Propaganda Detection

Candidate number: 246743

April. 2023

Abstract

Text classification has been a popular topic and different models have been applied to different text classification scenarios. In this experiment, the Convolutional Neural Network (CNN) and Bidirectional Encoder Representations from Transformers (BERT) are used to detect different texts that may contain propaganda. The goal is to determine whether a given utterance contains propaganda and to classify the types of it. This experiment uses the given dataset and validation set for training and evaluation of the model. Datssets contain 8 types of propaganda techniques and 1 label - "not_propaganda". Overall, BERT outperforms CNN - BERT achieves an accuracy of around 76% compared to CNN's 55%. More detailed metrics (precision, recall, f1) will also be mentioned and analysed within this paper.

Introduction

With the development of technology, more and more information is gradually flooding the social media and being read by people. However, more information sometimes does not mean more sources of knowledge. On the contrary, propaganda can have a potential impact on people's daily lives and on society, due to the diversity of sources and the lack of a consistent and precise review mechanism. This is why the topic of classifying propaganda has arisen. In this context, various language models have been used to train and perform classification tasks, such as the n-gram language model, word2vec, Convolutional Neural Network and Bidirectional Encoder Representations from Transformers. Each of these models has different characteristics and expertise for classification or recognition tasks.

For example, the n-gram model uses great likelihood estimation and the parameters are easy to train. However, due to the sparsity of the data, it is difficult to avoid Out of Vocabulary and performs poorly on long passages.

CNN is a learning model for images. It is also stress-free for high-dimensional data processing and can perform multi-classification tasks. However, CNN is strong in local feature collection ability, so it will be less accurate in understanding long passages of context.

BERT is capable of acquiring contextually relevant bi-directional feature representations. However, since only 15% of the tokens are predicted in each batch of training data, the convergence speed of BERT is slow.

Background Information

- CNN

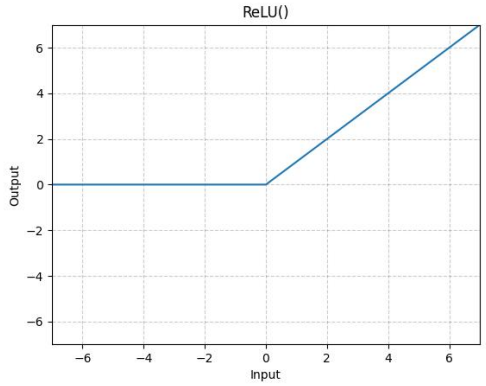
A classical CNN network structure typically consists of an input layer, a convolutional layer, a pooling layer and an output layer (fully connected layer + softmax layer). (Z Li, 2022)

Generally the raw input data needs to be pre-processed before it is brought into the input layer to conform to the training of the model.

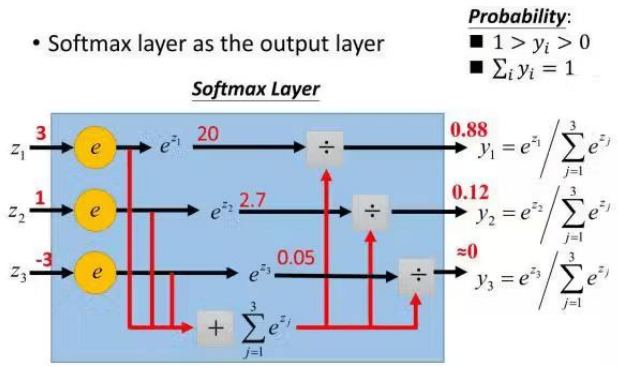
After this, the main function of the convolutional layer is to compress and remove redundant information. A matrix is used as the convolution kernel, and the operation performed here is basically to make the kernel work with a larger matrix of input data to achieve compression.

Since the default input and output are linear functions, an activation function needs to be introduced to bring a non-linear element to the model, allowing the neural network to approximate any non-linear function, so that the neural network can be applied to a wide range of non-linear models.

Commonly used activation functions are ReLU combined with Softmax.



The ReLU function is mostly used for the output of hidden layer neurons. When the input is less than 0, the output is all 0. When the input is greater than 0, the output is equal to the input.



Softmax is generally used for the output of multi-classification neural networks. In simple terms, it serves to map all input values to between 0 and 1.

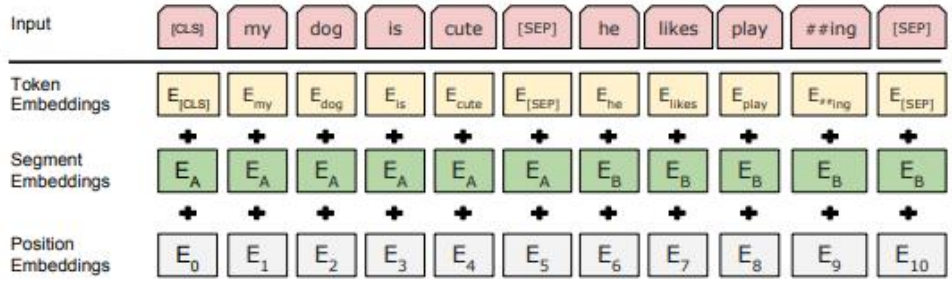
Generally, the convolutional layer is followed by a pooling layer. The pooling layer is mainly used to sample the features of the convolutional layer. Generally pooling is divided into average pooling and maximum pooling. As the name implies, average pooling is to sample the average value of each pool, while maximum pooling is to sample the maximum value in each pool.

A fully connected layer will contain an input layer, a hidden layer and an output layer, which will take the previously passed one-dimensional vectors and transform them into a smaller number of output layers for classification.

- BERT

BERT uses multiple encoders stacked on top of each other. For example, BERT-base uses 12 layers of encoders and BERT-large uses 24 layers of encoders, which have the same structure but are fed with different parameters.

Input can be divided into token embedding, segment embedding and position embedding.(Devlin J, 2018)



In the input column, the CLS takes on a binary classifier to perform the binary classification task.
The CLS does not represent the semantic information of the whole sentence; the SEP is the sentence segmentation character.

Token embeddings pass all words in the input as embeddings. Segment embeddings are for utterance differentiation. The first sentence is represented by EA (0) and the second sentence by EB (1). Position embeddings are positions that the model learns on its own, based on random initialization.

For the model, BERT chooses the MLM (masked language model) with masked training.
Its objective function is auto-embedding, which predicts the reconstruction of the original data from corrupted input data and can use contextual information.

Code Explanation

- CNN

In this experiment, pre-processing was required as the data source was stored in a given tsv file. After extracting these training and test data and load them into variables, word tokenization is necessary in order to separate each word for the subsequent analysis task. In the experiments, a tokenizer by word frequency was created and word frequency numbers were used instead of the original words. Then, padding was used to standardise the specification.

Next, a simple linearly structured CNN model is created. In the Embedding layer, the size of tokenizer is passed in as the input dimension and 100 is chosen as the output dimension. a padding length of the data is also passed in. The one-dimensional convolutional layer is tied to the pooling layer because this reduces the spatial size of the data and thus the number of parameters in the neural network, thereby reducing computational resource consumption and suppressing overfitting. This pairing is added twice in sequence, where the filter and kernel size of the convolutional layers are both decreasing in order to conform to the convergence pattern of the model. A DropOut layer with rate=0.5 is used to suppress overfitting as much as possible, and finally a fully connected layer with an activation function of softmax is used, with units=9 in the parameters corresponding to 9 propaganda labels.

Adam is used as an optimizer because of its strong convergence to improve the performance of the algorithm. Sparse categorical crossentropy was chosen as the loss function to fit the previous softmax activation function.

After this, batch training was performed and iterated continuously to update the weights. After 30 epochs of training, the validation set was used to evaluate the model. Here, the accuracy is used as an indicator to demonstrate the performance of the model, and the loss trends of both the training and validation sets are recorded to see how well the model over-fits. Confusion matrix and classification report are also printed out for analysis.

- BERT

Likewise, the data stored in the tsv file is first extracted into variables, and then both the train and validation data are separated into context and labels. in the preprocessing, all labels are first replaced with numeric designations from 0-8.

BERT-base is used in Tokenizer for pre-training, and BertForSequenceClassification is used to add a linear layer + activation function for classification. For the training and validation sets, I batch encoded their tokenizers. Max length was set to 256 and the input was automatically padded via pad_to_max_length. Special token and attention mask were both added to aid model understanding.

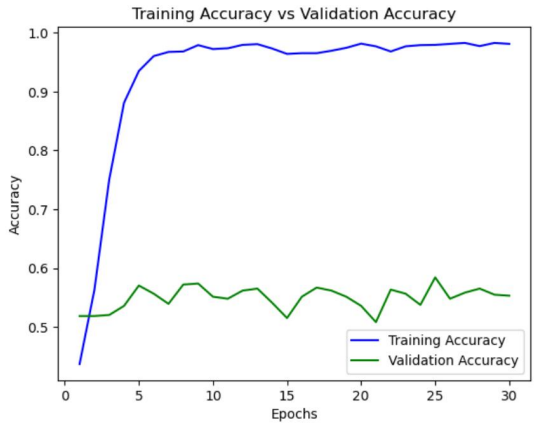
For each epoch of training, the optimiser first sets the gradient to zero and then, after importing the input ids, attention masks and labels into the model, calculates the model's loss value. Based on this loss value, the model performs back propagation and feeds back the optimal method to update the weights to minimise the loss function.

Each epoch of training is followed by a model evaluation, where the model is evaluated against the validation set and the results are printed out. The training for this experiment has a total of 10 epochs.

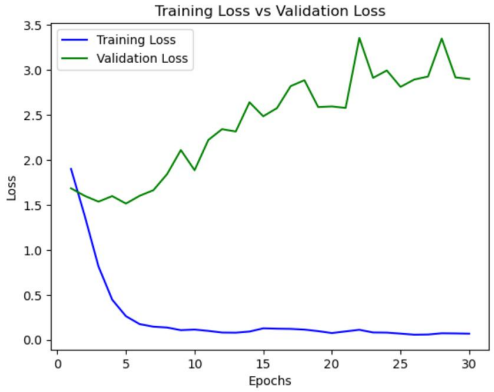
Again, a line graph of the variation of accuracy with epoch number will be visually plotted and the classification report will be presented to analyse the evaluation indices besides accuracy.

Result

- CNN



For this experimental model, the accuracy obtained after 30 epochs was around 55%, while the highest accuracy during the training period was over 58%. The average accuracy of this CNN model is between 54% and 55% over several runs. In the output graph, the training accuracy value goes up, leveling off at the 6th epoch and remaining above 95% thereafter. This indicates that the model was successfully trained. For the validation set, the accuracy only increases at a lower gradient, it essentially increases from 51% to about 58%. It is worth noting that the validation accuracy, after rising from 0.51 to 0.57, fluctuates considerably, leaving the accuracy hovering between 0.50 and 0.56.



Losses on the training and validation sets were also plotted. The training set's loss is decreasing over time, reaching a very low value at around epoch 5 and remaining stable from then on. The validation set's loss also decreases from epoch 1 to epoch 5, but continues to grow thereafter.

```
0 :  flag_waving
1 :  appeal_to_fear_prejudice
2 :  causal_oversimplification
3 :  doubt
4 :  exaggeration,minimisation
5 :  loaded_language
6 :  name_calling,labeling
7 :  repetition
8 :  not_propaganda
```

	0	1	2	3	4	5	6	7	8
0	12	2	8	3	3	2	3	2	4
1	0	8	6	9	1	3	3	4	9
2	1	2	11	7	1	0	3	0	6
3	0	1	6	13	5	4	2	0	7
4	0	0	2	1	6	1	3	2	13
5	1	2	3	1	2	7	2	3	16
6	0	1	0	6	2	4	9	0	9
7	1	1	2	0	1	3	3	11	10
8	3	5	4	5	6	18	11	5	244

The printout of the confusion matrix shows that the distribution of the predictions on the confusion matrix is sporadic due to the accuracy, the only accurate label is "not_propaganda", the rest of the results are scattered too much.

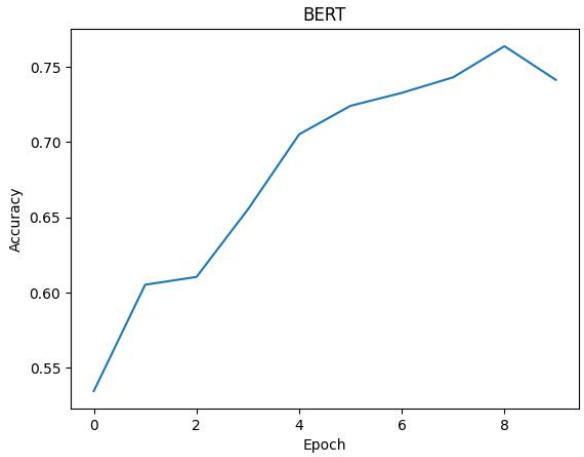
	precision	recall	f1-score	support
flag_waving	0.67	0.31	0.42	39
appeal_to_fear_prejudice	0.36	0.19	0.25	43
causal_oversimplification	0.26	0.35	0.30	31
doubt	0.29	0.34	0.31	38
exaggeration,minimisation	0.22	0.21	0.22	28
loaded_language	0.17	0.19	0.18	37
name_calling,labeling	0.23	0.29	0.26	31
repetition	0.41	0.34	0.37	32
not_propaganda	0.77	0.81	0.79	301
accuracy			0.55	580
macro avg	0.38	0.34	0.34	580
weighted avg	0.56	0.55	0.55	580

However, accuracy is not very useful for unbalanced data sets to obtain accurate predictive power for each label. Therefore the scores for precision / recall / f1 should be taken into account. The values in the "not_propaganda" column of the Classification report table (precision, recall, f1-score) are all the highest, all above 0.77. This indicates that the classification of "not_propaganda" is relatively robust and accurate overall, but as the absolute value does not exceed 0.90, the ability of this classification can be further improved by adjusting the hyperparameters.

In addition, the precision of "flag_waving" is 0.67, but its recall and f1-score are both very low, not exceeding 0.50, which indicates that the model cannot detect this class well, but when this label is detected, its judgement is relatively convincing. Again, the model needs to be further improved as the scores are not high in absolute terms.

For the other labels, all three metrics received very low scores. This indicates that the model is very poor at classifying this category. This may be related to the size of the training set, or the complexity of the model.

- BERT



The experiment was conducted over a total of 10 epochs, with a gradual increase in accuracy. As can be seen, the accuracy increases from 0.53 at epoch 1 to 0.76 at epoch 9, and then drops slightly to 0.74 at epoch 10.

	precision	recall	f1-score	support
flag_waving	0.69	0.64	0.67	39
appeal_to_fear_prejudice	0.65	0.70	0.67	43
causal_oversimplification	0.39	0.61	0.47	31
doubt	0.53	0.68	0.60	38
exaggeration,minimisation	0.56	0.54	0.55	28
loaded_language	0.59	0.51	0.55	37
name_calling,labeling	0.54	0.71	0.61	31
repetition	0.33	0.31	0.32	32
not_propaganda	0.98	0.88	0.92	301
accuracy			0.74	580
macro avg	0.58	0.62	0.60	580
weighted avg	0.77	0.74	0.75	580

But accuracy is not the best indicator of a model, nor is it the only indicator. So in the classification report precision, recall and f1-score are introduced. The table shows that the precision and f1-score for "not propaganda" are quite high, coming in at 0.98 and 0.92. Its recall is a little less impressive (0.88), but still an excellent score compared to other techniques. Overall, the model is more accurate and robust for the identification of "not_propaganda"

Slightly lower than the score for "not_propaganda" are "flag_waving" and "appeal_to_fear_prejudice ", all three indicators are relatively average. This indicates that the model is capable of classifying these two techniques, but there is still much space for improvement. Adjustments can be made by fine-tuning the hyperparameters.

In addition, there are some labels such as "name_calling, labelling" and "casual_oversimplification". Their recall is relatively outstanding, scoring 10 to 20 percentage points higher than both precision and f1. This shows that the model is able to detect the category well, but because it is so broad, the results judged to be true also include other categories, resulting in a decrease in overall precision. This could be improved by further clarification of the category.

Other technical labels, such as "competition", have very low scores for all three metrics, basically not exceeding 0.35, which indicates that the model is poor at detecting this label and is unable to detect it well. This may be due to the inadequacy of the training set, so that further experiments can be fine-tuned after the training set has

been expanded.

Discussion and Future Work

In fact, during model training, the evaluation of the completed model is a result of several factors. Firstly, the size of the dataset can have a significant impact on the final trained result. This is why tokenization, normalization and a number of other pre process steps need to be done. Secondly, when creating the model, each layer that is added and the hyperparameters that are passed in at that layer will have an impact on the final results of the model. For example, after epoch 20 of the model training, there are two large cliff drops, which may be due to instability caused by the gradual increase in the loss values. This may represent a gradual tendency of model overfitting. This can also be supported by the trend of validation loss of decreasing and then increasing , which is often caused by insufficient training data. Besides to have more training text, it can also be mitigated by using more DropOut, or reducing the number of epochs.

In addition, CNN may not perform well on text classification tasks due to their feature of focusing on local fields. The lack of long-range text understanding makes it difficult for the model to recognise word-phrase associations within context. Further fine-tuning of the hyperparameters is necessary, and using them together with other methods to build a joint model is also worthwhile to improve accuracy.

However, accuracy is not very useful for unbalanced data sets to obtain accurate predictive power for each label. Therefore the scores for precision / recall / f1 should be taken into account. The values in the "not_propaganda" column of the Classification Report table (precision, recall, f1-score) are all the highest, all above 0.77. The precision of "flag_waving" is also 0.67, but its recall and f1-score are very low, which indicates that the classification of this label is not stable enough. For the other labels, all three metrics received very low scores, which could perhaps be improved by redesigning the model or adjusting hyperparameters such as the learning rate.

The results of the BERT model are generally better than those of the CNN, with BERT reaching a maximum of 0.78 in terms of accuracy, while the CNN fluctuates between 0.50 and 0.55. This is because BERT's pre-trained model is combined with the downstream task model and the auto-embedding algorithm used is naturally suited to the text classification scenario, so it can be seen that its accuracy goes all the way up. Due to the limitations of this experiment, no more epochs were run, so although there is a slight decrease in accuracy of 2 percentage points at epoch 10 compared to epoch 9, it is still impossible to tell if 0.76 is the highest accuracy for this model.

Furthermore, the commonality between the CNN and BERT models in this experiment can be found in the classification report. They both have relatively outstanding performance in "not_propaganda". This is probably because the features of "not_propaganda" are more easily recognized by both models. In the "flag_waving" column, although both have very high precision, but for f1 and recall, BERT tends to be stable while CNN scores low. This indicates that BERT is relatively stable in classifying this label, while CNN does not find the category well. Finally, 'competition' was found to have the lowest overall score in both models, hovering between 0.3 and 0.4. This may be due to the fact that the training set for repetition is not large enough, or that the features in this category are difficult to mine and identify.

Conclusion

Overall, both CNN and BERT are unique in their approach to text classification, with CNN having a good extraction on local features and BERT having a contextual representation to capture long-distance utterance relationships. In terms of results, BERT is better than CNN, which is actually to be expected. In terms of model construction, CNN contains operations such as Pooling and Dropout to suppress overfitting, so some of the feature vectors are discarded, which leads to a reduction in accuracy. Even so, overfitting still exists, most likely because the size of the dataset is not large enough. In addition, the ability of BERT to capture context and the ability of CNN to capture localization can actually complement each other well (Zhu 2021), so a joint model combining the two is a worthwhile approach.

Reference

Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

Kesav, & Jibukumar, M. G. (2022). Multi-Channel CNN based image classification using SKIP connection and MSVM. International Journal of Computers & Applications, 44(10), 981–990. https://doi.org/10.1080/1206212X.2022.2047443

Kim, & Kang, P. (2022). Cross-modal distillation with audio–text fusion for fine-grained emotion classification using BERT and Wav2vec 2.0. Neurocomputing (Amsterdam), 506, 168–183. https://doi.org/10.1016/j.neucom.2022.07.035

Song, Liu, J., Qian, B., Sun, M., Yang, K., Sun, M., & Abbas, S. (2018). A Deep Multi-Modal CNN for Multi-Instance Multi-Label Image Classification. IEEE Transactions on Image Processing, 27(12), 6025–6038. https://doi.org/10.1109/TIP.2018.2864920

Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," in IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 12, pp. 6999-7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[YNU-HPCC at SemEval-2021 Task 6: Combining ALBERT and Text-CNN for Persuasion Detection in Texts and Images](https://aclanthology.org/2021.semeval-1.144) (Zhu et al., SemEval 2021)