minBERT and Downstream Tasks

Stanford CS224N Default Project

Halil Ibrahim Gulluk
Department of Electrical Engineering
Stanford University
gulluk@stanford.edu

Abstract

In this project we attempt to implement important components of the BERT model. We have implemented embedding layers, attention mechanism and several multi-layer perceptron parts of the model. We tested our model based on 3 different NLP tasks that are Sentiment Analysis, Paraphrase Detection and finding Semantic Textual Similarity. After implementing the model we made further improvements including different loss function trials, taking advantage of the contrastive learning and co-training on different datasets. We observe that these improvements indeed enhanced the overall performance.

1 Introduction

In the last decade the large language models have gained a drastic popularity in the artificial intelligence world (Thoppilan et al., 2022; Devlin et al., 2018; Brown et al., 2020; Raffel et al., 2020; Beltagy et al., 2020). One of the main reasons why the language models had a significant improvement is the invention of the transformer models based on the attention mechanism (Vaswani et al., 2017).

The BERT model (Devlin et al., 2018) uses transformers to generate contextual word representations. It can be used for numerous natural language processing tasks including question answering, translation, sentiment analysis, paraphrase detection and semantic textual similarity. In other words, it can be used for all tasks including text classification, and finding the similarities between text pairs.

We will use our model to perform sentiment analysis, paraphrase detection and prediction of semantic textual similarity. Sentiment analysis is predicting the polarity of a given text. Namely, it is nothing prediction of whether the sentence is positive or negative. Paraphrase detection is predicting whether given two sentences are paraphrase of each other or not. Lastly, semantic textual similarity is a measurement for the contextual similarities between two sentences. Note that semantic textual similarity between two sentences may not be binary. In other words, STS is the degree of similarity.

2 Related Work

For sentiment analysis of drug reviews, Min (2019) proposed a model which combination of a pretrained language model, CNN and Bi-LSTM layers. Similarly, (Rehman et al., 2019) proposed a model using word2vec embeddings, CNN and LSTM layers for sentiment analysis in IMDB reviews. Bi-LSTM and a self-attention approach are combined in (Li et al., 2020) for sentiment classification. (Kokab et al., 2022) proposed another BERT-based model callaed CBRNN for sentiment classification and they show model results on IMDB, Airline reviews and election reviews datasets.

Similar to sentiment analysis, there have been ongoing research about paraphrase detection using transformer models. Among them (Wahle et al., 2021) suggested BERT, RoBERTa and Longformer based models for paraphrase detection. Another fine-tuning of BERT model for paraphrase classification is suggested by (Arase and Tsujii, 2019). (Fenogenova, 2021) uses T5 models Raffel et al. (2020) for paraphrase detection in Russian language.

Additionally, for semantic textual similarity tasks (Ni et al., 2021) proposed Sentence-T5 model. Moreover, semantic similarity in medical domain is investigated by (Ormerod et al., 2021; Sun and Li, 2021; Mahajan et al., 2020; Yang et al., 2020). Another model enhancement for sts tasks proposed by (Gao et al., 2021) where they use contrastive learning based loss function, which enables them to find the similarities better. We will also investigate this paper more in the next sections.

3 Approach

3.1 Details of BERT Model

In this section we will give a brief overview about the BERT Model and its key components. The BERT Model consists of 5 main components: Tokanizer, embedding Layer, transformer layers, feed-forward layers and finally the output layer. Tokanizer takes all sentences and split it to words and then word pieces. To have an equal size (512) for each input sentence it is padding the end of the input sentence with the pad tokens. Moreover, for tasks including two sentences it uses a specific separation token. Finally, it converts tokens to ids and give the ids to the embedding layer. Embedding layer of the BERT model takes ids of the tokens as inputs. It sums positional embeddings, segmentation embeddings and token embeddings. Transformer layer takes output of the embedding layer and apply multi-head attention layers, residual connections, and feed forward layers as it can be seen at Fig. 3

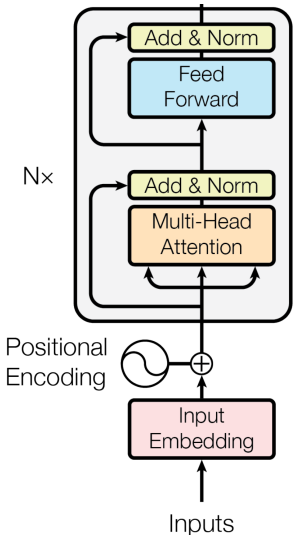

Figure 1: The Transformer architecture

The attention is calculted by the following equation where K,Q,V,d_k stand for keys, queries, values and dimensionality of key respectively.

$$Attention(Q, K, V) = Softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{1}$$

We leave the details of the attention mechanism and transformer layer to the reader which can be found in (Devlin et al., 2018) and (Vaswani et al., 2017).

The output layer consists of last hidden state and a pooler output. Last hidden state is the final word embeddings whereas the pooler output is nothing but the CLS token embedding, that can be used for several downstream tasks.

In our project, initially we implemented the key components of the BERT model. And we test its performance for the tasks described above before further improvements. Details can be found in section 4.

3.2 Further Improvements

We first provide the details of our baseline model, and on top of it we will explain more about other modifications to the model, loss function and training process.

3.3 Baseline Model

In our baseline model for the sentiment analysis we first just take the pooler output of the BERT model and feed this output to a linear layer to get the final prediction. This is also fed into a sigmoid function and binary cross entropy loss, because sentiment analysis is nothing but a binary classification.

When it comes to the semantic textual similarity and paraphrase prediction, we define similarity prediction. This can be used for paraphrase detection by outputting the similarity between the given sentences. In our baseline model, we do this by concatenating outputs of the two sentences and feed it into a final linear layer to get the similarity.

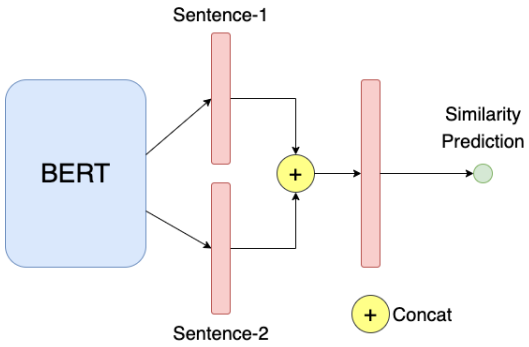

Figure 2: Baseline model similarity prediction

In the baseline training we use binary cross entropy loss for sentiment analysis and paraphrase detection. Note that the situation is different for the semantic textual similarity case. This similarity can take values in $0, 1, 2, 3, 4, 5$. However, we convert it to a binary case by changing labels 0,1,2 to 0 and labels 3,4,5 to 1. Namely, we threshold the labels and convert it to 0-1 values. As a consequence we can use binary cross entropy loss. To have a prediction between 0-1, we divide our predictions by 5 for the sake of loss function calculation.

$$\mathcal{L} = \mathcal{L}_{SA} + \mathcal{L}_{PD} + \mathcal{L}_{STS} \tag{2}$$

In the training process in each epoch we first go over the dataset for the task sentiment analysis, then go one loop over paraphrase detection dataset and similarly go over sts dataset. The total loss for one epoch is shown in Eq. 2. However note that for each batch we are taking gradient with respect to one of the 3 losses not all of them at time same time.

3.4 Dot Product Similarity

As a further improvement we measure similarities between sentences we use the dot product. As the dimension of the output of the BERT model is 768, dot product results might be very noise. That is why we reduce the dimension to r, which is smaller then take the dot product of the sentence representations to get the similarities. Other than that everything stays the same.

3.5 InfoNCE Loss

In order to improve the similarity results we can use the InfoNCE Loss which can enable model to learn dissimilarities as well (Oord et al., 2018). Suppose that our batch size is N and we have pair in semantic textual similarity, that are similar to each other s, s^+.We assume that in that batch all other sentences are not similar to the sentence s, that can be notated as $s_1, s_2, ...s_{N-2}$ Also suppose

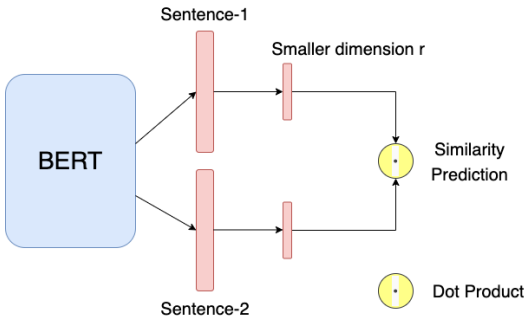

Figure 3: Dot product similarity prediction

that sentence outputs from the BERT model an a linear layer are $\mathbf{h}, \mathbf{h}^+, \mathbf{h}_1, \mathbf{h}_2, ...\mathbf{h}_{N-2}$ then the InfoNCE Loss for the sentence s is given in Eq. 3

$$\mathcal{L}_{InfoNCE}(\mathbf{h}) = -\log \frac{e^{sim(\mathbf{h},\mathbf{h}^+)/\tau}}{\sum_{j=1}^{N-2} e^{sim(\mathbf{h},\mathbf{h}_j)/\tau}} \tag{3}$$

where the $sim(\cdot,\cdot)$ is a similarity metric, we will use cosine similarity in the experiments and τ is the temperature parameter which is chosen to be 0.007, the same with the original paper.

3.6 MSE Loss for STS

As we know that in the original case, semantic textual similarities between sentence pairs can take values from a broad range rather than just binay labels. Motivated by this, we also run models where we do not threshold the semantic textual similarity values and just use mse loss for the STS prediction and all other things stayed the same with the baseline.

3.7 DPS with MSE

In that approach we not only use the DPS method explained in section 3.4 but also use a relu activation on top the similarity result. So the final similarity is between 0 and 1. We multiply it with 5 to get the final prediction. As the loss function for semantic textual similarity we use mean square error.

3.8 Ensemble Model

In order to get a better results in the test leaderboard, we combine the results of the models we have. This is one the best method to get a model with a higher generalization performance in deep learning. We will give the details of which models we combine for the test leaerboard submission in section 4.

4 Experiments

4.1 Data

We will use 3 different datasets for training and testing. For the sentiment analysis we will be using the Stanford Sentiment Treebank (Socher et al., 2013) which consists of 11,855 sentences from movie reviews. For each sample we have one of the following labels: negative, somewhat negative, neutral, somewhat positive, or positive. Additionally, we use CFIMDB dataset consists of 2,434 movie reviews with labels positive or negative.

For paraphrase detection we will be using the Quora dataset [1] which consists of 400,000 sentence pairs with labels are indicating that they are paraphrase of each other or not.

[1] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

For the semantic textual similarity SemEval STS dataset (Agirre et al., 2013)will be used. In this dataset we have 8,628 different sentence pairs with similarities on a scale from 0 (unrelated) to 5 (equivalent meaning).

4.2 Evaluation method

We are just using the classical accuracy metric. For the sentiments analysis and paraphrase detection it is just the accuracy of binary classification. For the semantic textual similarity we will investigate the pearson product-moment correlation coefficient between the output and ground truth labels.

4.3 Experimental details

For finetuning processes we used the same learning rate 1e-5 across all other methods. But for pretrain cases we used 1e-3 as the learning rate. In all training phases we run the models for 5 epochs, it is mainly because we did not realize any drastic improvement after epoch 5. Instead of training same models for long time periods we tried different strategies, models or loss functions in the remaining time.

Training times per epoch takes approximately 20 minutes for multi-tasks, as in the training we do not use all the paraphrase detection dataset but use one-fourth of it, which is randomly chosen in each epoch in order to gain from training times. As we are training our models for 5 epoch it approximately takes 100 minutes to train a model for multi-tasks.

4.4 Results and Analysis

We have stated in the project milestone that we got 0.391 and 0.535 sentiment analysis accuracies for SST dataset, as well as 0.718 and 0.963 accuracies for CFIMDB dataset in the pretrain and finetune phases respectively.

In the first part of the project, we just implemented the sentiment analysis training case and test it on the 3 tasks. Our results for single task training and multi task testing is: **Overall Score: 0.330, Sentiment Classification Accuracy: 0.529, Paraphrase Classification Accuracy: 0.395, Semantic Textual Similarity Accuracy: 0.067** for the dev set.

We implemented and tested our models described in section 3 in the second part of the project (multi-task part). In the table **??** BM, DPS, InfoNCE, MSE, DPS-MSE, EM stand for Baseline model, Dot Product Similarity, InfoNCE Loss, MSE Loss, DPS with MSE and Ensemble Model approaches elaboreted in 3 respectively. Provided accuracies and correlations are the highes numbers in the training process. That is while trainingafter each epoch we have a look at the dev dataset result and among all epoch results we take the the one with the highest sentiment analysis accuracy.

Table 1: Dev Dataset Results

ALGORITHM	SENTIMENT ACCURACY	PARAPHRASE ACCURACY	STS CORRELATION
BM	0.502	0.735	0.250
DPS	0.490	0.498	0.196
INFONCE	0.347	0.521	0.062
MSE	0.462	0.743	0.224
DPS-MSE	0.496	0.775	0.633
EM	**0.508**	**0.783**	**0.657**

Our final results for the test leaderboard is: **Overall Score: 0.639, Sentiment Classification Accuracy: 0.509, Paraphrase Classification Accuracy: 0.788, Semantic Textual Similarity Accuracy: 0.620**

We submit our EM (Ensemble Model) to the test leaderboard due to the its performance in the dev dataset.

We note that the results for the sentiment classification accuracies and paaraphrase classification accuracies are more or less the same across the algorithms. This is expected in the sense that across the algorithms we did not change the parts for sentiment classification and paraphrase classification.

We just use a simple linear layers for those tasks. The reason why we do not change is that they performed comparably well in the leaderboard.

However, main difference comes from the differences of algorithms for the semantic textual similarity. After getting the representations from the BERT model, concatenation and feed into a linear layer does not work for semantic textual similarity. Namely, linear layer can not predict the similarities well, this is the case for BM.

Similarly, just using dot product of the representation is not also a good way for predicting the similarities. Dot product and using binary cross entropy loss did not work well together which is the DPS case.

As InfoNCE loss is used in many contrastive learning settings, we had a motivation that InfoNCE loss can enable model to learn similarities and dissimilarities. But it did not worked in the practice. There might be two reasons for that. First, temperature parameter choice can be very important and we have not chosen the right one. Second, almost all pairs the similarity can be very small, so that the numerator of the InfoNCE loss which enables us to learn the dissimilarities may not contribute that much. Because almost all pairs are already are dissimilar, there is nothing to differentiate.

As we can see DSP-MSE model is indeed a good improvement to the baseline. We train our DSP-MSE models from scratch two times and take 5 model weights from these trainings. Finally, we ensembled those models and got the Ensemble Model (EM). Ensemble models output is equal to the average of the these 5 models. As we can see there is also a slight improvement thanks to the ensembling.

In the overall picture, for sentiment analysis and paraphrase detection cases we got results that are expected. However, for the semantic textual similarity part, we would expect using just one linear layer to have a better performance. But the poor performance can be also explained by the thresholding we use. Because we lose some information there, which is not used in DPS-MSE case.

We again see that, instead of thresholding and using binary cross entropy loss, using cosine similarity with mean squared error gives better performance. We conclude that, for hierarchical classification mse loss performs well and cosine similarity is a better way of calculating similarities.

5 Conclusion

Summarize the main findings of your project, and what you have learnt. Highlight your achievements, and note the primary limitations of your work. If you like, you can describe avenues for future work.

Thanks to this project I learned the BERT model and the usage of tokanizers, embedding layers in a large language model. Converting text data into vectors and using positional embeddings are some of the essential comonents in language model that I learnt in this project. Also, we note that, BERT model is a good feature extractor and for the similarity measurements using cosine similarity can work bettern than using linear layers.

Our main limitation was the time. As a person who is doing the project himself after implementing baselines and a couple of algorithms I could not try the algorithms that I had in my mind due to the limited amount of time.

As a feature work, we can first try InfoNCE loss with different settings like changing tempreature parameter or finding the negative pairs wisely. Because InfoNCE loss is known to be one of the best in terms of calculating similarities. As an example, CLIP model (Radford et al., 2021), which is trained on the similarities between images and texts uses InfoNCE loss.

Moreover, the paraphrase detection and sts are very similar tasks in the sense that if a pair of sentences are paraphrase of each other, this implies that they have similarity 4 or 5 over 5 which can be used in sts tasks. So either wisely combining datasets or using common final layers can improve the performance of both tasks. We leave it as a future work.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Yuki Arase and Junichi Tsujii. 2019. Transfer fine-tuning: A bert case study. *arXiv preprint arXiv:1909.00931*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alena Fenogenova. 2021. Russian paraphrasers: Paraphrase with transformers. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pages 11–19.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Sayyida Tabinda Kokab, Sohail Asghar, and Shehneela Naz. 2022. Transformer-based deep learning models for the sentiment analysis of social media data. *Array*, 14:100157.

Weijiang Li, Fang Qi, Ming Tang, and Zhengtao Yu. 2020. Bidirectional lstm with self-attention mechanism and multi-channel features for sentiment classification. *Neurocomputing*, 387:63–77.

Diwakar Mahajan, Ananya Poddar, Jennifer J Liang, Yen-Ting Lin, John M Prager, Parthasarathy Suryanarayanan, Preethi Raghavan, Ching-Huei Tsou, et al. 2020. Identification of semantically similar sentences in clinical notes: Iterative intermediate training using multi-task learning. *JMIR medical informatics*, 8(11):e22508.

Zhang Min. 2019. Drugs reviews sentiment analysis using weakly supervised model. In *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 332–336. IEEE.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Mark Ormerod, Jesús Martínez del Rincón, and Barry Devereux. 2021. Predicting semantic similarity between clinical sentence pairs using transformer models: Evaluation and representational analysis. *JMIR Medical Informatics*, 9(5):e23099.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Anwar Ur Rehman, Ahmad Kamran Malik, Basit Raza, and Waqar Ali. 2019. A hybrid cnn-lstm model for improving accuracy of movie reviews sentiment analysis. *Multimedia Tools and Applications*, 78:26597–26613.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Jimeng Sun and Si Li. 2021. Domain adaptation for medical semantic textual similarity. In *2021 7th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC)*, pages 319–323. IEEE.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jan Philip Wahle, Terry Ruas, Norman Meuschke, and Bela Gipp. 2021. Are neural language models good plagiarists? a benchmark for neural paraphrase detection. In *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 226–229. IEEE.

Xi Yang, Xing He, Hansi Zhang, Yinghan Ma, Jiang Bian, Yonghui Wu, et al. 2020. Measurement of semantic textual similarity in clinical texts: comparison of transformer-based models. *JMIR medical informatics*, 8(11):e19735.