BERT Based Natural Language Inference from Multiple Premises

Jiaxin, Lu
ACM Honors Class, Shanghai Jiao Tong University
lujiaxin@sjtu.edu.cn

Abstract

In this report, we will introduce several methods on handling a novel textual entailment task that requires inference over multiple premise sentences. We use pre-trained BERT model with techniques including fine-tuning, logistic regression, Long Short-Term Memory (LSTM) Networks to embed the sentences and do the classification work. We also evaluate the performance of these methods on this task and analyze the influence of the hyper-parameter.

1 Introduction

Though standard textual entailment task involves two sentences, one as the premise, while the other as the hypothesis, the reality is far more complex. As the increasing number of resources, there is growing need to use multiple independent statements to find out the truth, especially when reading the news, looking through social media posts, or dealing with crime reports. Moreover, such huge scale textual entailment is labor, time-consuming, and human might overlook some details. We therefore introduce artificial intelligence to this task.

In this task, we need to classify multiple independently written sentences and a hypothesis sentence, and we are requested to decide whether they can describe the same scene [Lai *et al.*, 2017]. The train set has 9000 entries and the data to predict has 1000 entries. Each entry is consist of four premise sentences, one hypothesis sentence, and tagged with ’contradiction’, ’neutral’, and ’entailment’.

2 Pre-process

This section will briefly introduce some pre-process we did on this task, concerning the given data set and the techniques we used in this project.

2.1 BERT

BERT [Devlin *et al.*, 2018], abbreviation of Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

pre-trained BERT model

BERT is the first unsupervised, deeply bidirectional system for pre-training NLP. It was trained using only a plain text corpus, and built on bidirectional contextual representations. It uses the *Pre-trained Distillation* algorithm fro pre-training, and has already published several pre-trained models ranging from BERT-Tiny to BERT-Large.

From the given test scores of each model, we found that the larger the model is, the better it performs. Meanwhile, the uncased model performs slightly better than the cased one. Thus, consider the given computing resources (GPU from Google Colab), we choose the **BERT-Base, Uncased** model with 12-layer, 768-hidden, 12-heads, 100M parameters for fine-tuning as well as generating sentence vectors, and we will also use **BERT-Large, Uncased (Whole Word Masking)** with 24-layer, 1024-hidden, 16-heads, 340M parameters to generate sentence vectors.

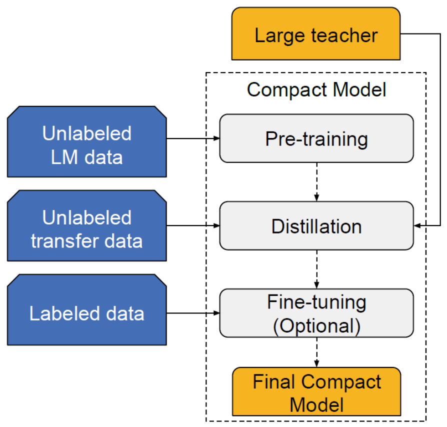

Figure 1: Pre-trained Distillation

BERT As Service

BERT as service [Xiao, 2018] uses BERT as a sentence encoder and hosts it as a service via ZeroMQ. It allows us to use the pre-trained or fine-tuned BERT model for encoding, and require only a short length of code. It is fast, scale nicely on GPU, and very stable for the encoding work. Thus, we choose BERT as service to help generate sentence vector for us.

2.2 Data

Each entry of given data set has 4 separate premises and 1 hypothesis. To meet the requirement of the techniques we used later, we first merge all the premises together with an identifier between these sentences as the notation of different sentences.

Then, for fine-tuning on the pre-trained BERT model, we simply pass the premises to `text_a` and the hypothesis to `text_b`. For BERT as service, we merge the premises and hypothesis together with the required notation `|||` to make them be treated as a pair of sentences.

Finally, since given label in the train set are in the text form, for the convenience of future implementation, we map them as $\{contradiction : 0, neutral : 1, entailment : 2\}$.

For future parameter selection, I printed out the text length information (in character) of the train set and the test set as below.

Range	Length
mean	282.5
min	125
25%	238
50%	273
75%	317
max	761

Table 1: Text Length of the Train Set

Range	Length
mean	280.0
min	140
25%	236
50%	275
75%	312
max	524

Table 2: Text Length of the Test Set

2.3 Sentence Embedding

The goal of sentence embedding is to represent a variable length sentence into a fixed length vector so that we could apply several standard machine learning models on it.

In this project, I mainly used BERT as service to do the sentence embedding with the fine-tuned BERT model, the pre-trained BERT-Base model and BERT-Large model.

To reduce the processing time, I saved the vector generated into the file each time for the convenience of later reuse, so that I only need to load the saved vector when using machine learning models later.

3 Model Selection

This part will introduce some models I used in this project.

3.1 BERT Fine-tuning

Fine-tuning is the process which the parameters of the model are adjusted precisely in order to fit with certain task.

The published BERT repository has already given us a strong tool to do fine-tuning on our own tasks. We only need to modify a processor class in `run_classifier` to do the fine-tuning on this task.

However, some parameter used in the fine-tuning process would significantly influence the outcome of this process so we need to choose them carefully.

First we choose the BERT-Base model since our resources couldn't afford the BERT-Large model. Meanwhile, given the data information above, I set `max_seq_length` to 384, which is sufficient for all the entries in the data set. To avoid the `OOM (out-of-memory)` issue, I set `train_batch_size` to the recommended 12.

To choose the optimal `learning_rate`, I ran a test on a small train set and dev set. I choose the first 1000 entries as the train set and the last 1000 entries as the dev set and then ran the fine-tuning and predict process on them. I tested learning rates $3e-4, 1e-4, 5e-5, 3e-5$ and found out $5e-5$ has the best accuracy on the dev set so I choose it for this parameter.

3.2 Logistic Regression

Logistic Regression is a classical method in machine learning. As we already embedded the sentences, the sentences are transferred into a fixed length vector. Assume the i-th sentence's vector is x_i, we need to train two parameters w and b which satisfy,

$$f_{w,b}(x) = w^T x + b$$

let y be the label of x, we define the loss function,

$$\mathcal{L} = |y - f_{w,b}(x)|^2$$

and we use softmax function as the activation function. We should minimize this loss function, and use the parameter w, b we trained to do the prediction.

We tried this method on several embedding methods to test their performances.

3.3 LSTM

LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Its structure can be summarized as below.

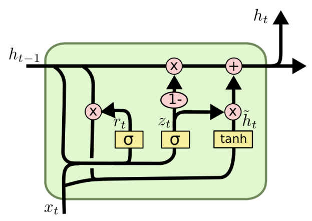

Figure 2: A peephole LSTM unit

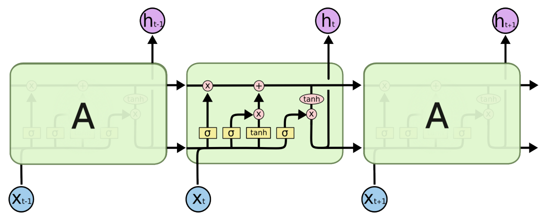

Figure 3: The repeating module in an LSTM

LSTM is widely used in many NLP problems since it can keep track of arbitrary long-term dependencies in the input sequences.

We implemented LSTM only to estimate how it works on this task.

4 Experiment

This section will discuss the detailed implementation of this project, including the tools we used, the models we implemented and the result we get. We will also analyze the results and point out the direction of possible future work.

4.1 Libraries And Tools

We used open source tools TensorFlow [Abadi *et al.*, 2015], Keras [Chollet and others, 2015], BERT, and BERT as service in this project.

4.2 BERT Fine-tuning

I fine-tuned the BERT model with parameters mentioned above. Here is the result.

Vectorize Method	Notation	Test Accuracy
BERT-Base	**BERT-FT**	0.708

Table 3: Result of BERT fine-tuning

I fine-tuned the BERT model several times to adjust the parameters. I found that to finish the whole process of fine-tuning, it took 8 hours on my computer's CPU and only took 40 minutes on the GPU of Google Colab. Such improvement is significant and necessary especially for a larger scale of data.

4.3 Sentence Embedding

With the fine-tuned BERT model, we now have three different BERT models for sentence embedding as listed below.

Vectorize Method	Notation
BERT-Base	**BB**
BERT-Large	**BL**
BERT-Finetuned	**BF**

Table 4: Information of vectorize method

We will use them in the following tests.

4.4 Logistic Regression

I implemented the logistic regression with TensorFlow and Keras. I use this method to test the performance of different embedding models. Here is a table of the detail of my implementation.

Vectorize Method	Tool Used	Notation
BERT-Base	TensorFlow	**LR-BB-TF**
BERT-Large	TensorFlow	**LR-BL-TF**
BERT-Finetuned	TensorFlow	**LR-BF-TF**
BERT-Finetuned	Keras	**LR-BF-KR**

Table 5: Information of logistic regression model

For these test, I divided the given train data into a 8000 entries train set and a 1000 entries dev set to test and select models used.

With the learning rate set to be 0.01, each trained for 1000 epochs, here are the test result of each model.

Model	Test Accuracy	Dev Accuracy
LR-BB-TF	0.577	0.552
LR-BL-TF	0.580	0.565
LR-BF-TF	**0.669**	**0.753**
LR-BF-KR	0.666	0.753

Table 6: Result of different vectorize method under logistic regression model

Below is the visualized loss curve of the model **LR-BF-TF**.

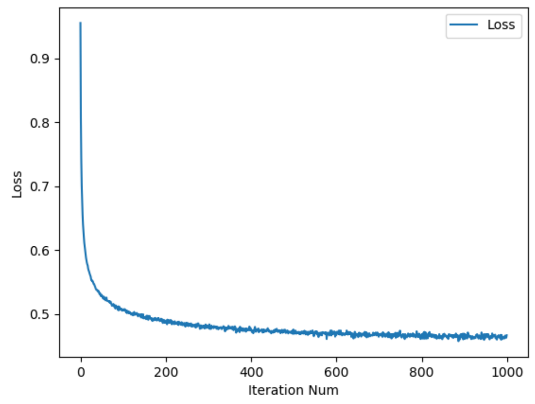

Figure 4: Loss curve of the model **LR-BF-TF**

It's easy to find that the fine-tuned BERT model is significantly better than the BERT-Base and BERT-Large model. Also, the BERT-Large model didn't out-stand the BERT-Base model in our task. We can also find that the BERT-Base and BERT-Large model performed better on the test set while the fine-tuned BERT model performed better on the dev set. This should raise our attention.

I also tested how the learning rate affect the outcome of the logistic regression model. I tested them on the **LR-BF-TF** model, with learning rates of 0.1, 0.07, 0.03, 0.02, 0.015, 0.01, 0.008, 0.005, 0.001. Here are the accuracy and loss of these learning rates.

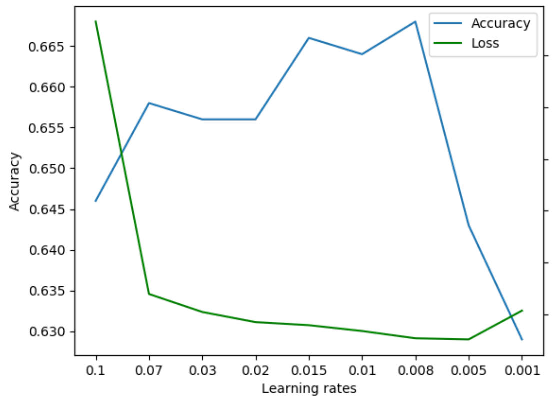

Figure 5: The effect of learning rates of the model **LR-BF-TF**

It's clear to see that the optimal learning rates of the model **LR-BF-TF** is in the range $[0.008, 0.07]$, This range is big enough for us to find a good hyper-parameter for this task.

4.5 LSTM

I implemented LSTM in TensorFlow. I use this to see how LSTM works on NLP problem and what the outcome may look like.

Here is a table of the detail of my implementation and its result.

Vectorize Method	Tool Used	Notation
BERT-Finetuned	TensorFlow	**LSTM-BF-TF**

Table 7: Information of the LSTM model

Model	Test Accuracy	Dev Accuracy
LSTM-BF-TF	0.664	0.705

Table 8: Result of the LSTM model

Below is the visualized loss curve of the model **LSTM-BF-TF**.

It's clear for us that the process of the LSTM model is unstable. Compare to the model **LR-BF-TF**, the loss curve of the logistic regression is much more smooth. Plus, LSTM didn't out-perform the logistic regression model in this task as we expected.

I also tested the accuracy on the train set for the model **LR-BF-TF** and **LSTM-BF-TF** which are listed below.

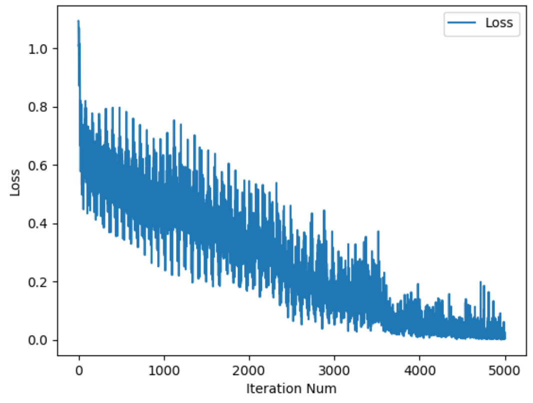

Figure 6: Loss curve of the LSTM model

Model	Train Accuracy
LR-BF-TF	0.825
LSTM-BF-TF	0.992

Table 9: Test accuracy of logistic regression model and LSTM model

We can see that the LSTM model's accuracy on train test is significantly higher than other model, which means its already fit the train data very well that may even lead to overfitting. This is a crucial problem we should take into consideration in our future works.

4.6 Result and Analysis

The best performance of this task is given by the **BERT-FT** model. Also, from the outcome of logistic regression model and LSTM model, we can find that fine-tuning on BERT could significantly raise the accuracy. This shows the necessity of fine-tuning since its not only efficient but also require fewer resources than the pre-train procedure.

However, the logistic regression model isn't quite optimal because from what we deduced, its performance should be better than simply fine-tuning the BERT model. In fact, the accuracy of the **LR-BF-TF** and **LR-BF-KR** model on the dev set is exactly what we expected. Such huge gap (almost 0.09) between the dev accuracy and the test accuracy is concerning. My assumption is that the train set and the dev set is not balanced so that it may lead to the bias in our model. This requests our further examination.

Our tests on the hyper-parameters shows that the logistic regression model is solid in this task, though our method is quite simple that we just manually choose some numbers for the learning rate. It might be possible for us to find the true optimal hyper-parameters with the tools like Bayesian optimization or K-Folding in our future works.

The LSTM has shown the over-fitting problem as I pointed out before. It might means that LSTM is not that optimal for some kind of NLP tasks, and we should always take the feature of the data into our consideration.

5 Future Works

This project is finished in a short time and there are a lot of work left which we could do in the future.

We could re-exam our train data, balance the amount of three labels in the train set and select a better dev set for validation.

If we could have access to larger resources, we can also fine-tune the BERT-Large model and use it for sentence embedding.

There might be many flaws in my implementation requiring examination. Some of the outcomes of the logistic regression model also need explanation.

The process of choosing hyper-parameters in my implementation is quite simple, we could use some mechanic techniques to help choose hyper-parameters.

I didn't do the ensemble learning across different models, but this method worth our attention.

We could also use some other model including ALBERT [Lan *et al.*, 2019], RoBERTa [Liu *et al.*, 2019], or XLNet [Yang *et al.*, 2019] for fine-tuning, sentence embedding and prediction.

6 Conclusion

In this project, we tried some basic models on NLP and learned some mechanism behind them. What this project brings me is that always be bold to try different techniques because these attempt could be useful experience for the future work. Also, be careful about the given data especially for the supervised model. Plus, communicate with each other since it could brings you new idea or help you find the bugs in your code.

Hope this project could be a solid foundation for our future researching.

Acknowledgments

Special thanks to Zonghan Yang and Yi Wang's help to me in this project.

References

[Abadi *et al.*, 2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Chollet and others, 2015] François Chollet et al. Keras. https://keras.io, 2015.

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

[Lai *et al.*, 2017] Alice Lai, Yonatan Bisk, and Julia Hockenmaier. Natural language inference from multiple premises. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 100–109, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing.

[Lan *et al.*, 2019] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.

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

[Xiao, 2018] Han Xiao. bert-as-service. https://github.com/hanxiao/bert-as-service, 2018.

[Yang *et al.*, 2019] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.