Data Augmentation for Multi-task BERT models

Stanford CS224N Default Project

Aditya Chandrasekar
Department of Computer Science
Stanford University
adichand@stanford.edu

Abstract

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained architecture that improves on previous models to create richer, context-dependent representations of words Devlin et al. (2018) More concretely, the hidden layers from the BERT model can provide contextual word embeddings for multiple subtasks, such as sentiment classification, paraphrase detection, and semantic similarity scoring Kokkinos and Potamianos (2017). This project is an exploration of methods to improve BERT architectures for this multi-task classification task, namely by scaling up the number of hidden layers of the model and by experimenting with a variety of dataset augmentation techniques.

1 Introduction

BERT (Bidirectional Encoder Representations from Transformers) was first developed by Google in 2018, showing impressive performance in capturing syntactic and semantic properties of text for downstream NLP tasks such as sentiment classification (Devlin et al., 2018).

These BERT models, however, only targeted one learning task at a time. With multiple prediction heads, we can extend BERT to the multi-task learning regime. Multi-task BERT models can be trained on a variety of tasks, such as sentiment analysis, question answering, and text classification. By training on multiple tasks at once, the model can learn to extract more useful, general features from the text, which can help improve performance on all of the tasks it is trained on independently.

This project is built on minBERT, originally developed by researchers at Carnegie Mellon University [2]. The minBERT project served as a smaller and faster BERT model implementation, packaged with additional pre-trained weights. Deviations from the original BERT model can be found in the scaffolding at https://github.com/neubig/minbert-assignment.

The design space of possible extensions in order to improve multi-task minBERT performance is large, including noise-contrastive learning, adding additional hidden layers, hyperparameter tuning, and pretraining on larger datasets. For the purposes of this work, we decide to restrict the design space to the number of hidden layers and explore various data augmentation techniques.

Although this restricted design space is in part due to working alone on this project, additional time constraints, and lack of deep learning experience, questions about data augmentation techniques have been independently interesting in Natural Language Processing. More concretely, previous work such as Wei and Zou (2019) (EDA) has shown simple augmentations of the original dataset, such as random insertions, synonym substitutions, and randomly swapping words in a sentence according to syntax constraints. Other popular techniques include back translation, which involves translating a dataset to another language and back.

Since some tasks are more word-order dependent than others, and since different data augmentation techniques preserve different structures, the minBERT project also provides a platform to explore the affinity between augmentation technique and task, and how these augmentation techniques or tasks in sum come together to create richer contextual embeddings.

2 Related Work

2.1 Scaling BERT

In the abstract, work exploring how BERT and GPT models scale with the number of hidden layers includes Shen (2022), which proposes a method for scaling BERT up beyond their original 10-20 layer implementations, reporting improvements on question-answering tasks on the Quora Question Answering dataset. However, since the number of hidden layers depends on the task being modeled, we can glean more insight from task-specific articles. These include Alaparthi and Mishra (2021) for a survey of sentiment analysis approaches and Zaib et al. (2021) for work done with the Stanford Treebank dataset.

Besides task-specific work, papers such Jia et al. (2021) show BERT working in the multi-task regime, and show a more principled comparison of various ways of performances that come from varying the number of hidden layers across tasks.

2.1.1 Data Augmentation

Easy Data Augmentation

Further, previous work on data augmentation includes Wei and Zou (2019). The authors show that simple augmentations of the input text can improve performance on a variety of NLP tasks in both pre-training and fine-tuning use cases. In the paper, the authors introduce and evaluate the following augmentations:

1. **Synonym Replacement** Randomly pick a word in the sentence and replace it based on hand-crafted word synonym lists
2. **Random Insertion** Randomly insert a word into the original text. Optionally, for extra computational cost, ensure that this word can be inserted in a way that maintains grammaticality by searching over syntax trees.
3. **Random Swapping** Randomly swap two words in a sentence. Similarly, we can optionally constrain our swapping to valid syntax trees.

Back-translation

Another popular set of techniques for data augmentation in Natural Language Processing is back-translation Corbeil and Ghadivel (2020). Backtranslation involves translating a sentence or text from one language into another using a pre-existing machine translation model and then translating the sentence or text back into the original language using another model. In doing so, we hope to keep the original sentence semantics while offering variety in vocabulary and syntactic structures.

People normally use English to French models for back-translation since both have a large amount of training data, and both have grammatical dissimilarities that allow for novelty in back-translated output.

3 Approach

3.0.1 Baseline minBERT Model

minBERT and AdamW

This project is based on minBERT, originally developed by researchers at Carnegie Mellon University [2]. The minBERT project served as a smaller and faster version of the BERT, with comparable performance to the original model, for students to implement. Deviations from the original BERT model can be found at https://github.com/neubig/minbert-assignment.

Another component of the minBERT approach is the AdamW optimizer. Adam is an optimization algorithm that combines adaptive learning rate methods and momentum-based methods to optimize gradient descent-based neural network training [3]. It maintains a decaying average of past gradients to adjust the learning rate for each parameter. AdamW is an extension of Adam that decays the parameters by some weight decay times the learning rate upon every timestep [4].

3.1 Multi-task Classification

In order to extend the BERT embeddings obtained from the working minBERT model, I tested a baseline multi-task classifier by adding a single hidden layer downstream of the embedding layers. More concretely, the embeddings for the sentiment task is fed into a hidden layer which is then fed into 5 output logits, each corresponding to a sentiment class. For paraphrase detection and semantic similarity, we instead have two embeddings, one for each sentence. To resolve this, we first concatenate the embeddings, but then follow the same architectural pattern: add a hidden layer downstream, and end in a single logit. It is worth noting that the weights of the model are shared across all three tasks, so backpropagated gradient updates for each task influence each other.

3.2 Datasets

Sentiment Classification

For the sentiment classification task, we use the Stanford Sentiment Treebank dataset, with 8,544 training examples, and the CFIMBD dataset, with 1,701 training examples [5].

The 728-dimensional minBERT embedding can be used as feature for sentiment detection by simply adding a single hidden layer on top of the embedding layer. This hidden layer maps the high-dimensional embedding to five logits, one for each sentiment class in the original dataset (strongly negative, negative, neutral, positive, and strongly positive).

Paraphrase Detection

For the paraphrase detection class, we use the Quora dataset of labeled pairs, with a total of 141,506 examples [6].

By concatenating the minBERT embeddings of two sentences and feeding them a final hidden layer with one output, the resulting final neuron is able to access the semantic meaning of both sentences, enabling detection of paraphrases based on their similarity. In a sense, this baseline is equivalent to building a logistic regression on the concatenation of the two embeddings.

Semantic Textual Similarity

The STS task is trained on the SemEval STS dataset, which provides 6,401 training examples of sentence pairs and their semantic similarity scores [7].

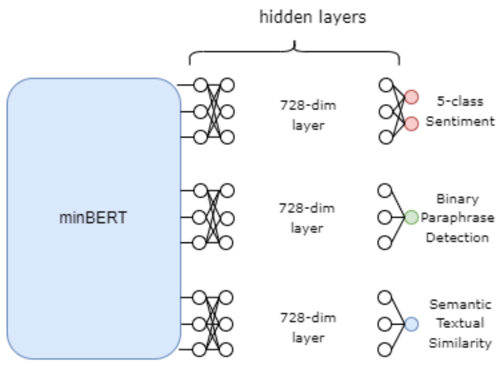
The basic baseline can be implemented similarly to the paraphrase detection, by adding a linear layer to the concatenation of both embeddings to create a similarity logit for backpropagation.

3.3 Scaling up BERT

After the baseline was implemented, we wanted to explore ways of improving the overall performance of the multi-task BERT model. Since, up to a point, model performance tends to increase with model capacity, we decided to explore how performance was affected by scaling our method up.

When deciding how to do so, we first consider adding additional hidden layers to each attention head. Intuitively, the addition of hidden layers to attention heads allows for each head to extract more information or detect more patterns in the pre-trained BERT embeddings. Although this does not necessarily help the fine-tuning of BERT, since it reduces the amount of total shared parameters, it provides insight into which tasks are able to make the most use of the pre-trained BERT embeddings outside of the box.

For our search space of various architectures, we scale up the number of hidden layers uniformly across all the tasks, fixing the dimension of each to 728:



3.4 Data Augmentation Implementation

For our implementation of the various data augmentation techniques, we make use of various NLP libraries that provide different functions for augmentations implemented in the previous literature. Although the library `textattack` provided a large variety of useful augmentation techniques, their implementation was too computationally complex for our use case. More concretely, they would do a deep search of a text perturbation graph to generate new sentences, in order to check certain constraints on the outputted text's syntax.

Since we were curious to see how data augmentation may differentially affect tasks that depend on word order, we decide to try simpler augmentations that may have fewer guarantees on the output syntax tree. As such, we leverage the library `textaugment` to get the extensible implementations of the following augmentations (Marivate and Sefara, 2020):

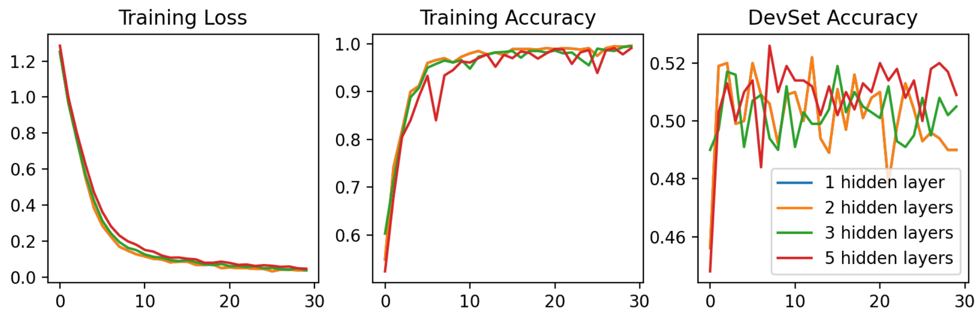
- **Synonym Replacement (WordNet Backend)**: Replace Synonyms according to EDA, replacing words based on pre-computed lists.
- **Synonym Replacement (Word2Vec Backend)**: Replace synonyms by substituting in words based off of the cosine similarity of word embeddings
- **Random Swapping**: Randomly replace two words in the sentence.
- **Random Insertion**: Randomly insert a word in the sequence.
- **Backtranslation**: English to French to English translation of text.

Since we wish to investigate how these techniques affect BERT's representation of sentiment and syntax, as well as the output results, we augment both the Stanford Sentiment Treebank and Semantic Textual similarities tasks with these augmentations, the details of which are elaborated on in the experiments section.

4 Experiments

4.0.1 Scaling Results

For our experiments, we scale up the hidden layers and get the following loss dynamics over time, when fine-tuning the model on our augmented dataset:



In total, it looks like our network starts to overfit fairly early, since the loss dynamics look roughly the same for many of the depths, and the devset accuracy fluctuates wildly as the model converges. Further, all the models converge to a high training accuracy, reflecting that the model capacity may be high enough to memorize the training dataset.

As a result, we focus more of our efforts on data augmentation in order to explore a novel area of this design space.

4.0.2 Data Augmentation

Using the data augmentation methods described earlier, we apply different combinations of them to the SST and the STS datasets to create the following datasets:

- **STS-synonym-replacement**: Randomly replace one synonym in each sentence. For each datapoint in the original dataset, generate 10 novel examples, and keep the original datapoint. The overall sentiment should (hopefully) remain unchanged, so keep the label.
- **SST-synonym-replacement**: Add 5 augmentations with the left sentence with replaced synonyms, and 5 augmentations with the right sentence with replaced synonyms.
- **STS-word2vec-replacement**: Randomly replace one synonym in each sentence based on word embedding cosine similarity. Otherwise equivalent to STS-synonynm-replacement
- **SST-word2vec-replacament**: Add 5 augmentations with the left sentence, and 5 with the right sentence.
- **STS-random-swap**: Randomly swap two words in the original dataset to generate 10 additional example sentences.
- **SST-random-swap**: Generate 5 random swap examples for the left sentence, and 5 random swap examples for the right sentence.
- **STS-translation**: Apply backtranslation to generate 10 additional example sentences.
- **SST-translation**: Apply backtranslation to generate 5 additional left sentences and 5 additional right sentences.

Additionally, we also train on the larger Quora paraphrase detection dataset, but we make no modifications to it in order to carefully observe the effects of augmentation on model performance.

4.1 Experimental details

Report how you ran your experiments (e.g. model configurations, learning rate, training time, etc.)

We can evaluate how these datasets influence multitask performance by comparing model performance trained on just the original dataset. In this case, we scaled the original dataset up by a factor of 11 in order to ease model comparison by making the dataset size uniform. We also can train the model on just the SST-augmented dataset, just the STS-augmented, or both. In order to compare different data inputs, we fix the model parameters to the following hyperparameters:

- hidden dimensions = 728
- learning rate = 1^{-5}
- number of pre-training epochs = 50
- number of fine-tuning epochs = 30

These details are the same across both the scaling and data augmentation experiments, except the scaling experiments increase the number of hidden layers for each task head.

4.2 Evaluation method

For our evaluation, we can simply use devset accuracy calculations in order to compare the data augmentation and hidden layers and see which model works better. This accuracy can be easily computed for the sentiment and paraphrase datasets as the average percentage of correct predictions. For the similarity score regression task, we can use the correlations as an accuracy measure.

The following table summarizes the sentiment analysis accuracies found for each data augmentation configuration:

Configuration	Neither	SST Augmentation	STS Augmentation	Both Augmentations
Synonyms	0.50	0.51	0.48	0.52
Word2Vec Synonyms	0.50	0.52	0.49	0.50
Random Swapping	0.50	0.50	0.48	0.49
Translation	0.50	0.52	0.51	0.52

4.3 Final Results

On the final results leaderboard, I submitted a model with 5 hidden layers with random swapping applied to just the test set, which was the model that had the best overall score in the test set out of my possible submissions (30%). In summary, my quantitive results have not shown great improvement over the original minBERT baseline, but I believe that this is to be expected due to the uncertainty of the experiments tried, and the relatively small gains of EDA on some learning tasks, such as sentiment analysis Wei and Zou (2019).

More specifically, most of the augmented models showed the best performance on sentiment analysis, but poor performance on semantic textual similarity (e.g. a score of 53%). This makes sense given the random swapping augmentation, and although different submissions had different performance profiles, this low-scoring submission still managed the best trade-off between tasks.

However, the experimentation with various techniques and their interactions with the different datasets still gives us insight into the relevant of task structure to learnability and allows us to focus further improvements on specific datasets.

5 Analysis

Firstly, our scaling experiments seem to show us that the model runs into overfitting issues our datasets early on. Adding downstream hidden layers does not necessarily monotonically improve model performance. More concretely, it appears that even with a single hidden layer, the model is able to achieve 99% accuracy on the training set. From my understanding, it seems that minBERT has enough capacity to fit the original dataset when fine-tuned. As a result, it seems like techniques at this point may be those less concerned with turning up the number of hidden layers, and first with improving the quality of the training data, regularizing the model, or adding additional domain assumptions. Since adding training data and adding additional domain assumptions could be framed as adding data augmentation, I thought that data augmentation would be an interesting avenue to explore, despite the relatively small gains documented in other papers.

Overall, from our set of experiments, it seems like back-translation is the augmentation technique that works on most datasets in a task-agnostic way – both the general sentiment and syntax of the original sentence seem well-preserved. Further, random swapping and random insertion also seem to hurt the performance of semantic textual similarity more when only applied to the **STS** dataset (e.g. training on **STS-random-swap**). This may point to a higher dependence on the syntactic structure of the sentence for similarity than for sentiment analysis, in expectation. This is also reflected in the fact that performance on the sentiment similarity task rarely degraded, leading us to believe that the task may be less order-dependent.

Otherwise, in some cases, applying the data augmentation to both the (**STS** and the **SST** dataset results in a model with better performance on both than if the model were to be trained on only one at a time. More concretely, although random swapping seems to hurt overall model performance when the model is fine-tuned on the standard **STS** dataset and the augmented **SST-random-swap** dataset, training the model on both **STS-random-swap** and **SST-random-swap** leads to slight improvement on both tasks. There are multiple reasons I believe this may be the case. In general, despite the nature of the augmentation, it could be that adding augmented data added enough dataset variability for the model to stop overfitting to latent patterns in the dataset and learn them properly.

6 Conclusion

In summary, this project was a great opportunity to get acquainted with the end-to-end process of writing deep learning applications for NLP and gaining practice with running principled experiments on models. Although our experiments did not lead to breakthrough model performance, we were able to learn interesting properties about data augmentation and multi-task learning. Further, these properties are simply a stepping stone for collecting information about our learning task, and it would be possible to make some of the following classes of extensions:

- **Syntax-aware Data Augmentations**: Some of the data augmentations applied to the training set were applied in a way that did not preserve the grammatical structure of the original sentence. By doing additional parsing on each sentence, or learning named entities, more sophisticated data augmentation techniques could be tried that preserve more of the original sentence semantics.
- **Composition of Data Augmentations**: Besides where the data augmentations are applied, we can also apply more than one at a time to get a sophisticated range of data augmentation pipelines that we could potentially optimize over.
- **Application to Paraphrase detection**: The augmentations we applied were mostly directed towards **STS** and **SST** so that we could tease out word-order dependence, but more experiments could be run to explore how paraphrase detection tasks may benefit from different augmentations as well.
- **Hyperparameter Tuning**: Since we mostly focused on the search space of data augmentations, there is still a vast space of unexplored hyperparameters to explore. Different tasks could be allocated different numbers of hidden layers, and various regularization parameters and techniques could be added to help overfitting. These techniques and their affinities to certain data augmentation techniques can also be explored.

References

Shivaji Alaparthi and Manit Mishra. 2021. BERT: a sentiment analysis odyssey. *Journal of Marketing Analytics*, 9(2):118–126.

Jean-Philippe Corbeil and Hadi Abdi Ghadivel. 2020. Bet: A backtranslation approach for easy data augmentation in transformer-based paraphrase identification context.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. Cite arxiv:1810.04805Comment: 13 pages.

Qinjin Jia, Jialin Cui, Yunkai Xiao, Chengyuan Liu, Parvez Rashid, and Edward F. Gehringer. 2021. All-in-one: Multi-task learning bert models for evaluating peer assessments.

Filippos Kokkinos and Alexandros Potamianos. 2017. Structural attention neural networks for improved sentiment analysis.

Vukosi Marivate and Tshephisho Sefara. 2020. Improving short text classification through global augmentation methods. In *Machine Learning and Knowledge Extraction*, pages 385–399, Cham. Springer International Publishing.

Dezhou Shen. 2022. Foundationlayernorm: Scaling bert and gpt to 1,000 layers.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks.

Munazza Zaib, Dai Hoang Tran, Subhash Sagar, Adnan Mahmood, Wei Emma Zhang, and Quan Z. Sheng. 2021. Bert-coqac: Bert-based conversational question answering in context. *CoRR*, abs/2104.11394.