Project I Report

Team members: Yena Kim , Krish Jhaveri , Tianna Calderon , Christopher Castrence , Adrian Balingit

Problem Statement

Conventional language models have a basic flaw in their comprehension of textual context. The majority of earlier methods exclusively process language in one direction: either from right to left, as in some recurrent networks, or from left to right, as in autoregressive models. When predicting or interpreting a word, these models can only employ partial context due to this directional limitation, depending only on what comes before (or after) it. Models are unable to completely understand how words interact and influence one another throughout a phrase because of its unidirectional design. As a result, they have trouble with language comprehension tasks like question responding, inference, and dialogue comprehension that call for deeper semantic reasoning. In order to overcome this difficulty, the BERT study presents a technique for pre-training deep bidirectional language representations that can concurrently learn from previous and succeeding words.

Motivation

Although a number of methods, including ELMo and OpenAI GPT, had made progress in the field of natural language processing before BERT, they were still plagued by intrinsic structural and contextual constraints. Because each new activity required specific modifications, this decreased their scalability and transferability. Existing methods could not fully model bidirectional context or generalize across tasks.

The goal of BERT was to develop a single, all-purpose language model that could directly learn strong contextual representations from vast amounts of unlabeled text. The same model may readily adapt to a wide range of NLP tasks with little fine-tuning thanks to this dual pre-training method. In the end, the

authors were driven by the desire to close the gap between general language comprehension and language representation learning.

Related work

In the past, natural language processing techniques concentrated on creating word representations that conveyed meaning separately, as opposed to in context. Regardless of how a word was used in various contexts, models like Word2Vec and GloVe generated static embeddings in which every word had a single fixed vector. This hindered their comprehension of complex sentence relationships and words with various meanings. Later, ELMo used bidirectional LSTMs to create contextual embeddings, which increased context sensitivity but still handled text sequentially and had trouble with long-range dependencies. In order to overcome these problems, Vaswani et al. (2017) introduced the Transformer architecture, which uses self-attention techniques to enable models to simultaneously record dependencies between all words in a sequence.

Later, OpenAI's GPT adopted a similar format, but its comprehension of complete context was limited because it could only read text from left to right. By using a bidirectional Transformer encoder that can read text in both directions simultaneously, BERT enhanced these concepts and allowed for a far deeper understanding of language. In contrast to earlier models, BERT was able to learn context-rich representations that could be applied to a variety of downstream tasks without requiring significant task-specific modifications because of its pretraining objectives, Masked Language Modeling and Next Sentence Prediction.

Dataset

The GLUE Recognizing Textual Entailment (RTE) benchmark, which evaluates a model's capacity to ascertain if one sentence logically follows from another, is the dataset utilized in our effort. Predicting whether the hypothesis is implied by the premise is the task for each sample in the dataset,

which consists of two sentences: a premise and a hypothesis. The TensorFlow Hub BERT preprocessing model was used to preprocess the dataset after it was loaded using TensorFlow Datasets (TFDS). BERT's WordPiece tokenizer, which divides words into subword units to handle rare terms and ensures compatibility with BERT's vocabulary, was used to tokenize the text during preprocessing. Special tokens were used to integrate each pair of sentences into a single sequence: [CLS] was added at the start to indicate the overall meaning for classification, and [SEP] divided the two input sentences. These tokenized sequences were transformed into input IDs, attention masks, and segment IDs by padding or truncating them to a constant length of 128 tokens. Compatibility with BERT's design, which needs consistent input forms for effective processing, was guaranteed by this formatting. The dataset was batched and sent into the model for fine-tuning after being preprocessed. A classification layer used the final concealed state of the [CLS] token to determine whether the first sentence implied the second. The RTE dataset successfully used BERT's bidirectional language understanding to carry out textual entailment classification through this data preparation and integration.

Model/algorithm and methods

BERT is based on a multi-layer bidirectional Transformer encoder, which reads text both left-to-right and right-to-left to better understand context. Before processing, it splits words into smaller pieces (WordPiece) and adds information about each token’s position and which sentence it belongs to. It also uses special markers: [CLS] marks the start of the input and is useful for classification tasks, and [SEP] separates sentences. To learn language patterns, BERT is pre-trained on large amounts of text using two main objectives. The first is Masked Language Modeling (MLM), where random words are hidden and the model predicts them based on context. The second is Next Sentence Prediction (NSP), which teaches the model how sentences relate to one another. After this pre-training, BERT can be adapted to different tasks, such as question answering or text classification, by adding a small layer on top and briefly fine-tuning it. What makes BERT stand out is that it learns from context on both sides at every

layer, allowing it to understand language more deeply than models that only read in one direction. Earlier models like OpenAI GPT read strictly left-to-right, and ELMo used separate forward and backward LSTMs. In contrast, BERT models both directions together within each layer. This unified pre-training and fine-tuning approach removes the need for complex task-specific architectures and delivers strong performance across many NLP benchmarks.

Results/Discussion

The goal of our project was to replicate the findings of BERT with RTE. To do this, we trained using the RTE task from a TensorFlow example. Our batch size was 16, epochs 5, and initial learning rate at 2E-5. Our goal was to obtain an accuracy of 66.4%, similar to the paper.

Our results in our first attempt was 60.3% compared to the paper we were short by about 6%. This may be due to our system constraints, and initial parameters. We used Google Colab and while we are able to get a cloud version of TPU, this process may slow down and also drop information that is processed. On the Google Cloud website, it even mentions that TPUs do not work for high-precision arithmetic. We also had initially set dropout to 0.1, we changed it to 0.2 and that improved our 2nd attempt in accuracy to 61.4%. This means that our data was overfitting. In our third attempt, we got an accuracy of 71.12%, this was much higher than both attempts. The model most likely went back to overfitting, based on the parameters we had set for this attempt. Our last attempt had an accuracy of 68.59% , with epochs set at 3 and using only the huggingface dataset versus the previous tests using both the tensorflow and huggingface datasets.

Conclusion

This project successfully replicated BERT fine-tuning on the RTE task from GLUE benchmark. Through the use of the HuggingFace Transformers library, we achieved 68.59% validation accuracy,

matching the paper's 66.4%. This shows that the transfer learning works even when there is limited data, despite the RTE having only 2,490 training examples. We improved our Bert-Base-Uncased by implementing a data pipeline that is for sentence pairs which use tokenization. The bidirectional nature of BERT, that reads from both sides at the same time is good for understanding relationships. Our application of the process was straightforward even with a few epochs.

We encountered challenges including overfitting on a small dataset and by adjusting our dropout and being careful with epoch tuning. Small datasets like RTE have high variance, which makes results unstable with each runtime. Despite these challenges we successfully implemented the full BERT pipeline, to match the paper results, while gaining hands-on experience with transformers. Overall, this project showed both the power and limitation of transfer learning.

Description of individual effort

- Christopher helped combine multiple BERT example notebooks from Github into a single notebook with RTE testing as well as write the algorithm/methods section in the project report.
- Tianna worked on deciding which BERT examples to implement, implementing testing of RTE Tasks, and writing results/discussion.
- Krish worked on Related Work and Dataset sections that describe how the RTE dataset was processed and integrated with the BERT architecture, as well as background research on earlier NLP models like Word2Vec, ELMo, and GPT.
- Yena worked on the Problem Statement and Motivation sections, explaining the existing limitations and main problem addressed in the paper, and also reviewed the report toward the end.
- Adrian worked on testing the model by creating the predication function which tests new examples and shows confidence scores. He also helped fix issues during training and tried different settings to get better results.

References

Google Cloud. (n.d.). Introduction to Cloud TPUs. Google Cloud. https://cloud.google.com/tpu/docs/intro-to-tpu

TensorFlow. (2024, March 23). *Fine-tuning a BERT model*. TensorFlow. https://www.tensorflow.org/tfmodels/nlp/fine_tune_bert#build_train_and_export_the_model

TensorFlow. (2023, May 27). *Solve GLUE tasks using BERT on TPU*. TensorFlow. https://www.tensorflow.org/text/tutorials/bert_glue