Human Language Techlogy Project: Sentimental Analysis task

Xiaoyan Li 627452

August 31, 2023

1 Introduction to the task: Sentimental Analysis

In recent years, there has been an explosive surge in the popularity of community portals across the internet. In addition to content curated by portal editors, user-generated content has become a pivotal component of numerous websites. Users contribute their insights not only through discussions and personal notes in various social web spaces (such as boards and blogs) but also on a large scale by leaving comments and reviews about products and services on various commercial websites. Despite the rapid proliferation of such content, its full potential remains untapped. Information provided by users often lacks organization, and many platforms that facilitate user input refrain from moderating the content added by users.

Sentiment analysis emerges as an endeavor to harness the immense volume of user-generated content effectively. It harnesses computational processing power to systematize the insights derived from user opinions and to subject them to analysis for subsequent utilization. The undertaking of sentiment analysis can be conceptualized as a text classification conundrum. This is because the process entails a series of operations that ultimately culminate in classifying whether a given text conveys a positive or negative sentiment.

Sentiments and opinions can be discerned primarily at three levels: the document level, sentence level, and aspect level. This particular project is specifically aimed at the sentence level. However, while pretrained large language models (LLMs) can offer valuable utility in this task, the challenge lies in effectively transferring the pretrained knowledge in the LLM models to the specific target domain. The gap in data and features distribution between the souce domain used in the pretrained phase for the LLMs and target domains is the major problem. For example, the BERT model is generally pretrained based on BookCorpus and English Wikipedia, which might degrade its performance in a certain specific domain. Consequentely, the need for finetuning LLMs becomes self-evident.

It is worth noting that supervised deep learning tasks are notoriously data-intensive. Fine-tuning all parameters in the LLMs slows down the training process significantly, concurrently making it susceptible to overfitting, particularly in the case of insufficient training data. This project aims at how to address the sentiment analysis task using pretrained LLMs, i.e., BERT, focusing on finetuning the model not just for parameter efficiency, but also for data efficiency.

2 Prior related work

2.1 Bert and prompt tuning

The BERT model [2] consists of a series of stacked Transformers encoders. Since its introduction by researchers at Google in 2018, it has pushed the boundaries of earlier model architectures, such as LSTM and GRU, which were either unidirectional or sequentially bidirectional. BERT considers context from both the past and future simultaneously, a contribution from the so-called "attention mechanism".

Prompt-tuning is an efficient, low-cost way of adapting an large model to target downstream tasks without retraining the model and updating its weights. Prompt-based methods have shown received great attention in few-shot setting, zero-shot setting, and even fully-supervised setting. Current prompt-based methods can be categorized as two branches according to the prompt is whether discrete or continuous.

- Discrete prompts: Discrete prompt is a sequence of words to be inserted into the input text, which helps the PTM to better model the downstream task. Sun [7] constructed an auxiliary sentence by transforming aspect-based sentiment analysis task to a sentence pair classification task, but its model parameters still need to be fine-tuned.
- Continuous prompts: Instead of finding the optimal concrete prompt, another alternative is to directly optimize the prompt in continuous space. The optimized continuous prompt is concatenated with word type embeddings, which is then fed into the pre-trained models. Typically, the model parameters are fixed which means the prompt tokens are also fixed by the model parameters.

This project mainly explored the continuous prompts methods. More specifically, it exprimented two types prompt tuning: the basic prompt tuning [3] and prefix prompting tuning [4]. In [3], sequence classification task is cased as a text generation task, in which the tokens that make up the class label are generated. Prompts are prepended to the input as a series of virtual tokens. While the model parameters are fixed and prompt tokens are also fixed, the embeddings of the prompt tokens are updated independently. Prefix tuning [4] is similar to prompt tuning in [3]. It also attaches a sequence of task-specific parameters to the beginning of the input, or prefix. Only the prefix parameters are optimised and trained, while the pretrained model parameters are kept frozen. The main difference is that the prefix parameters are added in **all** of the model layers, whereas prompt tuning [3] only adds the prompt parameters to the embedding of the model input.

2.2 Deep metric learning

Conventional deep neural networks utilize a simple linear classifier to make decisions on examples. The feature embeddings learned in this case are linearly separable in the feature space. Nevertheless, these methodologies typically require a large amount of data to achieve successful results.

Deep metric learning(dml) is an approach based directly on a distance metric that aims to establish the similarity or dissimilarity between data points. As a result, objects with the same labels are drawn closer, while objects with different labes are pushed apart. This leads to a more compact representation of data points, and thereby enabling robust performance even in low-data settings.

In this project, we will makes use the combination of the following two distance-based losses:

- Triplet loss: Triplet Loss was first introduced in FaceNet [6] and it has been one of the most popular loss functions for supervised similarity or metric learning ever since. Triplet Loss encourages that dissimilar pairs be distant from any similar pairs by at least a certain margin value.
- DisMax Loss: DisMax loss [5] was proposed to improve the out of distribution detection performance by encouraging clustering in a feature space. It is used in conjunction with crossentropy loss and encourages an input to minimise distance to its ground truth class centre, and maximise distances to the other class centres. The class centres are learnt simultaneously with the feature embedding during training.

In this project, we aim to integrate deep metric learning with prompt tuning to examine the potential enhancement that distance-based losses can bring to prompt tuning methods within a low-data setting.

3 Dataset description

The Stanford Sentiment Treebank is a corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. The corpus is based on the dataset introduced by Pang and Lee (2005) and consists of 11,855 single sentences extracted from movie reviews. It was parsed with the Stanford parser and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges.

Binary classification experiments on full sentences (negative or somewhat negative vs somewhat positive or positive with neutral sentences discarded) refer to the dataset as SST-2 or SST binary. Some examples are shown as in Figure 1.

sentence	label
a stirring , funny and finally transporting re imagining of beauty and the beast and 1930s horror films	1
apparently reassembled from the cutting room floor of any given daytime soap	0
they presume their audience won't sit still for a sociology lesson	0
this is a visually stunning rumination on love , memory , history and the war between art and commerce	1
jonathan parker 's bartleby should have been the be all end all of the modern office anomie films	1

Figure 1: Examples of SST2 datasets, credits to [1]

4 Architecture

Our goal in the project is to create a model that takes a sentence as input and produces either 1 (indicating the sentence carries a positive sentiment) or a 0 (indicating the sentence carries a negative sentiment). It can be depicted as in Figure 2:

Under the hood: the model is composed of two models:

- The BERT model processes the sentence, and passes along feature embeddings extracted from it to the next model. If we dive deeper into this process, it first uses BERT Tokenizer to split the sentence into tokens, and the corresponding token IDs from the embedding table are then fed to the BERT base model to get the sentence embeddings.

Figure 2: Architecture of model, credits to [1]

- The next model consists solely of a dropout layer, followed by a linear classifier layer.

For the purpose of fine-tuning the BERT base model, we employ two prompt-based tuning methods: the basic prompt tuning, which only inserts the prefix parameters into the input layer, and the prefix tuning, which inserts the prompt parameters into both the input layer and all hidden layers. Consequently, prefix tuning often involves more parameter tunings. The architecture for model promp tuning and prefix tuning are shown in Figure and Figure fig:Prefix Tuning.

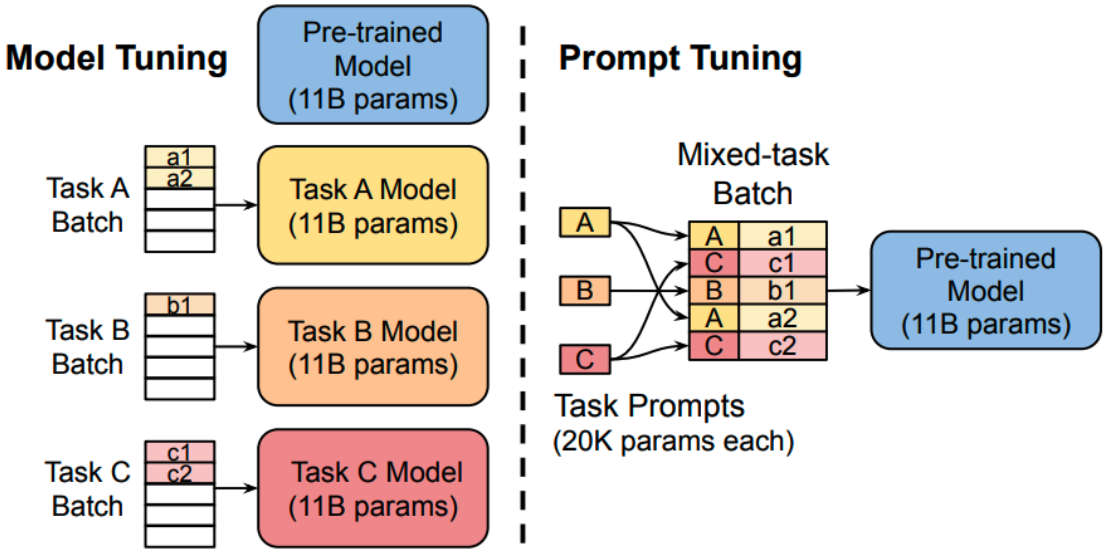

Figure 3: Architecture of prompt tuning, credit to [3]

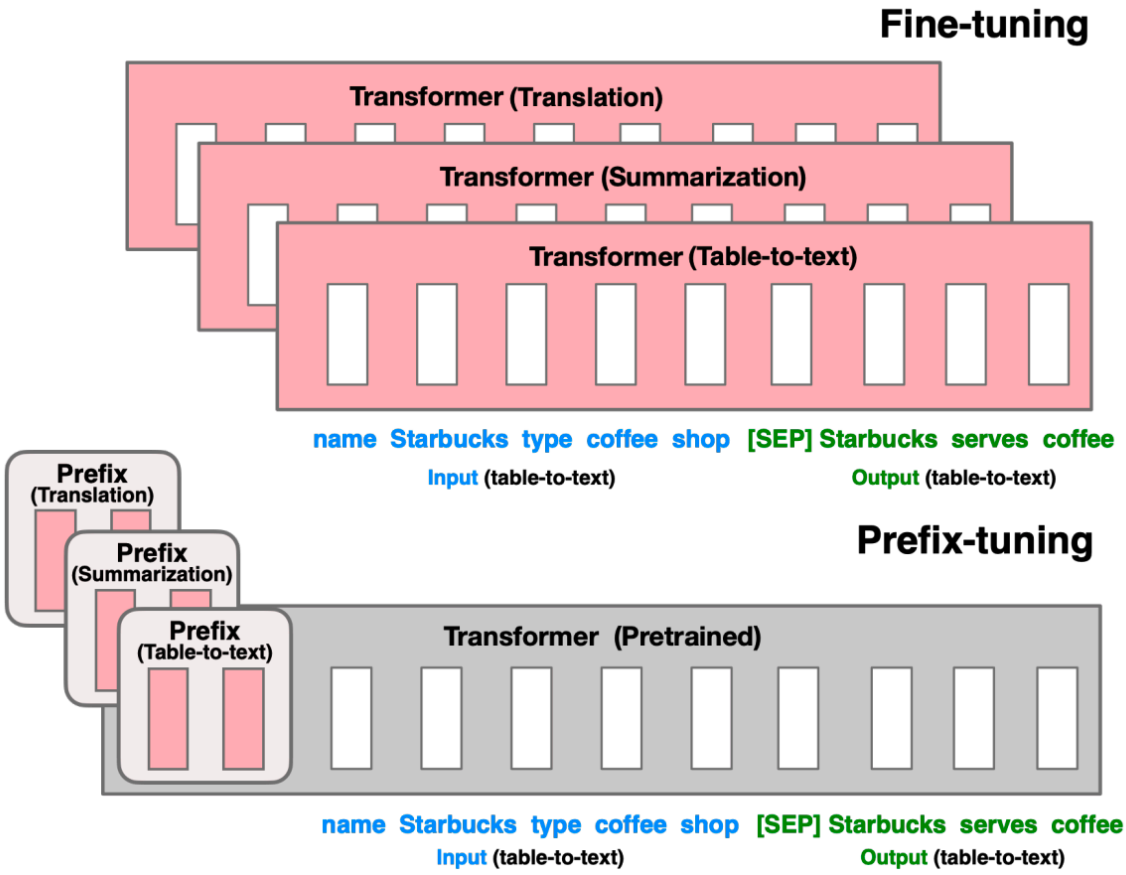

Figure 4: Architecture of prefix tuning, credit to[4]

5 Experimental setup

5.1 Datasets and Metrics

We evaluate the model's performance using the SST2 dataset. Due to limitations in computational resources, we drew a total of 4000 samples from the training set, 2000 samples of class 0 and 2000 samples of class 1, ensuring balanced classes. The full devset (868 samples) was used to fine-tune the hyperparameters, while the complete testset was utilized to evaluate the model's performance.

To further investigate the performance of different methods in a low-data setting, we randomly selected 200 samples from the training dataset for training, and 200 samples from the development set for validation. As previously, the evaluation was conducted using the complete testset. Furthermore, the results were measured by averaging the outcomes obtained from sampling with three different seeds. Given that this task fundamentally a classfication task, we report the accuracy as the metric criterion.

5.2 Hyperparameters

At training time, we use the AdamW optimizer and a linear learning rate scheduler, as suggested by the Hugging Face default setup. The hyperparameters we tune include the number of epochs, batch size, learning rate, and prefix length. In the table , we report hyperparameters used to train the models documented in the follow result section.

methods	**learning rate**	**num_epochs**	**batch size**	**num_virtual tokens**
	sufficient data setting	w/o deep metric learning		
fine-tuning	1e-5/1e-5	11/17	16/32	—
prompt tuning	1e-2/6e-3	11/38	16/16	7/7
prefix tuning	1/1e-2	1/32	32/16	10/7
	low-data setting	w/o deep metric learning		
fine-tuning	1	1	1/1	—
prompt tuning	1	1	1/1	10
prefixtuning	1	1	1/1	10

Table 1: Hyperparameter settings

5.3 Results

2. Please note that only the number of parameters for tuning the BERT base model was reported, while the number of parameters for the linear classifier was not taken into account.

methods	**accuracy w/o deep metric learning**	**#trainable parameters(% proportions)**
	sufficient data setting	w/o deep metric learning
fine-tuning	90.94%/91.05%	109M (100%)
prompt tuning	89.40%/89.51%	5K (0.0049%)
prefix tuning	1% /91.05%	184k (0.16%)
	low data setting	w/o deep metric learning
fine-tuning	1	1
prompt tuning	1	1
prefixtuning	1	1

Table 2: Results

Vis photos 200 samples

5.4 Results Analysis and conclusions

In terms of parameter efficiency for tuning the BERT base model, as demonstrated in Table 12 , prompt tuning involves the least number of parameters, followed by prefix tuning in the second place. Prompt tuning is the most parameter-efficient approach, as it solely trains and stores a minimal set of prompt parameters—10, 000 times fewer—compared to training all of the model's parameters.

Regarding performance, we consider two scenarios:

1. When working with abundance data for training and validation, fine-tuning yields the best results. Notably, both prompt tuning and prefix tuning demonstrate performance levels comparable to fine-tuning. Furthermore, deep metric learning exhibits minimal impact on all three methods.
2. In cases of limited samples for training and model tuning, as indicated in the lower section of Table 2, all the three methods tend to undergenerate in a low data condition, Here deep metric learning can significatnly contributes to enhancing results, aligning them more closely with the original performance.

   It is worth noting that with the integration of deep metric learning, prompt tuning achieves a performance on par with fully fine-tuned models that utilize 20 times more data and tunes 10,100 times more parameters.

6 Limitations and directions for improvement

In this project, we have focused on the challenges posed by the low-data setting and explored how prompt tuning can be enhanced by deep metric learning to achieve results comparable to scenarios with sufficient data. However, it should be noted that even though we have made decent progress in improving the performance, the linear classifier layer, serving as the final layer of classification, still encompasses a considerable number of parameters. In cases where we delve further into data scarcity, such as few-shot learning scenarios, the linear classifier layer could potentially become a bottleneck.

To tackle this issue, we propose considering the parameter-free classification techniques. More specifically, we suggest a two-step training approach:

- In the first step, our focus is solely on training the model to extract features using deep metric learning methods or contrastive learning methods. At the end of this step, the learned features should exhibit proximity for examples of the same label.
- In the second step, we can leverage tools like sklearn's NearestCentroid or k-nearest neighbors (knn) to classify unseen examples.

This two-step process holds promise for addressing challenges related to data scarcity and potentially reducing the parameter dependency of the linear classifier layer.

References

[1] Jay Alammar. A Visual Guide to Using BERT for the First Time. `https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time//`, 2018.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[4] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics.

[5] David Macêdo, C. Zanchettin, and Teresa B Ludermir. Distinction maximization loss: Efficiently improving out-of-distribution detection and uncertainty estimation by replacing the loss and calibrating. 2022.

[6] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[7] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. 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 380–385, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.