Michael Liu, Genfu Liu

CS462: Introduction to Deep Learning

Professor Elgammal

5/5/2023

Fine-Tuning BERT for Sentiment Analysis

Abstract:

Natural Language Processing (NLP) has emerged as a significant field in machine learning, enabling computers to understand and interpret human language. It plays a vital role in various applications, such as text summarization, machine translation, sentiment analysis, and more. Sentiment analysis, a subfield of NLP, aims to identify and categorize the sentiments expressed in textual data. This task has gained increasing importance in recent years, owing to the exponential growth of user-generated content on the internet, and has become crucial for businesses, governments, and other organizations to understand public opinion and make informed decisions.

In this project, we will be exploring BERT (Bidirectional Encoder Representations from Transformers), a state-of-the-art NLP model introduced by the researchers at Google AI Language back in 2018, and also using it to enhance the performance of sentiment analysis on the IMDb movie reviews dataset. We fine-tune the pre-trained BERT model using IMDB dataset, aiming to experiment as well as improve the strategies and approaches used in their paper - **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. Our results demonstrated the effectiveness of BERT in capturing complex language patterns and achieving superior performance compared to traditional sentiment analysis methods and other deep learning-based models.

Introduction

As mentioned in the paper, the introduction of BERT by Devlin et al. in 2018 aimed to address the limitations of previous models in understanding the contextual relationships within text. Prior to BERT, many models processed text in a unidirectional manner (either left-to-right or right-to-left), which led to a limited understanding of context and suboptimal performance in various NLP tasks. The goal of the BERT model is to solve this problem by introducing bidirectional capabilities, this means that it is able to process text in both directions, effectively capturing context of the entire text, allowing for a more accurate and nuanced understanding of language. According to the paper, the model works by utilizing a two-step process: pre-training and fine-tuning. During pre-training, BERT learns general language representations using two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). After the pre-training phase, it can be easily fine-tuned on specific NLP tasks using labeled data. These a wide range of tasks, such as named entity recognition, question-answering, language inference, and sentiment analysis, which is what we will be fine-tuning in this project.

Our study focuses on fine-tuning the BERT model for sentiment analysis on the IMDb dataset, which contains 50,000 where half of them are labeled as positive while the other half are negative. The IMDb dataset has been widely used to benchmark sentiment analysis models, making it a suitable choice for our research. By fine-tuning BERT on this dataset, we aim to assess the model's performance and effectiveness for sentiment analysis, as well as explore the challenges and potential improvements associated with the fine-tuning process.

Approach

In this section, we describe our approach to fine-tuning the BERT model for sentiment analysis on the IMDb dataset, including the detailed fine-tuning process we took to create our model. We've also created a Google Colab Notebook where we created the model.

https://colab.research.google.com/drive/1QJb_KHLsofFC4Cvn8xI3FbtiiQjftkkp?usp=sharing

To fine-tune the BERT model on the IMDB dataset, we used the steps as follows:

1. **Data Loading/Preprocessing**
   We first load the IMDB dataset in and split the training set into training, validation and testing subsets as usual. Next, before fine-tuning, the IMDb dataset must also be preprocessed to ensure compatibility with the BERT model. This includes tokenizing the reviews using BERT's WordPiece tokenizer, converting tokens to unique IDs, creating attention masks to differentiate between padding and actual tokens, and truncating or padding the sequences to a fixed length. Additionally, the sentiment labels must be converted into a numerical format compatible with the model.
2. **Loading and configuring the pre-trained BERT model**
   We use the pre-trained BERT base model as the starting point for fine-tuning. The BERT base model consists of 12 layers, 12 self-attention heads, and 768 hidden units, with a total of 110 million parameters. We add a classification layer on top of the BERT base model to perform binary sentiment classification, where the output logits are mapped to the corresponding sentiment labels, as well as a sigmoid function to map the output to a probability distribution over the two classes.
3. **Training the model**
   We first set up some initial hyperparameters that will be used to train the model, including the max sequence length, batch size, number of epochs and the learning rates. The choice of some of these hyperparameters, such as batch size and the number of epochs, will be based on the recommended ones from the paper, while others will be tuned later on. Next, we define the optimizer and the loss function for our training process. We chose the Adam Optimizer and CrossEntropyLoss as the Loss Function for our model. Then, we follow the usual steps for training a neural network, for each epoch, we iterate over the batches and do the forward pass, loss computation and backward pass for each batch. At the end of each epoch, we use the validation set to monitor the performance of our model based on new, unseen data.
4. **Evaluating the Model**
   After we finish training the model, we then use the test dataset to evaluate the model's performance based on the accuracy of the results on complete new and unseen data. We use a

similar step as training to iterate over the batches, get the output of each batch, and calculate the average accuracy for all the batches.

5. **Hyperparameter Optimization**
   To further enhance the performance of the fine-tuned BERT model, we perform hyperparameter optimization. For now, we will only be testing different learning rates since each model takes quite long to train and we do not have the resources or enough time to use technique such as grid search to find the best combination of hyperparameters over all the different hyperparameter that we have. So we chose to only tune learning rate, which is the most significant one.

After going through this process, we now have the best hyperparameters, a good model architecture, and all the training and evaluation steps step up. Now we can train a final model with more epochs to achieve a even higher accuracy and performance. The final model can now be used to predict sentiment labels for any movie review.

Challenges and/or Improvements

In this section, we discuss the challenges encountered during the fine-tuning of the BERT model on the IMDb dataset for sentiment analysis and propose potential improvements to address these issues.

- **Computational Resources**
  One of the main challenge that we came across is not powerful enough computational resources. The BERT model has a large number of parameters, which lead to high computational resource requirements, particularly in terms of memory and processing power. During the process of fine-tuning our model, we found that running out training loop required at least 23 GB of GPU RAM over a really long period of time.

  We first tried just using the normal version of Google Colab to run this, but obviously, the weak GPU did not have enough memory. So to solve this, we decided to purchase Google Colab Pro which offered double the GPU memory of the original one and also premium computing units made for high-RAM usage. This allowed us to run the fine-tuning program without it running out of memory and crashing.
- **Hyperparameter Optimization**
  Another major challenge we faced during this project was the time-consuming nature of the hyperparameter optimization process. As previously mentioned, finding the optimal combination of hyperparameters is crucial for maximizing the model's performance. However, tuning hyperparameters require us to train multiple test models which is extremely time consuming even with powerful GPUs from Google Colab Pro. In particular, training our model with 3 epochs took an average of 40 minutes to an hour because of the vast amount of data in the dataset and the huge parameter count that BERT has.
  Since the process of hyperparameter tuning often involves adjusting multiple hyperparameters, such as learning rate, batch size, number of epochs, it is clear that trying to force search the huge space for the best hyperparameter is unrealistic. Instead, we try to take suggestions from

the paper for things like batch size, learning rates, and the number of epochs. By doing that, we reduce the search space to an manageable amount that we can manually test.

- **Dataset Management**
  Another challenge that we faced was the inefficient organization and management of the dataset. Initially, we did not utilize any of the dataset handling tools provided by PyTorch, so the data management process became messy and cumbersome. This led to difficulties in efficiently loading, processing, and feeding the data into the model during training and testing.

  Later on, we found out about the Dataset and Dataloader classes provided by PyTorch. hese classes provide a clean and efficient way to manage the dataset, allowing for easier data loading, preprocessing, and batching. We used the Dataset class in PyTorch to create custom dataset objects that allows for easy access to individual data samples and their corresponding labels as well as more efficient tokenization of our data. The DataLoader class, on the other hand, is used to efficiently handle data batching, shuffling, and parallel data loading during the training and testing process. This is especially useful when it came to the training and testing part, since it automatically handling the creation of mini-batches, shuffling the data, and distributing the data loading process across multiple worker threads. This not only simplifies the data management process but also leads to performance improvements due to parallelized data loading.

Results

In this section, we present the results obtained from fine-tuning our BERT model on the IMDb dataset for sentiment analysis. We evaluate the performance of our final model and compute the validation as well as test accuracy, and we will also test it on some real data.

During the hyperparameter tuning process, we found that our models starts overfitting and the accuracy drops significantly at around 5 epochs, so at the end we decided to train our model with only 4 epochs, while that quite a low number, with the amount of complexity the original BERT model has, we are still able to get quite a high accuracy.

After training our final model, here are some of our metrics:

- Train Accuracy: 94.05%
- Validation Accuracy: 91.74%
- Test Accuracy: 91%

All of these are tested at the end of the last epoch on the corresponding datasets.

In conclusion, the results obtained from our experiments showcase the strong performance of the fine-tuned BERT model for sentiment analysis on the IMDb dataset.

Discussion and Conclusions

In conclusion, this study demonstrates the effectiveness of the BERT model for sentiment analysis on the IMDb dataset. By fine-tuning the model and optimizing its hyperparameters, we were able to achieve high performance, surpassing (or being competitive with) existing benchmarks in the literature. Our findings indicate that the BERT model, with its bidirectional pre-training and contextualized representations, is a powerful tool for sentiment analysis tasks and can be successfully adapted to various NLP challenges. The insights gained from this study can guide future research in fine-tuning the BERT model for other NLP tasks or on different datasets. Moreover, the proposed improvements and strategies can be employed to overcome challenges and enhance the performance and efficiency of similar models in the field of NLP. Overall, our study contributes to the growing body of research on the BERT model and its applications, further establishing its potential as a robust solution for a wide range of NLP tasks.

References

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). *Bert: Pre-training of deep bidirectional Transformers for language understanding*. arXiv.org. Retrieved May 5, 2023, from https://arxiv.org/abs/1810.04805

Bert. BERT. (n.d.). Retrieved May 5, 2023, from https://huggingface.co/docs/transformers/model_doc/bert

N, L. (2019, March 9). *IMDB dataset of 50K movie reviews*. Kaggle. Retrieved May 5, 2023, from https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

A Visual Notebook to Using BERT for the First Time. Colab Notebook. (n.d.). Retrieved May 5, 2023, from https://colab.research.google.com/github/jalammar/jalammar.github.io/blob/master/notebooks/bert/A_Visual_Notebook_to_Using_BERT_for_the_First_Time.ipynb#scrollTo=lZDBMn3wiSX6