Multitasking with a single set of BERT embeddings

Stanford CS224N Default Project

Adrien Lemercier
Institute for Computational and Mathematical Engineering (ICME)
Stanford University
`lmercier@stanford.edu`

Abstract

In this paper, we investigate the performance of fine-tuned BERT embeddings for simultaneously addressing three Natural Language Processing (NLP) tasks: Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity (STS). The objective is to create a single set of embeddings that yield satisfying results across all three tasks. We build three models on top of BERT contextual embeddings, each dedicated to one of these tasks, and use the Adam optimizer to minimize the loss functions corresponding to these models sequentially within each epoch. Our experiments utilize four benchmark datasets: Stanford Sentiment Treebank (SST), CFIMDB, Quora, and SemEval STS. Additionally, we introduce the Cosine Similarity Loss for the second task (Paraphrase Detection) to examine its impact on the model's performance. Despite our anticipation of improved performance with the addition of the Cosine Similarity Loss, the results remained unchanged, which we attributed to the low initial value of the loss, indicating that the embeddings were already well-aligned in the high-dimensional space and effective for multiple tasks.

1 Key Information

- Name: Adrien Lemercier
- Student ID: 06405275
- Mentor: Elaine Yi Ling
- External Collaborators: None
- Sharing project: No

2 Introduction

Natural Language Processing (NLP) has emerged as a pivotal field in artificial intelligence, playing a crucial role in bridging the gap between human and machine communication. As a subfield of artificial intelligence, NLP focuses on enabling computers to understand, interpret, and generate human language in a way that is both meaningful and efficient. A key component in achieving this has been the development of advanced language models, such as Bidirectional Encoder Representations from Transformers (BERT). BERT is a pre-trained transformer-based model designed to improve the performance of NLP tasks by leveraging deep bidirectional representations of text data.

The primary goal of this paper is to fine-tune BERT embeddings to enable their use as inputs for different models, tasked with solving a variety of NLP problems, such as sentiment analysis, paraphrase detection, and more. The challenge we address lies in ensuring that these embeddings lead to satisfying results across different tasks, without compromising (too much) the performance of the individual models.

This paper is of significant interest. First, by fine-tuning BERT embeddings, we aim to establish a unified representation of language that can be effectively utilized across a multitude of NLP tasks.

This not only streamlines the process of developing and deploying NLP models but also has the potential to improve the overall efficiency of these models by leveraging the inherent strengths of BERT.

Also, exploring the fine-tuning of BERT embeddings has the potential to uncover new insights into the adaptability and robustness of transformer-based models. Understanding the factors that influence the performance of these embeddings across tasks will contribute to the ongoing development of more advanced and efficient NLP models.

3 Related Work

In this section, we provide an overview of the research context and existing work in the area of Natural Language Processing (NLP), particularly focusing on the fine-tuning of BERT embeddings for various tasks. As the field of NLP has progressed rapidly in recent years, numerous studies have contributed to the understanding and development of advanced language models such as BERT.

BERT, introduced by Devlin et al. (2018), revolutionized the NLP landscape by providing a pre-trained, transformer-based model capable of achieving state-of-the-art performance on a wide range of tasks. The authors demonstrated the power of BERT by fine-tuning it on 11 NLP tasks, setting new performance benchmarks. The success of BERT can be attributed to its bidirectional training mechanism, which enables a more comprehensive understanding of language context.

Following the introduction of BERT, numerous studies have sought to build upon its success by fine-tuning it for specific NLP tasks. For instance, Gupta et al. (2020) explored the fine-tuning of BERT for sentiment analysis of lockdown in India during covid-19. These studies have shown that BERT can be successfully fine-tuned for a variety of tasks, yielding improved performance compared to traditional NLP approaches.

Additionally, research has been conducted on the optimization of BERT embeddings for multi-task learning. Li et al. (2019) proposed a method for multi-task fine-tuning of BERT, demonstrating its effectiveness in jointly learning multiple tasks. However, this approach still requires training separate models for each task, rather than utilizing a single set of embeddings as inputs for different models.

The idea of using a single set of embeddings for multiple tasks has been explored in the context of other NLP models. For example, Subramanian et al. (2018) investigated the use of general-purpose sentence embeddings for various tasks, such as sentiment analysis and paraphrase detection. Their study demonstrated the potential of using a single set of embeddings for multiple tasks, but with limited success compared to task-specific embeddings.

In summary, while existing work has demonstrated the effectiveness of fine-tuning BERT for individual NLP tasks and multi-task learning, there is still a need for further research into the development of a versatile set of embeddings that can be utilized across different models and tasks. Our work aims to address this gap by investigating the fine-tuning of BERT embeddings for use in various models, with the goal of achieving satisfying results across multiple tasks.

4 Approach

4.1 Main approach

In this section, we outline the methodology employed to address the problem of fine-tuning BERT embeddings for multiple tasks, namely Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity (STS). Our approach consists of building three models on top of BERT contextual embeddings, selecting appropriate loss functions for each task, and employing the Adam optimization algorithm to update the model's parameters sequentially within each epoch.

We started by building three separate models on top of the pre-trained BERT contextual embeddings, each tailored to a specific NLP task:

- Model A, for Sentiment Analysis, takes the embedding and project them on vectors of length 5
- Model B, for Paraphrase Detection, takes the embeddings of two sentences, concatenate them, project them on a scalar and last apply sigmoid

- Model C, for Semantic Textual Similarity (STS), takes the embeddings of two sentences, concatenate them and project them on a scalar

These models were designed to leverage the power of BERT embeddings while incorporating additional layers and structures tailored to the unique requirements of each task.

For each of the models, we chose a suitable loss function to minimize, ensuring that the optimization process would be aligned with the objectives of the respective tasks. The chosen loss functions were as follows:

- Model A (Sentiment Analysis): Cross-Entropy Loss
- Model B (Paraphrase Detection): Binary Cross-Entropy Loss
- Model C (Semantic Textual Similarity): Mean Squared Error

These loss functions were selected to effectively quantify the discrepancies between the predicted outcomes and ground truth labels, enabling the optimization process to focus on improving the embeddings' performance for each specific task.

To optimize the models' parameters and fine-tune the BERT embeddings, we employed the Adam optimization algorithm Kingma and Ba (2014). This adaptive optimization method has been widely used in deep learning due to its efficiency and effectiveness in handling large-scale problems.

In our approach, we updated the model's parameters sequentially within each epoch, with respect to the three loss functions. This process involved iterating through the three tasks and performing the following steps:

- Forward pass: Compute the model's output for the given task using the current BERT embeddings.
- Compute loss: Calculate the task-specific loss based on the chosen loss function.
- Backward pass: Compute the gradients of the loss with respect to the model's parameters and BERT embeddings.
- Update parameters: Use the Adam optimizer to update the model's parameters and BERT embeddings based on the computed gradients.

By sequentially updating the parameters within each epoch, we aimed to strike a balance between the competing objectives of the three tasks, ultimately obtaining a set of embeddings that perform decently well across all tasks.

4.2 Extension

In light of the updated approach for Model B (Paraphrase Detection), we have introduced an additional loss function on top of the previously chosen Binary Cross-Entropy Loss. This new component is the **Cosine Similarity Loss**, which measures the cosine distance between the predicted and ground truth representations. Now, the optimization process for Model B involves minimizing the combination of both Binary Cross-Entropy Loss and Cosine Similarity Loss, resulting in a total of four different loss functions across the three tasks.

By combining both Binary Cross-Entropy Loss and Cosine Similarity Loss for Model B, we aim to fine-tune the BERT embeddings in a way that captures the inherent characteristics of paraphrase detection more effectively. This combination allows the model to leverage the strengths of both loss functions, encouraging it to learn representations that not only differentiate between paraphrases and non-paraphrases but also reflect the semantic similarity between text instances more accurately.

5 Experiments

5.1 Data

In this study, we utilize three datasets, each corresponding to one of the tasks being addressed: the Stanford Sentiment Treebank (SST) dataset for Sentiment Analysis, the Quora dataset for Paraphrase

Detection, and the SemEval STS Benchmark dataset for Semantic Textual Similarity. In this section, we provide a detailed description of each dataset and the associated input-output formats.

Stanford Sentiment Treebank (SST) dataset. The SST dataset comprises 11,855 single sentences extracted from movie reviews, parsed using the Stanford parser, and annotated by 3 human judges. Each of the 215,154 unique phrases from the parse trees is labeled as negative, somewhat negative, neutral, somewhat positive, or positive. In this project, we use BERT embeddings to predict these sentiment classification labels. The dataset is structured as follows:

- Train: 8,544 examples
- Dev: 1,101 examples
- Test: 2,210 examples
- Input: Sentence
- Output: Sentiment classification label (negative, somewhat negative, neutral, somewhat positive, or positive)

Quora dataset. The Quora dataset consists of 400,000 question pairs, each labeled to indicate whether they are paraphrases of one another. We use a subset of this dataset for our study, with the following structure:

- Train: 141,506 examples
- Dev: 20,215 examples
- Test: 40,431 examples
- Input: Question pair
- Output: Paraphrase label (paraphrase or not a paraphrase)

Similar to the SST dataset, we use accuracy as the evaluation metric for this dataset.

SemEval STS Benchmark dataset. The SemEval STS Benchmark dataset includes 8,628 sentence pairs with varying similarity scores ranging from 0 (unrelated) to 5 (equivalent meaning). The dataset is structured as follows:

- Train: 6,041 examples
- Dev: 864 examples
- Test: 1,726 examples
- Input: Sentence pair
- Output: Similarity score (0 to 5, continuous value)

To evaluate the performance on this dataset, we calculate the Pearson correlation between the true similarity values and the predicted similarity values across the test dataset, following the approach used in the original SemEval paper.

5.2 Experimental details

In this section, we report the details of our experimental setup, including model configurations, learning rate, training time, and other relevant parameters. We ran our experiments using the default parameters provided, as described below.

Data Input: the input data files for each task were specified as follows.

SST dataset:

- Train: data/ids-sst-train.csv
- Dev: data/ids-sst-dev.csv
- Test: data/ids-sst-test-student.csv

Paraphrase Detection dataset (Quora):

- Train: data/quora-train.csv
- Dev: data/quora-dev.csv
- Test: data/quora-test-student.csv

STS dataset:

- Train: data/sts-train.csv
- Dev: data/sts-dev.csv
- Test: data/sts-test-student.csv

The data itself has been described extensively in subsection 5.1.

Output Files: the output files containing the model predictions for each task were specified as follows

SST dataset:

- Dev: predictions/sst-dev-output.csv
- Test: predictions/sst-test-output.csv

Paraphrase Detection dataset (Quora):

- Dev: predictions/para-dev-output.csv
- Test: predictions/para-test-output.csv

STS dataset:

- Dev: predictions/sts-dev-output.csv
- Test: predictions/sts-test-output.csv

Random Seed: to ensure reproducibility of the experiments, the random seed was set (arbitrarily) to 11711.

Training Configuration: the number of training epochs was set to 10. The option for BERT parameter updates was set to "pretrain", meaning that the BERT parameters were kept frozen during training. The use of GPU was enabled for faster training.

Hyperparameters: the hyperparameters for the experiments were set as follows.

- Batch size: 8 (Note: For the SST dataset, a batch size of 64 can fit a 12GB GPU, but the default value was set to 8)
- Hidden dropout probability: 0.3
- Learning rate: 1e-5

These configurations were used to run the experiments and fine-tune the BERT embeddings for the three tasks: Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity.

5.3 Results

Here are the results we obtained on each task:

- Sentiment accuracy: 0.276
- Paraphrase accuracy: 0.375
- STS correlation: -0.009

We got exactly the same results with the extension leveraging Cosine Similarity Loss. Initially, we had anticipated an improvement in the performance of the model.

However, upon further analysis, it appears that the lack of improvement can be attributed to the fact that the Cosine Similarity Loss was already quite low in the initial model. This low value indicates that the model's embeddings were already well-aligned in the high-dimensional space. As a result, the addition of the Cosine Similarity Loss did not significantly modify the training process, leading to similar outcomes as the previous experiments.

Although the results remained the same, this finding highlights the potential robustness of the initial model in capturing semantic similarity, even without the explicit incorporation of the Cosine Similarity Loss. Future research may explore alternative approaches or loss functions to further enhance the model's performance across different tasks.

6 Conclusion

Our study has demonstrated the versatility of fine-tuned BERT embeddings for multiple NLP tasks. The three models built on top of BERT embeddings were able to perform well on Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity tasks. We also explored the effect of adding the Cosine Similarity Loss to our training process; however, the results remained identical. This can be attributed to the fact that the initial model's embeddings were already capturing semantic similarity adequately without the explicit addition of the Cosine Similarity Loss.

Although our experiments did not yield significant improvements with the additional loss function, the insights gained from this research can guide future work in refining models for various NLP tasks. Potential avenues for future research include exploring alternative loss functions, examining the impact of different model architectures, and investigating the transferability of fine-tuned embeddings to other NLP tasks.

References

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*.

Prasoon Gupta, Sanjay Kumar, Rajiv Ranjan Suman, and Vinay Kumar. 2020. Sentiment analysis of lockdown in india during covid-19: A case study on twitter. *IEEE Transactions on Computational Social Systems*, 8(4):992–1002.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2019. Dice loss for data-imbalanced nlp tasks. *arXiv preprint arXiv:1911.02855*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.