Grayson Byrd
HW2 Report
Github: https://github.com/gbbyrd

Project Description:

The purpose of this project is to gain a better understanding of the transformer architecture through fine-tuning BERT, a pre-trained transformer model that can be fine-tuned for a variety of different natural language modeling tasks. For this project, the question answering application was chosen. Two datasets were used for this work: Spoken-SQuAD and SQuAD. The Spoken-SQuAD dataset is a speech-to-text version of the SQuAD dataset. The Spoken-SQuAD presents many challenges due to the errors in the dataset associated with the faults of the speech-to-text technology. In this work, a pre-trained BERT model is finetuned on both dataset and the performance of each model is compared and discussed.

Technology Review

There are many techniques and technologies that have been developed to improve the performance of fine-tuned transformer models. In this section, I will review the basic technology and principles surrounding the transformer model and will discuss various open source libraries for implementing a fine-tuned BERT model.

Architecture

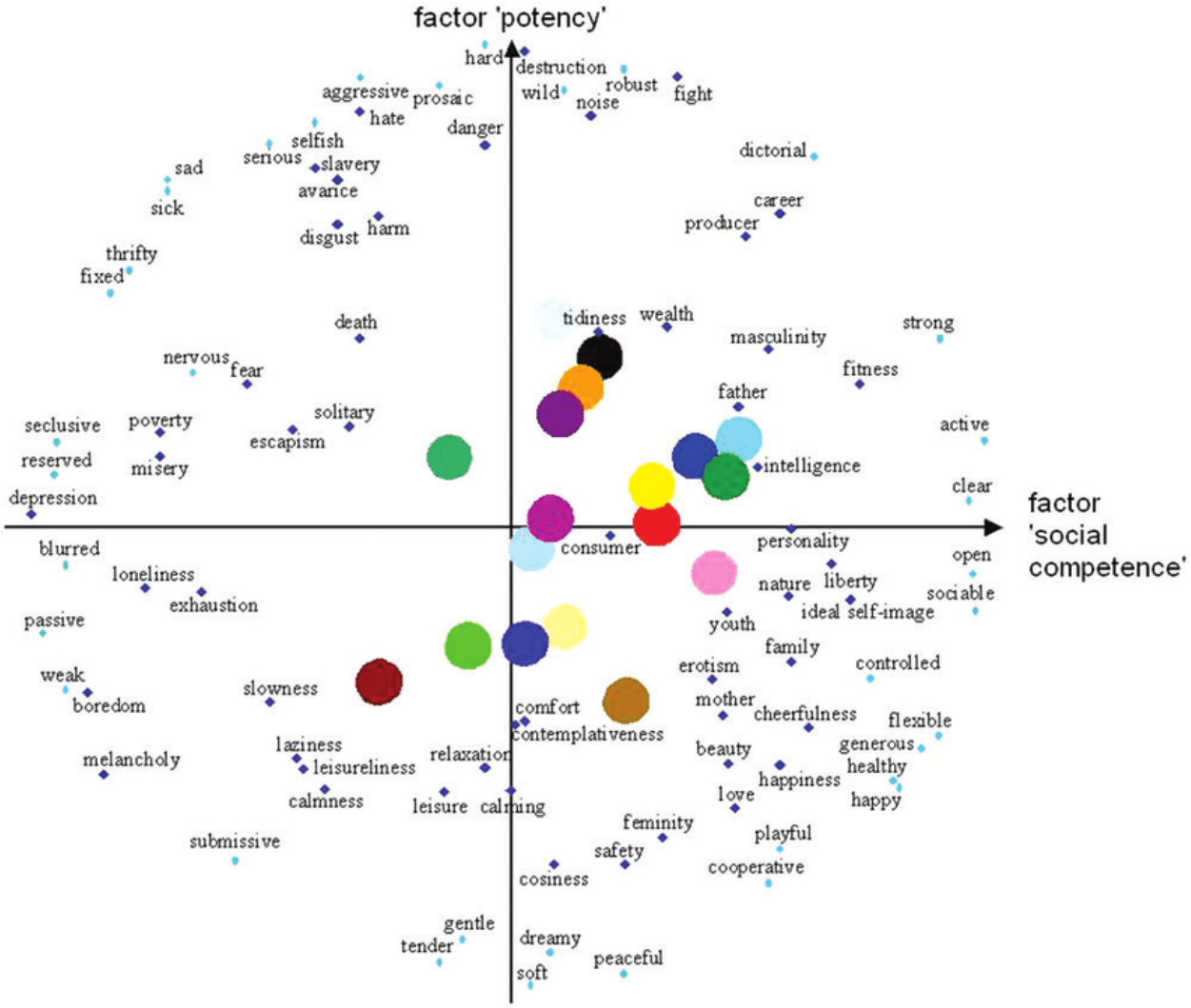
Transformer

Top technology company, NVIDIA, defines the transformer model as “a neural network that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence.” To begin to understand how transformers work, we must first understand a basic principle of the transformer architecture: Tokenization.

Tokenization

Tokenization is the act of “embedding” words into a vector. There are many fascinating tricks that can be used to embed not only information about the letters in the words but also context and semantic information of the words. These tricks effectively give the transformer model a more sophisticated understanding of the sentences that are being input into it. Figure 1 below depicts how this is done.

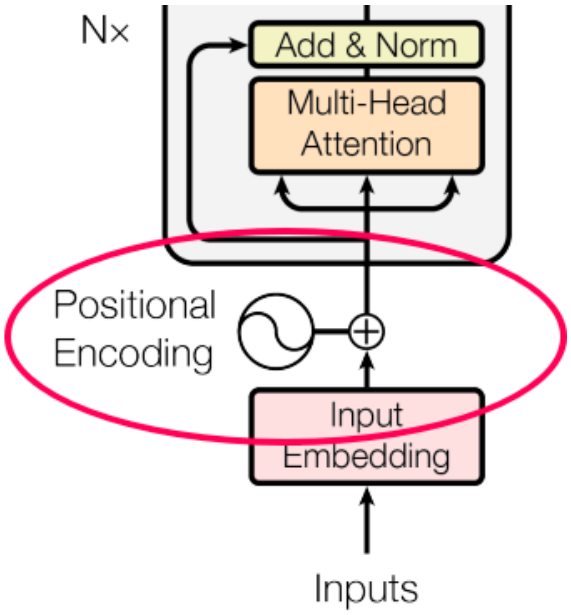
Figure 1: A representation of semantic tokenization of words



Words that are more closely related to each other, father and masculinity for example, are located close together in the embedded vector space.

In a transformer, this semantic encoding of the word in the sentence is combined with a positional encoding of each word in the sentence, effectively giving the input sentence situational understanding of the words being input into it by taking into account the basic meaning of the word and the sequential structure of the sentence.

Figure 2: Positional encodings are added to the semantic word encodings to provide an understanding of the words that is dependent on the sentence structure

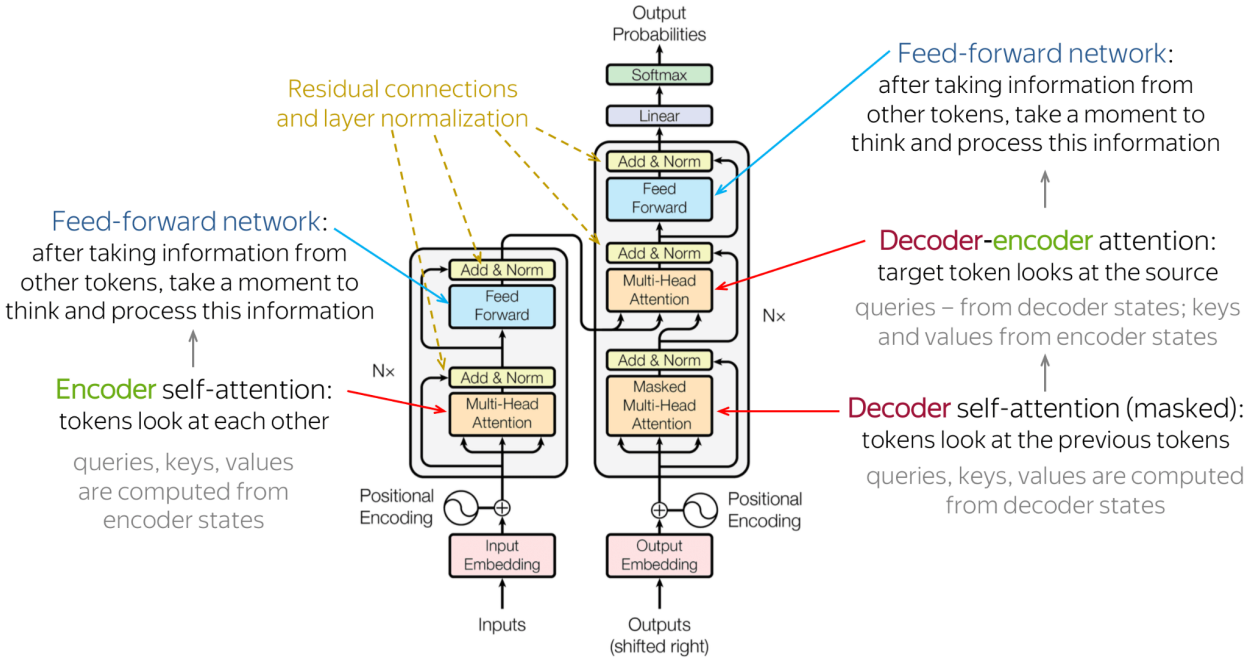


BERT uses a special kind of tokenization called WordPiece tokenization. More can be read about this technique here:
https://ai.googleblog.com/2021/12/a-fast-wordpiece-tokenization-system.html

Attention

The transformer model uses a neural network with self attention that tasks as inputs the contextualized word embeddings. Since attention was discussed in my previous report, I will not go into detail here, but here is a link to the original attention paper:
https://arxiv.org/abs/1706.03762

Figure 5: Full architecture of the transformer model



Now that the transformer architecture has been discussed, we can begin to look at the innovations of the BERT architecture.

BERT

BERT stands for Bidirectional Encoder Representations from Transformers and is "designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. (Devlin et. al)" BERT's architecture can be described as a multi-layer bidirectional transformer encoder. BERT has two standard model sizes, BERT-base (110M parameters) and BERT-large (340M parameters). In this work, BERT-base was used. BERT was pre-trained using two unsupervised tasks: Masked LM and Next Sentence Prediction. In order to fine tune the model for its various NLP applications, a new head must be added to BERT.

BERT For Question Answering

To fine tune for question answering, the corresponding head was added to the BERT model using the BertForQuestionAnswering model from the huggingface transformers library. This head outputs the start and end probabilities of each word in the input sentence (excluding the question portion of the input). The answer to the question would then be the sequence within the context portion of the input sentence that starts with the word corresponding to the greatest start probability and ends with the word corresponding to the greatest end probability. In order to improve the performance of my models, the following techniques were used.

Doc Stride

Doc stride was used to reduce the training time and computational power required to train the model. As the self attention algorithm in a transformer has O(n^2) time complexity, the computation time required to train the model increases exponentially with an increase in the size of the input to the model. To mitigate this problem, large input sequences were split into multiple smaller sequences or “windows”. To get the best answer, the maximum start and end probabilities for each window are found. The total score for the sequence in each window is then the sum of the start and end scores. Whichever window has the greatest score (which can be seen as whichever window is most confident) will be chosen.

Linear Learning Rate Decay

A linear learning rate was used to improve the training of the model. This learning rate starts off at a peak and then slowly and linearly decreases to zero throughout the course of the training. That means that the model will change significantly more at the start of the training process than at the end. This was implemented using the huggingface ‘scheduler’ function.

Preprocessing

To improve the performance of the model, effective preprocessing of both datasets used in this work was done. This preprocessing included dividing large input sentences into multiple windows and ensuring that a truncated answer (one that starts in one window and ends in another) was dealt with appropriately. This was done by setting the start and end labels to (0, 0) for any example where the answer was not completely inside that specific window.

Accelerate

The huggingface accelerate function was used to improve the speed at which my model was trained. This function allowed me to use multiple gpus concurrently during the evaluation portion of my project.

Multiple Pretrained Models

For this work I trained using bert-base-uncased and distilbert-base-uncased. The performances of each pretrained model was also compared. Distilbert is a “distilled” version of the BERT architecture. Here is a brief description from the huggingface documentation: “DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than bert-base-uncased, runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark.”

Evaluation

The evaluation of each model was done using an “exact match” percentage as well as an average f1 score for each model across all testing data. The f1 score was implemented using the huggingface evaluate function. For a better understanding of how the f1 score is calculated for nlp tasks, refer to the referenced article:
https://kierszbaumsamuel.medium.com/f1-score-in-nlp-span-based-qa-task-5b115a5e7d41
To get a fair comparison, the same testing dataset was used for all models. The standard SQuAD dataset was chosen as I wanted to see how a model with errors in the training dataset (SpokenSQuAD) would perform on a testset with no errors in it.

Results

Below are the exact match and f1 results for each of the four models.

Training Dataset	Testing Dataset	Pre-trained Model	F1 Score	Exact Match Score
SQuAD	SQuAD	bert-base-uncased	88.27	80.76
Spoken SQuAD	SQuAD	bert-base-uncased	79.17	68.96
SQuAD	SQuAD	distilbert-base-uncased	4.38	8.82
Spoken SQuAD	SQuAD	distilbert-base-uncased	3.10	8.01

As you can see, the SQuAD trained, bert-base-uncased model performed the best with an F1 score of 88.27 and an exact match score of 80.76. This makes sense, as this model was trained on the most reliable dataset and has the most parameters of all of the models. The second best model was the Spoken SQuAD trained, bert-base-uncased model. This came as a surprise as I expected the SQuAD trained, distilbert-base-uncased model to be the second best.

Both of the distilbert models performed very poorly. There may be an error here, but more analysis is needed. The only difference between the training codes is the string passed into the .from_pretrained function from the huggingface transformers library, so I am not sure if there is an error or if distilbert is just this much worse than the standard BERT architecture.

References:

"Accelerate." *Hugging Face – The AI Community Building the Future.*, https://huggingface.co/docs/accelerate/index.

Devlin, Jacob, et al. "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding." *ArXiv.org*, 24 May 2019, https://arxiv.org/abs/1810.04805.

Li, Chia-Hsuan, et al. "Spoken Squad: A Study of Mitigating the Impact of Speech Recognition Errors on Listening Comprehension." *ArXiv.org*, 1 Apr. 2018, https://arxiv.org/abs/1804.00320.

Optimization, https://huggingface.co/docs/transformers/main_classes/optimizer_schedules#transformers.get_linear_schedule_with_warmup.

Song, Xinying, and Denny Zhou. “A Fast Wordpiece Tokenization System.” – *Google AI Blog*, 10 Dec. 2021, https://ai.googleblog.com/2021/12/a-fast-wordpiece-tokenization-system.html.

Merritt, Rick. “What Is a Transformer Model?” *NVIDIA Blog*, 16 Sept. 2022, https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/.

Vaswani, Ashish, et al. “Attention Is All You Need.” *ArXiv.org*, 6 Dec. 2017, https://arxiv.org/abs/1706.03762.

Samuel Kierszbaum, PhD. “F1 Score in NLP Span-Based QA Task.” *Medium*, Medium, 9 June 2021, https://kierszbaumsamuel.medium.com/f1-score-in-nlp-span-based-qa-task-5b115a5e7d41.