minBERT and extensions for downstream tasks

Stanford CS224N Default Project

Shiqi Xia
Department of Electrical Engineering
Stanford University
shiqixia@stanford.edu

Yixing Jiang
Department of Biomedical Data Science
Stanford University
jiang6@cs.stanford.edu

Abstract

Sentiment classification, paraphrase identification and semantic textual similarity are among those main-stream NLP tasks. The goal of this project is to implement BERT models from basic building blocks and to develop a model which performs well on these three tasks. The baseline method is to use pre-trained BERT encoders and add three task-specific heads at the end for prediction. Following that, six extensions on top of the baseline method were explored, including revising the training scheme, loss functions, model scales, sentence fusion method and adding in-domain training.

Results show fine-tuning the entire model significantly improves performances on all three tasks, compared with freezing the BERT-based encoder. Also, concatenating the two sentences at the beginning to use as input is very effective for sentence pair tasks, compared with concatenating the two sentence embeddings at the end for task-specific heads. Furthermore, it was shown that scaling up helps only if other design choices such as the sentence integration method are sound. Detailed analysis reveals that the model for sentiment analysis is capable at predicting polarity, but sometimes faces difficulty in predicting specific extents of polarity. It also indicates the importance of analysis beyond simple numerical metrics, and the necessity for data quality control. As a pedagogical project, these main findings and analysis are very useful for future projects as they can inform future project planning and design choices.

1 Key Information to include

Our mentor is Gabriel Poesia. There's no external collaborator nor project sharing with other course.

2 Introduction

Sentiment classification, paraphrase identification and semantic textual similarity are among those main-stream NLP tasks with publicly available benchmark data-sets. Sentiment Analysis task deals with determining whether sentence is positive, negative or neutral. Paraphrase Detection task deals with examining two text entities and determining whether they have the same meaning. Semantic Textual Similarity task deals with comparing two texts and determining how similar they are.

BERT models is a popular choice for these three tasks as its pre-training scheme for language representation allows **bidirectional** context extraction. Furthermore, empirical evaluations show that a pre-trained BERT model can be easily fine-tuned to tackle a variety of downstream tasks such as entailment classification, question answering. Therefore, implementing BERT models from basic building blocks is a good way to dive deep into the technical details of BERT models.

However, there are many new advances after the initial introduction of BERT. Therefore, this project also aims to explore several extensions on top of the baseline BERT model to develop a model which

performs well on the above mentioned three tasks. We have experimented with six extensions, and they will be detailed in the Approach section.

3 Related Work

BERT is a transformer-based NLP model designed to pre-train on textual data and then fine-tune to work for downstream tasks (Devlin et al., 2018). It understands the context and meaning of words in sentences bidirectionally, resulting in state-of-the-art performance on many NLP benchmarks. There are some follow-up work after BERT to explore its potential. For example, RoBERTa improves the pre-training scheme via dynamic masking, larger batch sizes, removal of next sentence prediction and introduction of BPE tokenization (Liu et al., 2019) . StructBERT proposes two new pre-training tasks: recovering shuffled tokens and predicting sentence ordering (Wang et al., 2019). DistilBERT focuses on computational efficiency and distills BERT models into smaller models while preserving performances (Sanh et al., 2019) .

However, these approaches require new pre-training of BERT models, so they are infeasible for this project. Therefore, we decided to focus on the fine-tuning to downstream tasks. Sun et al. (2019) looks for the proper fine-tuning methods in several ways, including different fine-tuning strategies, further pre-training BERT on in-domain data, and fine-tuning BERT with multi-task learning for the target task. It shows that further in-domain pre-training and fine-tuning for the target task can significantly improve the performance.

In addition, although Sentence-BERT focuses on sentence-level embeddings to retrieve similar sentences efficiently, it shows cosine-similarity between embeddings is indicative for semantic meaning, thus can be used for semantic textual similarity prediction (Reimers and Gurevych, 2019) .

4 Approach

As the aim is to study extensions of BERT models, the BERT architecture was used thorough-out the project, and it was detailed in Devlin et al. (2018). The implementation was coded from basic building blocks rather than using existing packages. The final output of BERT is a 768-dimensional vector representations for each token in the input sentence. The first token of every sequence is consistently a unique classification token [CLS], whose final hidden state contains the whole information of the sentence and is often utilized as the overall representation for sentence level tasks.

Cross entropy was used as as the loss function for sentiment classification and paraphrase detection, and mean square error loss was used for semantic textual similarity as it's directly related to correlation coefficient.

The baseline method used a frozen BERT encoder for extracting individual sentence-level features, and specifically the embeddings for the [CLS] token. Three two-layer MLP with ReLU activation were used for the three downstream tasks, taking sentence embeddings as input. For sentence pair tasks (paraphrase detection and semantic textual similarity), the two embeddings of individual sentence were concatenated and passed to task-specific heads. These three task-specific heads have a scale-down factor of 32, meaning the hidden dimension in the two-layer MLP is 24 for sentiment analysis and 48 for the other two sentence pair tasks.

Six extensions over the baseline method were explored, and they were added in an accumulative manner. It means the extensions which improve performances will be kept when exploring the following extensions.

Extension 1: fine-tuning the entire model.

Although the pre-trained BERT encoder was trained on a large corpus, the embeddings might not be optimal for specific downstream tasks, so this extension utilized an independent BERT encoder for each individual task and un-freezed the BERT encoders and trained them together with task-specific heads. There seems no strong shared structure among the three downstream tasks, so the parameters for the three BERT encoders were not shared.

Extension 2: scaling-up to BERT-large model.

Many recent studies have demonstrated the benefit of scaling up for deep learning-based NLP models, so this extension changed the pre-trained BERT encoder from base to large, increasing the number of parameters from 1.1×10^8 to 3.4×10^8.

Extension 3: using cosine loss for similarity prediction (Reimers and Gurevych, 2019).

Instead of using a MLP to calculate similarity from concatenated sentence embeddings, this extension calculated cosine similarity directly from individual sentence embeddings, leading to similarity scores within ranges [0, 1]. The labels were normalized into [0, 1] ranges before calculating the loss. Given sentence embeddings x_1 and x_2, the cosine similarity was calculated as

$$\texttt{cosine similarity} = \frac{x_1 \cdot x_2}{max(||x_1||_2 \cdot ||x_2||_2, \epsilon)}$$

Extension 4: integrating sentence pairs at the beginning.

All previous method used late fusion as the two sentences in a pair was only integrated after BERT encoder before task heads. This extension uses early fusion and joins the two sentences in a pair with [SEP] token as the input to BERT encoder. It then uses the representation of the [CLS] token for the combined input sentence for sentence pairs tasks, including paraphrase detection and semantic textual similarity. This allows the BERT model to directly extract contextual information from the other sentence.

Extension 5: adding in-domain pre-training.

As the pre-trained BERT is used on BookCorpus and English Wikipedia, there might be some domain gap between the text for pre-training and the text for downstream tasks. For example, movie reviews might not be very common in either book chapters or Wikipedia pages. Therefore, this extension conducted additional pre-training using the text for downstream tasks directly. The pre-training task used was masked language modeling, which is to predict the original tokens for those 15% tokens masked off based on context.

Extension 6: scaling-up to BERT-large model with revised design choices.

The final extension was to scale up the model from BERT-base to BERT-large with previous extensions incorporated. The comparison between Extension 3 and Extension 6 can inform the conditions where scaling might help.

5 Experiments

5.1 Data

For the improved BERT model for downstream tasks, the Stanford Sentiment Treebank (SST) dataset curated by Socher et al. (2013) is used for sentiment analysis, the Quora dataset is for paraphrase detection, and the SemEval dataset is for semantic textual similarity. SST dataset consists of 11,855 single sentences from movie reviews with 5-class labels (negative to positive). Quora dataset consists of 400,000 question pairs with labels of whether they are paraphrase of each other, and it is of note only a subset of Quora dataset is used in this project. SemEval STS Benchmark dataset consists of 8,628 different sentence pairs with similarity scores (unrelated to equivalent meaning). The following table shows the specific sizes for different splits.

Dataset	Task	Train Size	Dev Size	Test Size
SST	sentiment analysis	8,544	1,101	2,210
Quora	paraphrase detection	141,498	20,212	40,431
SemEval	semantic textual similarity	6,040	863	1,725

5.2 Evaluation method

Accuracy will be used to evaluate sensiment classification and paraphrase detection tasks, and Pearson's correlation coefficient will be used for Semantic Textual Similarity. The three individual metrics will be weighted to get the final leader-board score. Ablation studies will also be conducted to show the effect of design choices.

5.3 Experimental details

The following table contains the detailed configurations of the experiment, here "Ext." is the acronym for extension. The triplets in the table refer to different values used for each individual task, while a single number means it was shared among the three tasks. The three task heads are all two-layer MLP with ReLU activation, except in extension 2 where there is no task head for semantic textual similarity. The number of training epochs was selected based on validation metrics, so it would be increased if the validation metrics were still improving at the end of training. It is of note that the training time for Ext.3 only includes one task (semantic textual similarity).

Configuration	Baseline	Ext.1	Ext.2	Ext.3	Ext.4	Ext.5	Ext.6
BERT model	base	base	large	large	base	base	large
Freezing encoder	True	False	False	False	False	False	False
Learning rate	1e-3	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5
Epochs	5	5	5	10	5	5	(5,5,10)
Batch size	32	32	(64,16,64)	32	(64,16,64)	(64,16,64)	(64,8,32)
Dropout rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Task-head hidden dim.	24	24	32	0	24	24	32
Training time (mins)	43	119	284	22	91	88	335

5.4 Results

The following table reports the quantitative results evaluated on the development set. The SOTA results were evaluated on the test set and were taken from PapersWithCode, but are included here for benchmark purpose.

Configuration	Acc.(%) on SST	Acc.(%) on Quora	Corr. on SemEval
Baseline	37.1	69.0	0.235
Ext.1	52.6	79.5	0.354
Ext.2	52.0	79.4	0.346
Ext.3	-	-	0.751
Ext.4	52.0	89.0	0.819
Ext.5	52.6	88.6	0.831
Ext.6	52.0	89.1	0.871
SOTA	59.8(sst)	92.4(par)	0.929(sts)

Based on the above table, Ext.1, Ext.4 and Ext.6 turn out to be very effective in improving performances.

It's expected that Ext.1 (fine-tuning the entire model) significantly improves performances on all three tasks. One possible explanation is that the representations learned from masking prediction are not explicitly trained for downstream tasks, making it hard to directly use for prediction.

Ext.4 (integrating sentence pairs at the beginning) works better than expected, and one possible explanation is that the capabilities to incorporate contextual information from the other sentence in an early and complex manner are essential for sentence pair tasks. Another explanation is that the BERT was pre-trained on next-sentence prediction task as well, giving it the ability to identify the connections between two sentences, thus helping the paraphrase detection and similarity prediction.

Ext.2 (scaling to BERT-large) works worse than expected, but later on Ext.6 (scaling-up with revised design choices) works as expected. It demonstrates the importance of optimal design choices in other perspectives before scaling up.

Ext.5 (additional in-domain training) does not lead to a huge boost for performances. One reason might be the pre-training corpus is already large and diverse, so the domain gap between pre-training text and task text is not huge.

The following table shows the performances evaluated on test set. As of March 18, we **ranked #2** in the leader-board. The SOTA results were evaluated on the test set and were taken from PapersWithCode, but are included here for benchmark purpose.

Configuration	Acc.(%) on SST	Acc.(%) on Quora	Corr. on SemEval
Baseline	54.2	89.0	0.873
SOTA	59.8(sst)	92.4(par)	0.929(sts)

It is expected that the performances on test set are similar to those on development set. There's some gaps between our performances and SOTA results, but one caveat is that those methods all have used external training data besides the specific downstream task datasets.

6 Analysis

To further inspect the models beyond the evaluation metrics, some analysis was also conducted to understand the performances on certain sub-groups and examine certain error examples.

To understand the performances on specific classes for classification problems, confusion matrices were generated. Figure 1 shows the confusion matrix for sentiment analysis.

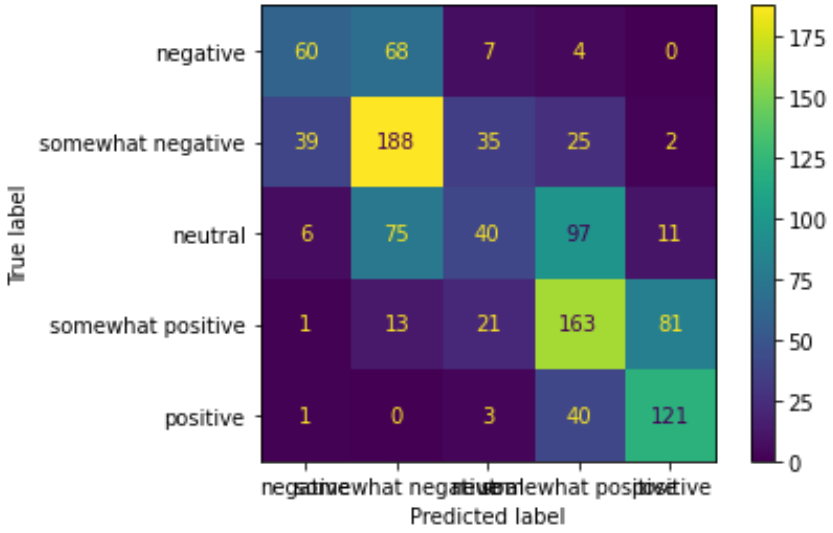

Figure 1: confusion matrix for sentiment analysis on development set

Based on Figure 1, the dominating errors are confusion between 'somewhat negative' and 'negative', and confusion between 'somewhat positive' and 'positive'. It indicates that the model is capable at predicting polarity (positive versus negative), but still faces some difficulty in fine-grained prediction.

The following table shows some examples of sentences labeled as 'somewhat positive' but classified as 'positive'. However, it seems more likely the labels are wrong and the model predictions are correct. It highlights that we should not assume the quality of dataset to be perfect, and label errors are possible and can affect the numberical metrics significantly.

id	sentence
dc6cfe4866d513e284799d0f2	This is human comedy at its most amusing , interesting and confirming .
75a8778fdbb5f1be3d23eea06	Scooby Dooby Doo / And Shaggy too / You both look and sound great .
a7d575c9ed86ea9633660c67e	It 's a lovely film with lovely performances by Buy and Accorsi .

Therefore, it is critical to analyze the NLP system beyond simple numerical metrics. A deep dive into model behavior is essential before potential deployment.

Figure 2 shows the confusion matrix for sentiment analysis and paraphrase detection.

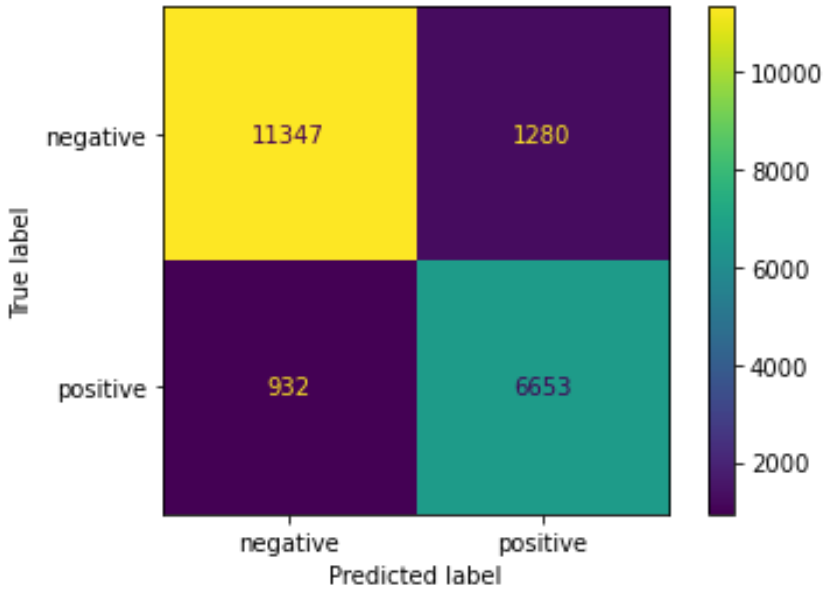

Figure 2: confusion matrix for paraphrase detection on development set

It seems the model tend to make more false positive predictions than false negative predictions, but there seems no clear pattern in the error. There are also some potential label errors, and some examples are shown in the following table.

id	sentence1	sentence 2	label	prediction
fc492d..	Can I use Jio in 3G phone?	How is Jio 3G?	1	0
814d74..	What are the best and profitable ways for saving money?	What are your best ways to save money?	0	1
b92bfff..	What are some really cool working science models I can prepare for my school science exhibition?	I am in 10th grade. I need to make a working model for my science exhibition. What can I make?	0	1

To understand the performances on specific cases for semantic similarity prediction, scatter plots were generated. Figure 3 shows a scatter plot.

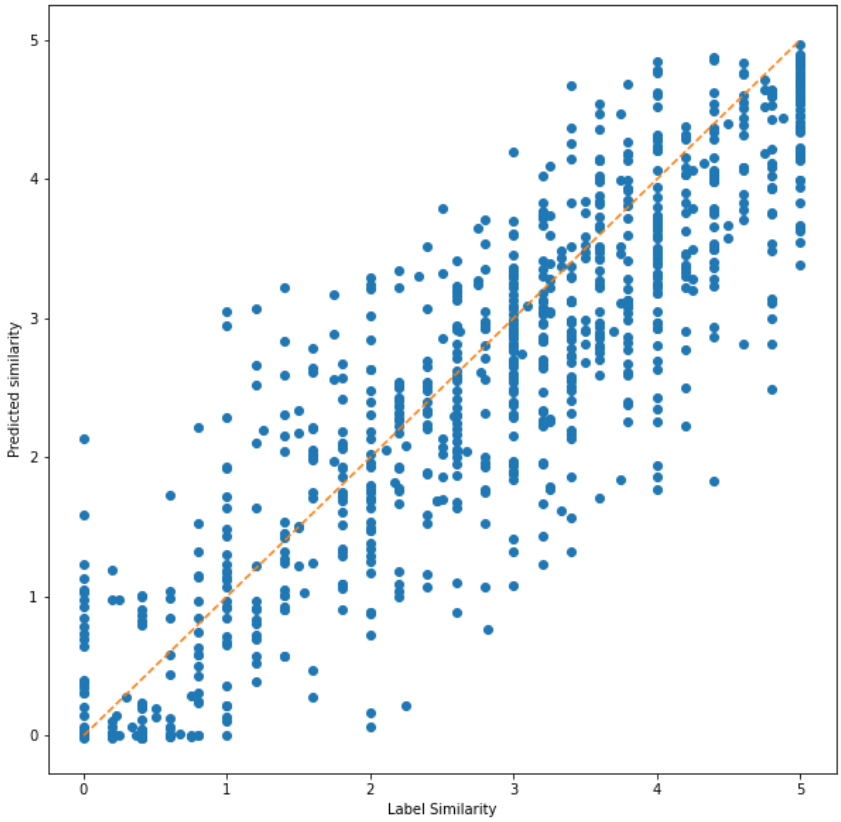

Figure 3: scatter plot for semantic textual similarity prediction on development set

There seems no clear pattern in the scatter plot, and that explains why the numerical metric (correlation coefficient) is pretty high (0.87). The following table show the top 5 examples with largest errors. The second last row seems like a label error.

id	sentence1	sentence 2	label	prediction
14cdb6..	Stock index futures point to lower start	Stock index futures signal early losses	4.4	1.8
53f621..	A swimmer is doing the backstroke in the swimming pool.	a person doing the back stroke in a swimming pool	4.8	2.5
e54438..	non-proliferation expert at the international institute for strategic studies mark fitzpatrick stated that the iaea report – had an unusually strong tenor.	senior fellow at the international institute for strategic studies mark fitzpatrick stated that– the international atomic energy agency plan is superficial.	4.0	1.8
241e4cf..	I think there isn't a general answer.	I don't think there is a single definition.	4.0	1.9
82d73bd..	Work into it slowly.	It seems to work.	0.0	2.1

In general, these detailed analysis shed further light on the model behavior, and they revealed some behavior and patterns which we did not think of before. Also, label errors seem more frequent than we expected, and that highlights the importance of data quality control.

7 Conclusion

In conclusion, this project implemented BERT models from basic building blocks and explored six extensions over the baseline method. Empirical evaluations show strong performances on three tasks: sentiment classification, paraphrase identification and semantic textual similarity.

Main findings include fine-tuning the entire model significantly improves performances on all three tasks, compared with freezing the BERT-based encoder. Also, combining the two sentences at the

beginning to use as input is very effective for sentence pair tasks, compared with concatenating the two sentence embeddings at the end. Furthermore, it was shown that scaling up helps only if other design choices such as the sentence integration method are sound. Detailed analysis also reveals the importance of analysis beyond simple numerical metrics, and the necessity for data quality control, so it is critical to have a detailed inspection to understand model behavior of NLP systems before potential deployment.

One major limitation is this project focused on method development due to time constraints, so the hyper-parameters were not extensively tuned. More fine-grained analysis might also reveal more insights into how the models work and their limitations. As the recent trend is moving to general-purpose large language models trained on huge corpus, it's interesting to see how well GPT-4 works on these tasks out of box and with careful prompt engineering.

References

Papers with code - paraphrase identification on quora question pairs. `https://paperswithcode.com/sota/paraphrase-identification-on-quora-question`.

Papers with code - semantic textual similarity on sts benchmark. `https://paperswithcode.com/sota/semantic-textual-similarity-on-sts-benchmark`.

Papers with code - sst-5 fine-grained classification benchmark (sentiment analysis). `https://paperswithcode.com/sota/sentiment-analysis-on-sst-5-fine-grained`.

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*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

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.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*.