BERT Extension Using SMART and Cosine Similarity Methodology

Stanford CS224N { Default} Project

Xinwei Liu
Department of AeroAstro Engineering
Stanford University
liuxw@stanford.edu

Daniel Donghun Kim
Department of International Policy
Stanford University
dandhkim@stanford.edu

Victor Cheruiyot
Department of Computer Science
Stanford University
vcheruiy@stanford.edu

Abstract

The aim of this project is to apply adequate extensions to the BERT model to deal with its complexity issue. To do so, we implemented three major extensions. We first implemented a round-robin approach to achieve simultaneous training for three NLP tasks, sentiment analysis, paraphrase detection, and semantic textual similarity. We then implemented a linear layer with concatenation for paraphrasing to distinguish it from the similarity comparison task. We implemented the cosine similarity for similarity comparison **?**. Finally, we implemented the smoothness-induced regularization as proposed by **?** to increase testing accuracy. The results show that these methods both require hyperparameter tuning to increase prediction accuracy and may actually decrease prediction accuracy otherwise.

1 Key Information to include

- Mentor: Shai Limonchik
- External Collaborators (if you have any): No one
- Sharing project: Not shared

2 Introduction

Natural Language Processing is widely studied. With the rise of transformers, researchers come up with many ways to process language using deep neural network models. In our project, we focus on three major tasks, sentiment analysis, paraphrase detection, and semantic textual similarity. These tasks become important in the age of the internet, affecting political elections, the stock market, and even people's fundamental beliefs. To achieve more reliable results and better accuracies, researchers have been experimenting with different deep-learning models. For sentiment analysis, Dang et. al applied RNN, CNN, and LSTM for case studies and concluded that CNN outperforms the other models**?**. Chan et al. also studied the sequential transfer model to tackle the problem of the lacking of data**?**Some of the noticeable development include the ELMo pre-train method by Sarzynska-Wawer et al.**?** to tackle the problem caused by cross-lingual data. Radford et al. improved this work by using transformers with a three-stage fine-tuning framework**?**. However, this model is unidirectional, which decreases training performance when the task requires bidirectional information. Finally, we have the bidirectional encoder representation from transformer (BERT) model, which is used as the baseline of our study. There exist several improvements for BERT, including ALBERT**?** and DistillBERT**?**. The former reduces the model size, and the latter utilizes the compression technique. **?**

For paraphrase detection and semantic textual similarity, there exist several machine learning methods to tackle these two problems, ranging from the traditional CNN and RNN to LSTM and the most recent transformer-based models. The latter, with the parallel computation structure, achieves faster learning and training. As in the case of sentiment analysis, GPT based model and BERT are the two popular approaches with similar pros and cons as discussed for the sentiment analysis tasks **?**.

While the BERT model works efficiently, we aim to improve the test accuracy by implementing several extensions to the baseline BERT model. First, we aim to have an implementation that can perform the three above-mentioned tasks simultaneously. Second, we aim to improve the BERT model testing accuracy by implementing several extensions. Specifically, we implement a linear concatenation layer for paraphrase detection and cosine similarity for semantic textual similarity. Furthermore, we implement the Smoothness-inducing Regularization to improve the testing accuracy, reduce model complexity, and avoid overfitting. In section III, we give the background of the above-mentioned improvement; in section IV, we detail our modification approach; in section V, we present the experiment result; in section VI, we provide an analysis of the result; and in the last section, we conclude our work.

3 Related Work

In this section, we give the background or related work for the extensions we use in our project. Specifically, we provide a brief introduction to BERT, cosine-similarity, and smoothness-inducing regularization.

3.1 BERT

The BERT model consists mainly of three parts, the sentence conversion, the embedding layer, and the transformer layer. In the first part, the algorithm converts the sentence to tokens. The embedding layer consists of three embeddings, token embedding, segment embedding, and position embedding. Token embedding is widely used for language models, and it converts word pieces into corresponding indices in the vocabulary. The segment embedding differentiates sentences, and the position embedding denotes the position of words. Finally, the transformer layer is the classical transformer layer used in deep learning. It consists of the multi-head attention layer, the add-norm layer, the feedforward layer, and another add-norm layer. We apply dropout and masking to the model. **?**.

3.2 Cosine-Similarity

Rather than using the $l2$ distance to compute the difference between two embeddings, we use the cosine-similarity. It is more widely used to capture semantic differences. The equation for such computation is given by the following given by **?**

$$cosinesimilarity = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

It allows the user to capture the directional difference and ignore the magnitude of the vector. As a result, it captures the sentence meaning difference rather than the sentence length similarity. Since we are interested in the meaning of the sentences, we use cosine similarity in the similarity task. We got the inspiration from **?**

3.3 Smoothness-inducing Regularization

The smoothness-inducing regularization adds an extra regularization term after the loss function. This formulation utilizes the KL divergence as the regularization term in the optimization equation to control the complexity by using a smoothness-inducing adversarial regularizer to control the local Lipschitz continuity. More specifically, the regularizer is defined by Equation2 **?**

$$\mathcal{R}_{\mathrm{s}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\widetilde{x}_i - x_i\|_p \leq \epsilon} \ell_{\mathrm{s}}\left(f\left(\widetilde{x}_i; \theta\right), f\left(x_i; \theta\right)\right), \tag{2}$$

Here, f is the model, x_i is the input data, θ is the parameters, and l is the loss function. Loss function is further defined by $\ell_{\mathrm{s}}(P, Q) = \mathcal{D}_{\mathrm{KL}}(P\|Q) + \mathcal{D}_{\mathrm{KL}}(Q\|P)$. The $\widetilde{x}_i$ is calculated by injecting some noise to x_i. In this case, l_s essentially calculates how much the predicted label is different from the predicted label after noise injection. By making it to be a regularize, we are minimizing this difference and making the prediction to be close to the "corrupted" prediction. By doing so, we produce a smoother solution. While this method might increase the training loss, it avoids overfit since now the data are not perfectly fitting the labels. The norm condition makes sure that the corrupted data is not too far from the true data. In the formulation, we use KL divergence, which calculates the difference in probability distribution between two distributions, but in regression tasks, we can simply use a $l2$ norm difference**?**

4 Approach

We have implemented the BERT model with four major modifications,the linear concatenation for paraphrase detection, the cosine-similarity for semantic textual similarity, the round-robin multi-task training technique, , and smoothness-inducing regularization for training.

4.1 Forward Passing Layer

In the forward passing before each task, we pass through a bert layer, a dropout layer, and a pooler layer.

4.2 Paraphrase: Linear Concatenation

For paraphrase detection, we choose to use the linear concatenation method rather than using the cosine-similarity method that we use for semantic textual similarity. In doing so, we differentiate the two tasks. Rather than computing the consine-similiarity, we treat this task as a simple classification task by concatinating the two embedding vectors and passing them through a linear layer. In this case, we avoid having the exact setup for two different tasks.

4.3 Similarity: Cosine-Similiarity

For the semantic textual similarity task, we use the cosine-similarity layer in the training step.

4.4 Training: Round-Robin

By using a round-robin approach, we can train all the tasks in one batch. This approach is better than training one task at a time, which might cause the task trained first to be overridden by the task trained later.

4.5 Training: Smoothness-Inducing Regularization

Finally, we implement the smoothness-inducing regularization using the equation described in the previous section. More specifically, for each sample x_i we do computation as listed in Algorithm 1

We then sum up all the l_smax from all samples to be the regularizer and add it to the loss function multiplied by a coefficient hyperparameter. This algorithm is implemented in a batched version.

5 Experiments

5.1 Data

We used the following datasets for our training and evaluation.

- SST dataset
- Quora dataset
- IMFDB dataset
- STS dataset

Algorithm 1 Smoothness-Inducing Regularization for a single data set x_i

Sample n data points $\widetilde{x_i}$ around x_i
$l_smax = 0$
for A **do**ll the n samples
  if $\widetilde{x_i} - x_i \leq \epsilon$ **then**
    Access $f\left(\widetilde{x_i}; \theta\right)$
    Access $f\left(x_i; \theta\right)$
    compute $l_stemp = \ell_{\mathrm{s}}(f\left(\widetilde{x_i}; \theta\right), f\left(x_i; \theta\right))$ using KL divergence or $l2$ norm for regression
    if $l_smax \leq l_stemp$ **then**
      $l_smax \leq l_stemp$
    end if
  end if
end for

5.2 Evaluation method

Our model's performance has been tested based on its prediction accuracy for three tasks 1) Sentiment Classification, 2) Paraphrase Detection, and 3) Semantic Textual Similarity.

The score has been based on the model's prediction accuracy for the datasets delineated above.

5.3 Experimental details

We used the following variations of the models.

- Cosine Similarity
- Cosine Similarity with Hyperparameter Tuning
- Cosine Similarity with Hyperparameter Tuning and Smoothness Inducing Regularization

First, as baseline, we implemented the cosine similarity extension for the STS.

Second, our best overall score was the one that we implemented using the above method while tuning its hyperparameters. We used the learning rate of 1e-3, dropout probability of 0.3, and the batch size of 100.

Third, we additionally implemented the Smoothness Inducing Regularization model as explained by Jiang et al. The method introduces a small regularization term for its loss during its training phase. We experimented on multiple different epsilon values, for the cutoff to ensure that the random regularized term is introduced frequently enough but does not play a larger part in the loss update than the loss itself. The value originally implemented by Jiang was 10e-5, but we discovered that for our model's implementation, using the same value meant that it was never being introduced in the optimization at all. After a few experiments, we came up with the value of 6.7 * e-4.

5.4 Results

Method	Overall	Sentiment	Paraphrase	STS
Cosine	0.239	0.339	0.376	0.002
Cosine+HypTune	0.346	0.384	0.382	0.272
Cosine+SmthReg+HypTune	0.323	0.399	0.373	0.197

6 Analysis

6.1 Hypotheses

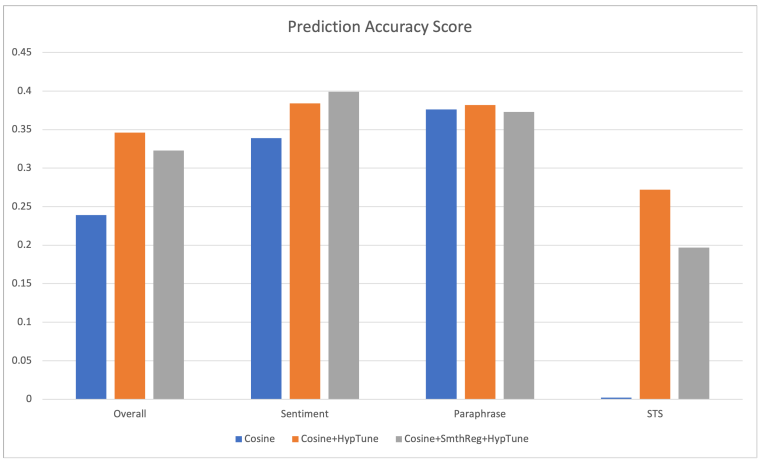
Going into the experiments, we established the following hypotheses.

1. Cosine similarity method will result in high accuracy for the semantic similarity comparison.
2. Smoothness-inducing regularization will result in increased prediction accuracy for the three tasks.

First, we hypothesised that cosine similarity will increase the robustness and the accuracy of our model. Cosine similarity comparison is a widely used method for comparing semantic similarity, as the method by nature captures the similarity between the semantic traits between the sentence vectors, rather than their less relevant traits.

Second, we hypothesised that the smoothness-inducing regularization proposed by Jiang et al. will result in increased prediction accuracy for the three tests due to its prevention of overfitting during the training phase. The smoothness-inducing regularization adds a small regularization term to the loss, and trains on the sum. This is because training on the loss without the regularization term may result in the model overfitting to the training dataset, and may lead to less accurate predictions.

6.2 Results



The experiments led to a result as shown in Figure 1. We added the results on leaderboard: *fuzzy-programmer, fuzzy-programmer-improved*

1. The implementation with cosine similarity comparison for the semantic textual similarity analysis and the tuned hyperparameters led to the best results.

2. Smoothness inducing regularization significantly increases the computation time.

3. Smoothness inducing regularization along with cosine similarity comparison actually led to a mixed result, increasing accuracy for sentiment classification but decreased accuracy for paraphrase and semantic textual similarity analysis.

6.3 Assessment

A surprising aspect of this implementation was that cosine similarity initially led to a poor result in semantic textual similarity analysis, returning a value of 0.002. This may be due to the model not fitting the default hyperparameter.

However, once we used different hyperparameters, including the setting the learning rate to 1e-3 and dropout rate of 0.3, this actually produced the best results, with the STS task results increasing to 0.272, which is the highest of our implementations and highest overall score thanks to the substantially higher score for STS.

Finally, the smoothness inducing regularisation, although it increased the score for the sentiment classification, actually led to a decrease in overall score. This maybe due to the hyperparameter tuning used in the previous implementation not being entirely suitable for the implementation using smoothness inducing regularisation. Also, this might be because the smoothness inducing

regularisation was implemented without the Bregman optimisation, mostly due to computational efficiency reasons.

Bregman optimization prevents aggressive updating by penalizing it using a Bregman divergence term, which means that the update does not allow the iteration at t+1 to deviate excessively from the iteration at t. This wasnot implemented in our model due to two reasons. (1) The loss was continuously going down in each epoch during our training phase for the model with just the smoothness-inducing regularization term. (2) The model using the smoothness-inducing regularization substantially increased the computation time of our training. Maybe the the Bregman optimization term could have led to an increase in overall result.

7 Limitations and Next Steps

Limitations

Our implementation showed the following limitations, mostly resulting from limited computational power within the given time frame. Based on these identifications, the research could be furthered to increase the model's performance in terms of prediction accuracy.

First, most significantly, the limited attempts at hyperparameter tuning might not have led to the optimal numbers to train the model. We identified that the implementation of the extensions at the absence of hyperparameter tuning can rather lead to the low accuracy of the model's predictions and tuned them to lead to increase its accuracy, but we do not know if the hyperparameters could further be optimized.

Hyperparameter tuning seems particularly important especially in combination with the extensions. For example, using a large batch size may decrease the computation time, but can lead to the issue of overfitting. An analysis can be conducted as to how a larger dataset can be used in combination with the smoothness inducing regularization technique, balancing between computation time and overfitting-prevention.

Second, a more diverse and a higher number of datasets can potentially improve the model's performance. The model had the lowest maximum score for the semantic textual similarity analysis. This may be because of the fact that the paraphrase detection trained on 141,506 examples from the Quora dataset, whereas semantic textual similarity was only trained on 6,041 examples from the dataset.

Given that the semantic textual similarity classification is more complex than paraphrase detection as paraphrase detection is a binary classification, using more data for training would likely improve the prediction accuracy of our model.

8 Conclusion

We were able to successfully implement two extensions to BERT. (1) Cosine Similarity Comparison for the semantic textual similarity analysis. (2) Smoothness-inducing regularization for the overall model. Further, our research showed that hyperparameter tuning is an essential step for any model to achieve high prediction accuracy.