model, a random sequence can be generated by a random function and the seed. Once the seed has been determined, each time the random function is used it is equivalent to obtaining a random number from the random sequence, with the random number obtained being different each time.

While randomness allows the same code to produce different results and potentially better results, it can also lead to results that are not reproducible. As we did not have a lot of time and conditions to carry out extensive testing, we decided to fix the random seed, which facilitated the adjustment of the remaining parameters [2].

4.2 Processing forms

train.csv has four columns: **id**, **category**, **title** and **body**. in order to process the data to the extent that the bert model can be used, the data is processed as follows:

(1) Combine the title and body of each line with the separator **'[SEP]'** to form a new element **'text'**
(2) Convert all the content in **'text'** to lowercase.
(3) Divide these contents randomly into a new training set and a testing set.

After processing, the resulting data meets the requirements of the bert model.

4.3 Set text truncation length

The data entered into the model needs to be of the same length. Although shorter data can be filled in with blanks later, filling all text to the maximum will result in a less accurate model after training. Therefore, it is therefore necessary to select an appropriate text truncation length [6].

The length analysis of the processed **train.csv** was followed up with the following data:

```
Out[9]:
count    200000.000000
mean        241.759190
std         466.160979
min           3.000000
25%          28.000000
50%          43.000000
75%         302.000000
max       37465.000000
Name: text, dtype: float64
```

Fig. 2. Data length

It can be seen that each sentence consists of an average of 241 characters, with the shortest sentence being 3 in length and the longest being 37,465 in length, with less than 75% of these data being less than 302 in length.

As most of the data were less than 302 in length, we chose two truncation lengths (300 and 400) and tested them at the beginning. With all other values guaranteed to be the same, it appears from the results that when 300 was chosen over 400, the accuracy increased by 0.00002, and similarly, when the truncation length was set to 200 as a control group, the accuracy decreased by 0.00336 over 300, so the inference is that the optimal truncation length should be between 200 and 400, probably closer to 300. As there was not enough time to find the optimal value, the cut-off length chosen in this model was 300.

5 METHODOLOGY

5.1 Bert Model

Recently, the Bert model newly released by Google AI team has shown amazing results in machine reading test, and the Bert model has made rapid progress in the field of NPL recently. The Bert model has two characteristics:

(1) Its depth is 12 layers, but its width is not very wide. The middle layer is only 1024, compared with 2048 in the previous transformer model. This means that a deep and narrow model is better than a shallow and wide model.
(2) It applies two-way transformer to language model. The language model of two-way training will have a deeper understanding of context than the one-way language model.

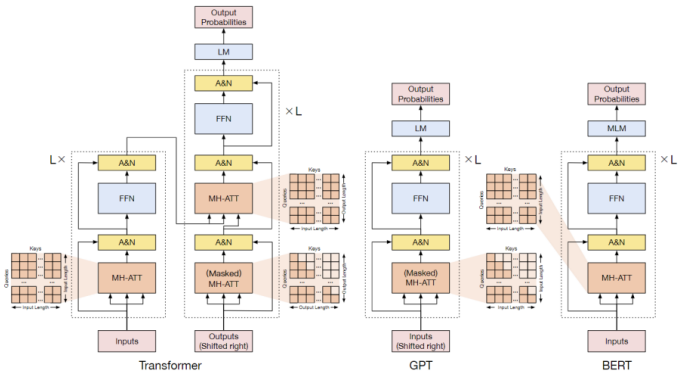

Fig. 3. Recent progress in NLP — Pre-trained LM

This is the picture in our PPT, which also shows that BERT is better than GPT and Transformer. Bert is trained by huge data, huge model and huge computational overhead, and achieves the best results in 11 natural language processing tasks. That's why we choose it in our assignment.[1]

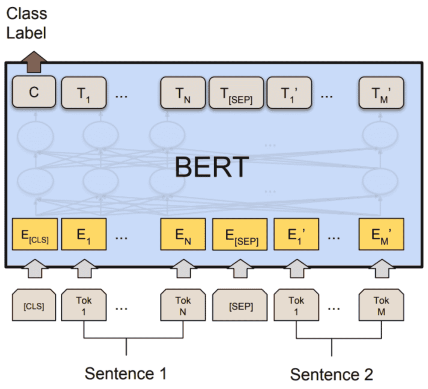

Fig. 4. Bert classification model structure

For text classification, Bert only accepts the input of a sequence of no more than 512 tags. The length of the sequence is within two segments. The first tag is CLS, which contains special classification embedding. The other tag is SEP, which is used to separate segments. Bert takes the final hidden state h of the CLS as the representation of the whole sequence. Add a simple softmax classifier at the top of Bert to predict the probability of tag C.

$$p(c \mid \mathbf{h}) = \text{softmax}(W\mathbf{h}) \tag{1}$$

BERT uses two unsupervised tasks for parameter pre-training. The first is MLM (Masked Language Model), in which Jane randomly scoops out several blanks in a sentence and allows the model to predict the words that were originally in those blanks. The other pre-training task is NSP (Next Sentence Prediction), in which the model is pre-trained to understand the relationship between two sentences.[7]We also need to adapt Bert to specific NLP tasks, which requires appropriate adjustment. There are usually three ways: Fine-Tuning Strategies, Further Pre-training and Multi-task Fine-Tuning[8].

News classification is a type of text classification, and text classification is a very important task in NLP. Our classification task Bert text classification model often treats the first token position (CLS position) of the output of the last layer of bert as a representation of a sentence, followed by a fully connected layer for classification.

5.2 Model Construction

A news classifier is first defined, importing the previously defined **PRE_TRAINED_MODEL_NAME**, which was **'bert-base-uncased'**, with all tokens in lower case. The next init and forward propagation sections initialise the information, providing bert and the fully connected layer used for classification, respectively. Looking at the forward function, the data is first fed into the BertModel, then dropout is performed, followed by a Linear layer used for classification. The **n_class** here is also the number of news categories we have, which can already be derived during data processing, with a total of 32 news categories. This means that the classification task is simply adding a Linear layer to bert's model.The explaination of the basic parameters are:

- **input_ids** is the word index in list corresponding to the word(word2index)
- **attention_mask** illustrate a subword as 1 and a padding as 0

- **pooled_output:** used to train the output of the two-sentence bert model, FloatTensor type, with a shape of $[batch\_size, hidden\_size]$.

Next the optimizer is defined and a warm-up of the learning rate is performed. After the warm-up phase a schedule is created to linearly reduce its learning rate from the initial lr in the optimizer to 0. The three parameters and our settings are:

- **num_warmup_steps:** the number of initial warm-up steps —— parameter is set to 0, the learning rate has no warm-up process, only a gradual decay from the initial set learning rate to 0
- **num_training_steps:** The total number of training steps is $[number\_of\_batches]x[number\_of\_epochs]$

5.3 Dataset Testing Category Prediction

The dataset to be tested will be imported into the model for prediction. Among them, the variable called item is set as the data index. **Tokenizer** was used to segment the sentence, so as to obtain the word vector and encode the sentence. Then, the self-attention operation is carried out on these divided words to find out the key points, and these results will be returned to the arrays. Then, the data will be trained in batches by using iterators, which means **batch_size** samples will be trained and returned.

The information obtained in the process above is written into the model and loaded on the specified device. Then, the category of data is determined by returning the maximum value and the index of the maximum value, and the **softmax** operation will be performed on all elements in outputs with different first-dimensional subscripts and the same other dimensional subscripts to calculate a probability for the results of each classification, then these will be spliced. The results are written into an array named category, and the data number (id) and final result (category) are imported into the csv file.

Finally, because the whole training process might take a long time, it is a better choice to use **tqdm** to monitor the progress of this process.

6 EVALUATION

In addition to our final bert model, we also tested several other models with the following accuracies:

Model	Accuracy
TextCNN	0.58
LSTM Model	0.63
Glove TextCNN	0.75146

Since none of them exceeded the bert model in terms of accuracy, we chose to use the bert model and optimise it.

Although not fixing the radom seed may give better results, we hope that similar distractions can be excluded as far as possible

The final bert model we submitted was 2 epochs with an limit the length of each text as 300.

REFERENCES

[1] 2018. [NLP Natural Language Processing] An in-depth analysis of Google's BERT model. https://blog.csdn.net/qq_39521554/article/details/83062188
[2] 2020. A brief description of random seeds for python pytorch. https://www.bilibili.com/read/cv8427731
[3] 2020. [NLP-14] GPT Model (Generative Pre-Training). https://www.cnblogs.com/yifanrensheng/p/13167796.html
[4] 2021. [NLP] Common Models for NLP Text Classification with Deep Learning. https://blog.csdn.net/weixin_43935696/article/details/113854130
[5] 2021. [NLP] Xunfei English Academic Paper Classification Challenge Top 10 Open Source Multi-Solution - 5 Bert Solution. https://blog.csdn.net/weixin_43935696/article/details/119666802
[6] 2021. Tenchi Zero Beginner NLP Competition Practice: Task1 Task2 Data Reading and Data Analysis. https://blog.csdn.net/weixin_43243315/article/details/120802406
[7] Ming-Wei Chang Jacob Devlin and Kristina Toutanova Kenton Lee. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018).
[8] Yige Xu Xipeng Qiu and Xuanjing Huang. 2019. How to Fine-Tune BERT for Text Classification? (13 10 2019).

,