Impact of BERT Extensions on Cross-Domain Text Classification

Stanford CS224N Default Project

Arun Karthikeyan
Department of Computer Science
Stanford University
akarthi2@stanford.edu

Michelle Lok
Department of Computer Science
Stanford University
lokm@stanford.edu

Abstract

In the era of Deep Learning, there is a plethora of neural language models that deal with the text classification problem reasonably well but these are almost always expensive to train from scratch. Reducing this expense is one of the many primary motivations to employ pre-trained models like BERT which essentially provide a springboard to near State-of-The-Art (SoTA) results at a fraction of the required computational costs. By leveraging the power of pre-trained language models coupled with a collection of curated fine-tuning techniques like additional cross-domain pre-training based on masked language modelling objective coupled with regularization, we extend the base-bert model to perform well on multiple downstream text classification tasks. We draw on the fine-tuning techniques discussed in Sun et al. (2019) coupled with Joshi et al. (2019) and Jiang et al. (2020) to analyze and report what works for and what hurts the performance.

1 Key Information to include

- Mentor: Gabriel Poesia Reis e Silva, Hans Hanley
- Arun and Michelle started working together after the project milestone with approvals from TAs because Michelle's original partner had to leave their project due to a personal matter. We would like to share Michelle's two remaining late days for a one-day extension.

2 Introduction

Pretrained language models using transformers have become the norm for state-of-the-art Natural Language Processing tasks. Amongst them, BERT (Devlin et al., 2019) is the first to show the importance of bidirectional information. Unlike previous models, BERT is trained to extract deep bidirectional representations by masking around 15% of its input tokens and attempting to predict them. This technique is termed Masked Language Modelling (MLM). For any given input sequence, the MLM objective not only consumes information preceding the [MASK] token, but also the information that succeeds it, this Bidirectional view made BERT standout with impressive performance. Adapting pre-trained models to downstream text classification tasks via additional pre-training and fine-tuning is a popular technique that paves way for state-of-the-art results for most text classification tasks. The popularity itself can be attributed to its economical computational requirements, the ability to allow quick iterations on model development saving hours or even days in the process, its adaptability to a wide variety of text-classification tasks all while also producing state of the art performance. This has also allowed more people from a variety of backgrounds to enter this space nudging it even further in popularity and performance.

In this project, we implement the base BERT model along with additional pretraining and fine-tuning extensions for multiple (cross-domain) text classification tasks. Figure 1 below gives an abstract idea of our system.

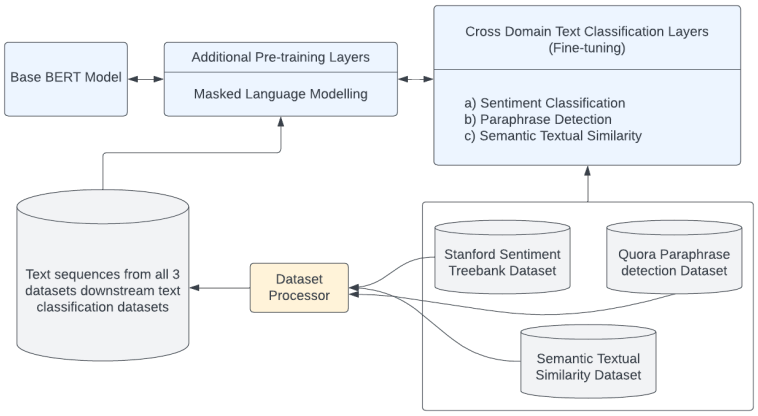


Figure 1: High-level system architecture.

Our implementation of the base-BERT model generates contextual word and sentence embeddings which are further pre-trained with additional Cross-Domain datasets relevant to our downstream text classification tasks. We have analyzed additional pretraining performance based on the original masked-language modelling task (masking individual word-piece tokens) and span-based mask-language modelling task (Joshi et al., 2019). The tail end of our system consists of multiple cross-domain text classification layers for fine-tuning the (additional) pre-trained embeddings. Please refer the Approach and Experiments sections for corresponding implementation details and results.

This model applies the bidirectional training of. It has two prediction goals: 1) prediction of masked words using the original value of the masked words based on the context provided by the non-masked words in the surrounding, and 2) next-sentence prediction to determine whether the second sentence in a pair is the subsequent sentence in the original document. As a secondary goal, regularization and masking alternatives will be explored to find the optimal masking technique to further improve the minBERT model as the original masking technique can be too simple. Many research papers have shown considerable improvements through minor and elegant changes to how masking is applied; e.g. the problem becomes much more interesting if we challenge the network to answer harder masking problems, such as predicting a span of masked tokens.

3 Related Work

Pretrained language models using transformers have become the norm for state-of-the-art Natural Language Processing tasks. Amongst them, BERT is the first to show the importance of bidirectional information. Unlike previous models, BERT is trained to predict words given both the contextual information before and after the unknown token. This technique is termed Masked Language Model (MLM), as tokens within a given sequence are masked with "[MASK]" token, and the model is asked to predict the original masked tokens in their place. According to the authors, MLMs learn to not only apply information preceding the unknown token but also use the information that succeeds it. Although MLM has achieved state-of-the-art performance at its time, it's far from perfect. First, as the authors noted in their paper (Joshi et al. (2019)), there is a mismatch of the inputs seen during pretraining and fine-tuning, as the "[MASK]" token only appears during pretraining. Second, the mask generation scheme is at its infancy. BERT only proposed a single configuration of MLM. It left many potentially useful questions unanswered: What is the best unit (sub-words, words, phrases, or full sentences) of masking? What proportion of the text should we mask? Numerous follow-up papers propose different versions of MLM that show meaningful improvement from the original method. This area of research has been refining our pre-training approach to put constraints on our models and avoid overfitting and generalize better on languages.

SpanBERT explores a more optimal MLM by changing the unit of masking to sub-word to segments of words. It tries to challenge the model to generalize more and depend less on closer words because it's intuitively harder to guess multi-word phrases over a single word, especially if an entity spans

multiple words. For example, the original BERT masking representation for "The Rolling Stones" has a mask that looks like "I went to The [MASK] Stones concert last weekend" which is much easier to learn than the spanBERT's "I went to The [MASK] [MASK] concert last weekend". Research in this area has shown different masking approaches in pre-training have implications on what the transformer learns about languages. The reason SpanBERT has been chosen amongst the other papers is the intuition that is applied to modifying the network. Furthermore, the idea behind SpanBERT is very intuitive yet very powerful. SpanBERT is known to perform better than the base model on question-answering tasks. We wanted to explore if this improvement is also transferable to our three tasks.

4 Approach

4.1 Building the BERT Model

The first half of the project is dedicated to building primary components that make up the BERT model, the starter code is provided in (2023, a) and additional details to get started on building BERT are provided in (2023, b). At a high-level we will be working with the BERT-base variant that contains 12 tranformer layers, an embedding/hidden-state size of 768, multi-head attention with 12 heads. It approximately contains 110 million params.

4.1.1 Tokens & Embeddings

Instances of input dataset are segmented at the subword level using word-piece tokenization and then mapped to their corresponding word-embeddings coupled with their position-embeddings. We prepend each input line with the '[CLS]' token which is designed to model the representative embedding for the sentence that follows it. Another important special token is the '[SEP]' token which is used to mark the end of a particular sentence. And finally, input sequence length is padded using the '[PAD]' token for consistency across all samples.

4.1.2 BERT Transformer

The transformer employs multi-headed self-attention Vaswani et al. (2017) which aims to capture embeddings across multiple contexts of meanings. Every transformer contains for ResNet connections which allows it to propagate signals across multiple layers without having to worry about the vanishing gradient problem. This architecture is made more efficient with LayerNormalization, this allows the Transformer layers to train smoothly, moving faster towards the optimal state. Figure 2 represents the implemented transformer architecture in detail.

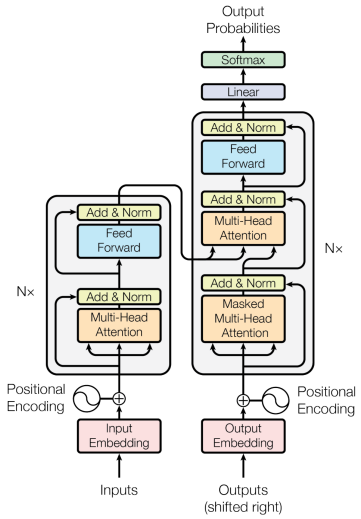


Figure 2: Implemented transformer model architecture from Vaswani et al. (2017).

4.2 Base Bert Model Training

The Base Bert model employs various architectures to train simultaneously on the three datasets. Depending on the dataset, the model follows a different procedure. For the sentiment dataset, a dropout layer and linear layer are used to output five logits for each of the classes, with cross-entropy serving as the loss. The paraphrase dataset applies the pooler outputs of a pair of sentences to a dropout layer, concatenate the results, and feeds them into a linear layer. This dataset uses a binary cross entropy with logits. For the semantic similarity dataset, the model uses the same approach as the paraphrase dataset, but with an additional cosine similarity layer that generates a continuous value between 0 and 5. Its loss function is mean squared error loss. The three tasks' losses are stacked and model parameters are updated in parallel using gradient descent through our Adam optimizer implementation.

4.3 Extension 1) Standard Masked Language Modelling (MLM)

The standard MLM objective was the original objective function (Devlin et al., 2019) used to train the BERT embeddings. We have re-implemented this objective function to perform additional pre-training on a collated dataset based on input sequences from all 3 of our downstream text classification datasets. We've ignored the labels, sentence-ids and other metadata in the downstream datasets and used only the raw sentences/sequences.

We mask approximately 15% random tokens in our input sequences (carefully making sure we don't mask any of our special tokens like [CLS], [PAD], or [SEP]) and run them through the base-BERT model followed by the additional pre-training layer. This additional pretraining layer consumes the last hidden-state of the BERT model and follows with two MLP blocks each containing a fully connected layer, GeLU activation function, LayerNormalization, and a dropout layer. The objective function is designed to predict the originally masked input tokens. The MLM was able to predict 96% of the masked tokens correctly on the dev set.

4.4 Extension 2) SpanBERT

SpanBERT proposes a novel input preprocessing method where it masks continuous segments of words instead of individual sub-word tokens. Instead of recovering individual sub-word tokens, BERT's learning objective becomes recovering the masked sequence of words. It uses a very lightweight secondary network only composed of two linear layers followed by GeLU and softmax layers. The network, then, predicts the masked sequence only using the output embedding from SpanBERT that corresponds to the border of the sequence plus position embedding corresponding to the masked sequence. Please note our extension largely has the same ideas as SpanBERT but differs in some details, such as the loss function.

1. SentenceSpanBertDataset selects a random 20% of sentences and masks a span of 3 consecutive words per sentence, starting at a random index. It then provides the word embeddings of the border word to the left and the border word to the right of the masked span, position ids of the masked tokens, and masked tokens' actual token ids.

2. The training process randomly samples from the Quora dataset to maintain balance before concatenating the datasets of the three tasks and passing them to SentenceSpanBertDataset.

3. The outputs of SentenceSpanBertDataset are then passed to a function to predict a single masked token. The model feeds the concatenated bert embeddings for the two boundary words and the position embedding of the single token at hand into a two-layers of LayerNorm(GeLU(Linear(x))) and a final linear layer that predicts the masked token, followed by a final linear layer that produces logits with a size of 30K representing the vocabulary size.

4. The previous process is repeated for each of the 3 masked tokens. The model calculates and stacks each of the 3 masked tokens' cross-entropy loss between the actual token ids and predicted logits for the 3-word span. The model parameters are updated using an optimizer. To overcome long computation time and OOM errors, the spanBERT pretraining model parameters are saved locally and loaded for the finetuning of the three tasks.

5. The model train randomly samples from the Quora dataset to ensure balance across the 3 datasets before concatenating the 3 tasks' respective datasets together and passing it to SentenceSpanBertDataset.
6. The outputs of SentenceSpanBertDataset are then passed to a function to predict the masked tokens. The model architecture is two-layers of LayerNorm(GeLU(Linear(x))) and a final linear layer that predicts the masked token. The prediction returns the logits with a size of 30K which represents the number of words in the vocab.
7. The model computes the cross entropy loss between the actual 3-word span's token ids and the logits of the predictions.
8. The model parameters are improved via the optimizer. To overcome the long computation time and OOM errors, the spanBERT pretraining model parameters are saved locally. After this additional layer of pretraining using spanBERT was complete, the baton was passed over to finetuning the three tasks by loading the model parameters from spanBERT.

4.5 Extension 3) Regularization

Using pre-trained models for downstream text classification tasks comes with its own caveats. One important and often overlooked aspect of fine-tuning such models is that the size of downstream classification task dataset is often much smaller than the original pre-training dataset which results in a model with the classic high-variance problem for the down-stream task failing to generalize well for unseen downstream data. To analyze this effect for our cross-domain tasks we leveraged the work of Jiang et al. (2020) that discusses a smoothness-inducing regularization that can tackle high-variance/overfitting problems in complex models.
Given a loss function $\mathcal{L}(\theta)$ for a particular text-classification, we update our objective function as follows,

$$min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta) \tag{1}$$

The hyperparameter λ_s controls our regularization penalty $\mathcal{R}_s(\theta)$ which is defined as follows,

$$\mathcal{R}_s(\theta) = \frac{1}{n} \sum_{i=1}^{n} max_{||\tilde{x}_i - x_i||_p \leq \epsilon} l_s(f(\tilde{x}_i; \theta), f(x_i; \theta)) \tag{2}$$

where $\epsilon > 0$ is another hyper parameter. For regression tasks, l_s is simply defined as the squared loss and for classification tasks, it is defined as the symmetrized KL-divergence ($\mathcal{D}_{KL}(P||Q) + \mathcal{D}_{KL}(Q||P)$). Modifying our objective function this way helps avoid aggressive fine-tuning.

5 Experiments

5.1 Data

We use the following datasets provided for use in the default final project, (please refer 2023 (b) for more details).

- (n=12K) *Stanford sentiment tree-bank*[1], with input natural language text sequences and output class ∈ {*negative*: 0, *somewhat negative*: 1, *neutral*: 2, *positive*: 3, *somewhat positive*: 4}.
- (n=200K) *Quora Paraphrase detection datasets*[2], with input natural language text sequences and binary output classification that specifies if the input is paraphrased or not.
- (n=8K) *Semantic textual similarity*(Agirre et al., 2013), with input natural language text sequences and output continuous similarity scale between 1 to 5 with 5 specifying complete equivalence and 1 specifying no-equivalence.

5.2 Evaluation method

The evaluation metric includes the accuracy across the tasks for both dev and test set. Computation time is another important factor to consider as spanBERT took twice as long as base BERT to complete training and yielded only slight increases in one task.

[1] `https://nlp.stanford.edu/sentiment/treebank.html`

[2] `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`

5.3 Experimental details

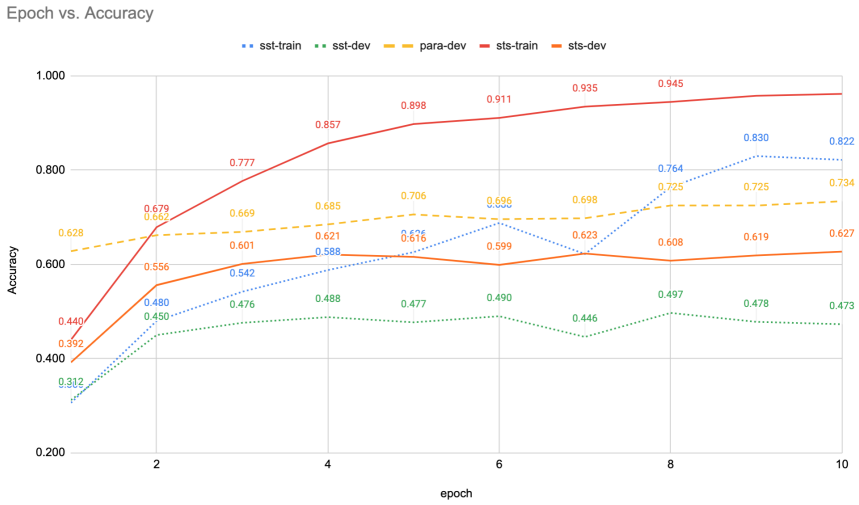
The standard MLM objective for addtional pretraining was trained on all 3 datasets combined with an epoch of 10 and batch size of 128. It trained for approximately 32 mins. Even though we used an epoch size of 10 we were able to see that the model peaked with train and dev accuracy in the first 4 epochs and maintained those values for the rest of the training.
The embeddings resulting from this additional pretraining was later used for further fine-tuning on all the downstream tasks this was again run on 10 epochs and trained for approximately 110 minutes.

The final base Bert and spanBert models in the leaderboard table were trained with epoch of 5, batch size of 5, the learning rate of 1e-5, and trained for approximately 40 minutes and 2 hours respectively. Our team experimented with different sizes of epoch (up to 10), batch sizes (2, 5, and 8), and learning rates (1e-5, 1e-2); however, the difference in accuracies was negligible, at most 1%.

5.4 Results

Leaderboard results					
Implemented models	Type	SST Accuracy	Paraphrase Accuracy	STS Correlation	Overall
base BERT	Dev	0.49	0.682	0.644	0.605
masked LM BERT	Dev	0.482	0.702	0.591	0.591
regularized BERT	Dev	0.476	0.692	0.613	0.593
spanBERT	Dev	0.497	0.725	0.608	0.61
spanBERT	Test	0.516	0.724	0.608	0.616

It is also worth noting that both the standard MLM model and spanBERT model overfitted on the STS training set. The graph below shows the spanBERT model's accuracy across the datasets over several epochs. Overfitting is exemplified by a training accuracy of 96% but a dev accuracy of 62% on the 10th epoch:



6 Analysis

The Standard MLM additional pretraining objective performed exceptionally well at the word-token prediction task and peaked its accuracy at early epochs (train and dev acc of 0.999 at epoch 6 with a batch size of 64). Howeveras reported in the results table, it didn't contribute significantly to the downstream text classification tasks. Similarly when combined with regularization we didn't observe significant improvement in downstream text classification task performance.
Although SpanBERT has demonstrated promising outcomes in research, our experiments showed that it did not significantly outperform the baseline in our sentiment analysis, paraphrase detection, and semantic textual similarity tasks. Several factors may have contributed to the absence of accuracy

improvement when utilizing SpanBERT in pretraining. One possible explanation is the characteristics of the dataset and the task. SpanBERT has been proven to excel in question-answering or generative AI tasks that require predicting or generating the next word or filling in the blank, which reflects the SpanBERT problem objective of predicting text segments. However, our tasks do not involve generating masks but rather focus on analyzing sentence structure and semantics, which may explain the suboptimal results. Furthermore, our SpanBERT implementation employed the original BERT model's vocabulary size to define the model layers' output size, rather than the three datasets' combined vocabulary size which contains twice as many words. This approach can limit the model's performance as unique words that appear in these datasets may not be well represented.

7 Conclusion

Our best model SpanBERT, although it slightly outperformed our model based on standard MLM objective pretraining and our baseBERT model, disappointingly it was only by 1%. Taking the current state of the model, it is not recommended to pursue this pretraining method as it is not well-aligned with the 3 tasks in this project and takes twice as much time to implement and train compared to the base model. Similarly adding regularization into the mix only slightly improves downstream text classification task results (but this is in line with the improvements the original authors (Jiang et al., 2020) are observing). If more time is available and the experiments mentioned in the analysis are completed, it may be considered worthwhile to consider. Keep in mind the original SpanBERT paper (Joshi et al. (2019)) was only able to improve the accuracy by 4 to 5%.

If additional time were available, our team would have wanted to conduct experiments with alternative masking approaches, such as RoBERTa, as well as original ideas that involves incorporating supplementary information, like part-of-speech, into the mask training. We would also consider employing a broader range of hyperparameters for pretraining, including variations in the learning rate or batch size, to assess their impact on the model's ability to learn useful representations for the given task. Surprisingly, the overfitting issue mentioned previously was not fixed by regularization. To mitigate overfitting concerns regarding the STS task, we wanted to explore whether task-specific regularization or early stopping based on the number of steps taken in the optimizer improves accuracy.

References

CS224N Winter 2023. a. Default final project - multitask bert starter code.

CS224N Winter 2023. b. Default final project handout: minbert and downstream tasks.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *CoRR*, abs/1907.10529.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.