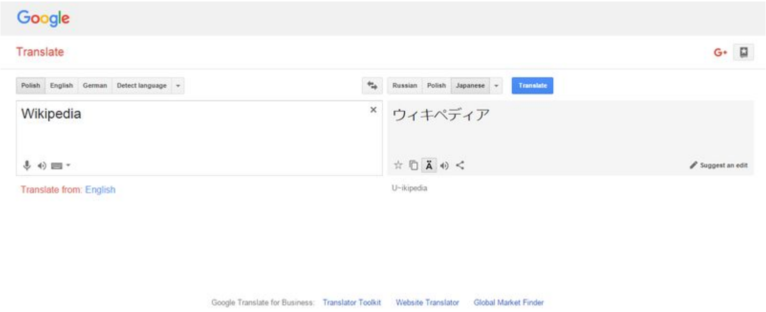
A Comparison of Two NLP Frameworks for General Research Purposes

Presentation by Edward Fisher Jr, Vincent Strzelecki, and Connor Munnis

What is NLP?

NLP stands for Natural Language Processing. This is a subsection of artificial intelligence that works to understand human language. It relies on machine learning to be able to read words and work to understand the context of a phrase or sentence. The most common uses are spell-check and grammar check programs, translation programs, and virtual assistants like Siri and Alexa.



Wikipedia

ウィキペディア

Translate from: English

U~ikipedia

Google Translate for Business: Translator Toolkit Website Translator Global Market Finder

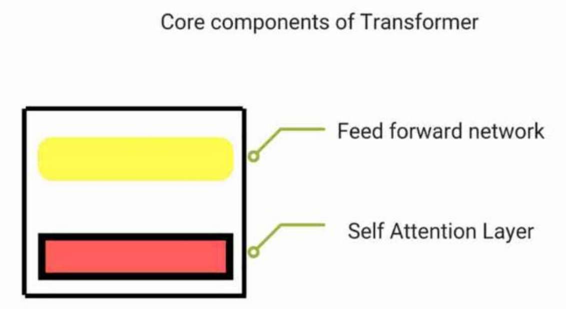
Our Project

We wanted to test two different natural language processors against each other, BERT and Transformer-XL, both created for deeper NLP tasks, and see how they perform in different tasks. We aimed to test them in three different areas:

- Question-and-answer
- Summarization
- Recognition of tokens and named entities.

Base Model: Transformers

Both BERT and Transformer-XL run off the base NLP model Transformer. The impressive part Transformer is that it has a Self Attention Layer, which computes the attention scores (measure of relevancy) of words in their relationship to a word being processed. This information goes to the other part of Transformer, the feed forward network, in the form of a matrix of weights to form a weighted representation of all the words previously encountered.



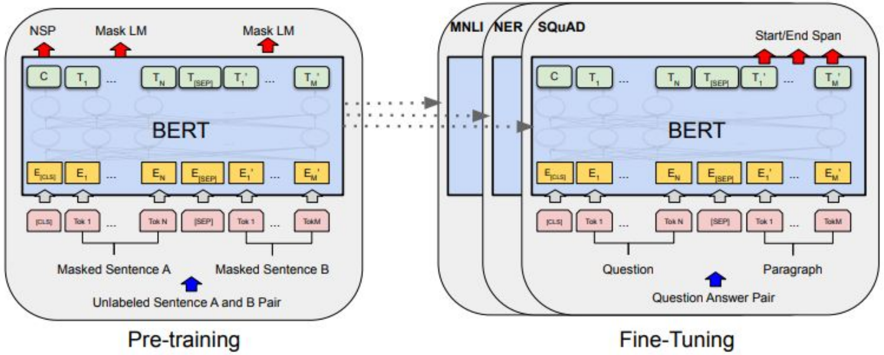
BERT

BERT: What’s different about it? Part 1

BERT stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. While vanilla Transformers process words unidirectionally, with one vector for calculating attention scores, BERT uses bidirectional encoding, meaning two vectors for calculating attention scores, using the words before and the words after said word. The increased complexity of the calculation allows for greater results in all NLP tasks, especially Q&A.

BERT: What’s different about it? Part 2

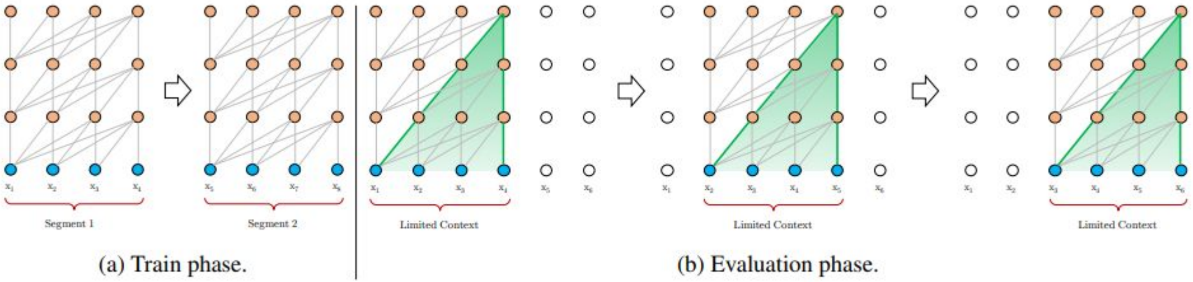
BERT stays on top of its attention scores by using a Masked Learning Model. What it does is it goes through text randomly masking some of the input words and then predicting what they are based on the surrounding words. This reinforces correct word scoring while correcting improper scores that may have been set for previous contexts.



Transformer-XL

Transformer-XL: What’s different about it? Part 1

The XL in Transformer-XL refers to the increased effective context size. Vanilla Transformer models split a text into independent segments to analyze for context and language modeling loss. These are normally kept separate, with no information flowing over the segment boundaries. The hidden states that are required for context don’t get passed over, and tokens at the beginning of a segment suffer a loss in optimization because of it.

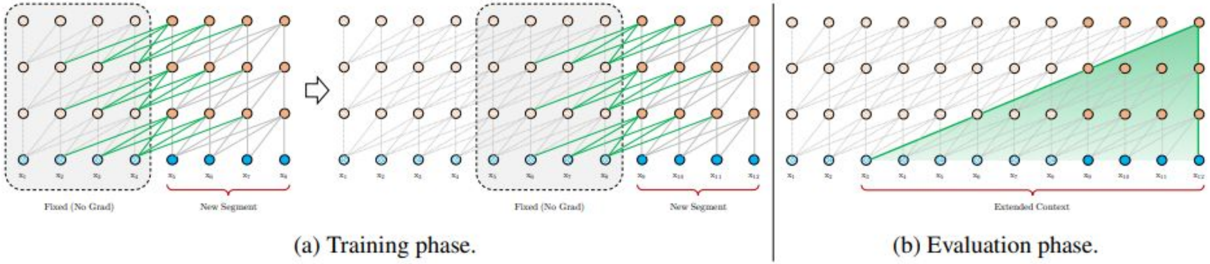


(a) Train phase.

(b) Evaluation phase.

Transformer-XL: What’s different about it? Part 2

Transformer-XL gets past the segment boundary issue by caching the hidden states from one segment and using it as fixed memory for the next segment, saving time and resources that would be used by recomputation, and also increasing the effective context length.

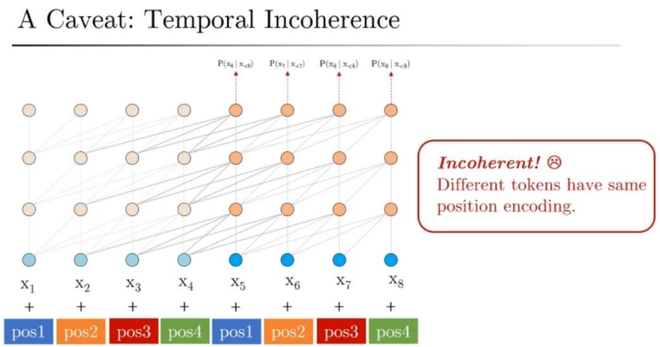

x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
Fixed (No Grad)
New Segment
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
Fixed (No Grad)
New Segment
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
Extended Context

(a) Training phase.

(b) Evaluation phase.

Transformer-XL: What’s different about it? Part 3

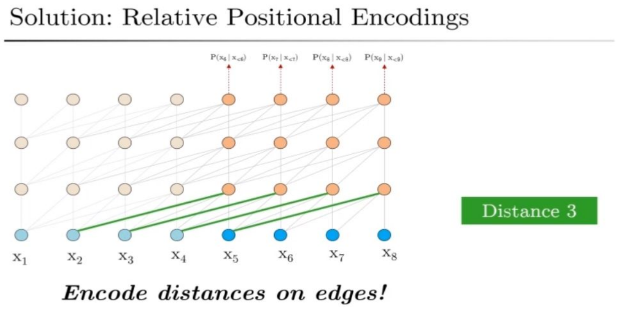
An issue that comes with a larger effective context is that sometimes multiple tokens have the same positional encodings, so what Transformer-XL does is encode the distance from other referenced tokens instead of their actual position.



A Caveat: Temporal Incoherence

Incoherent! ☹
Different tokens have same position encoding.

$P(x_6 | x_{<6})$
$P(x_7 | x_{<7})$
$P(x_8 | x_{<8})$
$P(x_9 | x_{<9})$
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
+ + + + + + + +
pos1 pos2 pos3 pos4 pos1 pos2 pos3 pos4



Solution: Relative Positional Encodings

Encode distances on edges!

Setup

All of the code is located on a server that Ed set up. He used huggingface.co/transformers to download all the dependencies and libraries for BERT and Transformer-XL. Originally the BERT model we had was pretrained for the IMDb database, once we got BERT to execute (this took forever, there were so many errors) we changed it to a Wikipedia-pretrained model. Once that was done, Vincent and Connor were given access to a section of the server that was for the testing scripts.

Setup

After that, we implemented scripts for entity recognition, scanning through a text corpus, identifying entities and tokens, and categorizing them based on what kind of word they are and how it relates to the words around them. The plan was to test the efficiency of our two scripts (entity.recognition.py for BERT and pipeline.py for Transformer-XL) and accuracy, unfortunately issues regarding our Q&A scripts had us put this on the backburner.

Setup

The QA scripts were the ones that we got working the best over the course of the project. Our BERT script, bert-qa-advanced.py, takes both the question and the context in the form of strings, maxed out at 512 characters each, and was formatted to put the results of each question into originally a spreadsheet for viewing, but later that was changed so it went to a .json file for the same ease in viewing but with increased functionality. We gathered 50 test cases to test its accuracy.

Transformer-XL didn’t come with any tools for QA due to how it was set up, so we had to research an additional framework to implement. Finding another research paper “Applying Transformer-XL to Q&A” by Sam Xu gave us insight into the model XLNet. After working with that we were able to make xlnet_qa.py

Results

BERT and Transformer-XL are not directly comparable.

Creating the scripts and doing further research, the two models have different structures and are made with different purposes in mind. BERT is specifically trained for question-and-answering, providing strong extractive results on data pools smaller than 512 tokens. Transformer-XL, with its increased effective context, is better for handling long-form data beyond 512 tokens, which is better suited for tasks such as summarizing an article or book.

Results

In addition to that, during our research we discovered that Transformer-XL's attention module can be combined with other models to increase the size of the data pool they can process. Also the context handling of Transformer-XL can be applied to other models to improve processes like question-and-answer tasks. What we used for the QA section of our scripts was a combination of the model QANet with Transformer-XL's benefits, this is known as XLNet.

Results

In our research we were able to use XLNet to process QA for questions with much larger data pools than 512 characters. This ability to process long-form data allows for Natural Language Processing to be used for tasks where data cannot be converted to 512 character data-frames. In experiments completed by Dian Ang Yap on the SQuAD2.0 dev data set XLNet was able to achieve an 87.9 compared to BERT's score of 78.89. BERT requires more fine-tuning using specific datasets to achieve the higher results it's capable of.

Conclusion

Since our original goal was to compare only Transformer-XL and BERT for general research purposes, not XLNet, BERT is the best model where large data pools are not necessary or beneficial due to the flexibility and variety of problems it is well suited at answering. It has better ease-of-use, a larger development community as it was created solely by Google employees, better support, more pre-trained models, and an easier fine-tuning process.