A Comparison of Two NLP Frameworks for General Research Purposes

Edward Fisher Jr., Vincent Strzelecki, Connor Munnis
{efisher, vstrzelecki, cmunnis}@qmail.qcc.edu

Abstract

Our project will examine two powerful Natural Language Processing algorithms, BERT and Transformer-XL, in their abilities to extract and summarize data from chosen pieces of literature. Both have the attention model Transformer as their base. "[Transformer-XL] consists of a segment-level recurrence mechanism and a novel positional encoding scheme" (Dai, et al. 2019), meaning it takes segments of data and not only individually analyzes each segment, but also references segments against each other for increased accuracy regarding context. BERT focuses on working around the Transformer constraint of unidirectionality, where context is analyzed in only one direction, leaving room for error when the context from the other direction is needed. The strategy for its bidirectionality is "using a 'masked language model' (MLM) pre-training objective," which "randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word using only the [left and right] context" (Devlin, et al. 2019). We will provide each algorithm with the same dataset and judge the results for each algorithm on its accuracy compared to its execution time.

1 Introduction

The goal of Natural Language Processing (NLP) is to train computers to analyze human language. The widest-used versions of NLP are used in spell-check and grammar-check programs, but more advanced versions have been developed into tools used for much more than just identifying context within search queries. NLP is becoming increasingly more useful for researchers to summarize large amounts of data or long-form documents without the need for human supervision.

Our research project was more about learning than anything else. None of us had any experience with machine learning, much less Natural Language Processing. We wanted to learn about how NLP algorithms worked, learn how to access and use them, learn how to implement them using a specific database, and run different types of test cases for each to see how each algorithm performed against the other. We chose BERT and Transformer-XL as our two algorithms, both were created by Google employees and both use the base attention model Transformer. The learning curve was steep, as this is an advanced area of study. Luckily both Ed and Vincent had enough coding experience to strengthen their grasps on the concepts, whereas Connor had a lot of reading and video-watching to do to catch up. The goal was to test the two languages in three areas: question and answering; summarization; and recognition of tokens and named entities. Then we would use data from those tests to compare them.

2 Background on Transformer

Both BERT and Transformer-XL utilize a similar NLP model referred to as Transformer. A Transformer model consists of a self attention layer and a forward feed model. The self attention layer computes the attention scores of words in their relationship to the current word being processed. These calculations form a matrix that feeds the forward network with weights to form a weighted representation of all the words previously encountered.

While many other model types (convolutional, recurrent, etc) utilize an attention layer, Transformers receive their name, and their high success rate through their use of "Self Attention". This Self Attention layer allows for Transformers to accomplish higher complexity per layer as well as an increase in computational performance due to the models ability to perform parallelized computations.

The main problem Transformers were developed to solve was the problem of sequence transduction. Sequence Transduction requires models to have some sort of memory, this is especially true for language translation. In most languages, the meaning or context of a word is based on the words around it, or on the context of how the sentence is being presented. Without memory, a model has no way to retain these contexts over long sentences or multiple sentences, the context of each sentence can be retained through abstractions included in the RNN (Recurrent Neural Network) and CNN (Convolutional Neural Network) architecture. RNN allows the meaning of the word to be calculated against its neighbors through calculating permutations via a loop through the network. Prior to transformers the RNN architecture was the best possible solution available to data scientists due to its ability to send information learned to the next step in the RNN chain. However, there is a direct correlation in information loss and chain length. Therefore, in long sentences where the meaning of the last words are reliant on the context of the beginning of the sentence (masculine feminine conjugation) the RNN model would predictably generate unreliable results.

Transformers improve on RNN's and the LSTM structure by utilizing attention to retain critical information to the model. In addition to attention Transformers also employ hidden states that are unallowed to be changed by the RNN structure over the course of the models operation, allowing select data to be encoded 'as is' and decoded 'as is' over the course of the models operation. These hidden states are created by the attention model within Transformers, and later these scores of the hidden state are 'soft maxed' to produce the likely translations or outputs based on the scoring of the individual input information.

The general Transformer architecture is incredibly powerful and has been utilized to build several of the most popular models. These models include both BERT and Transformer-XL, as well as OpenAI's GPT/GPT-2, XLNet, XLM, RoBERTa, DistilBERT, CTRL, CamemBERT, ALBERT, T5, XLM-RoBERTa, FlauBERT, BART, DialoGPT and Reformer.

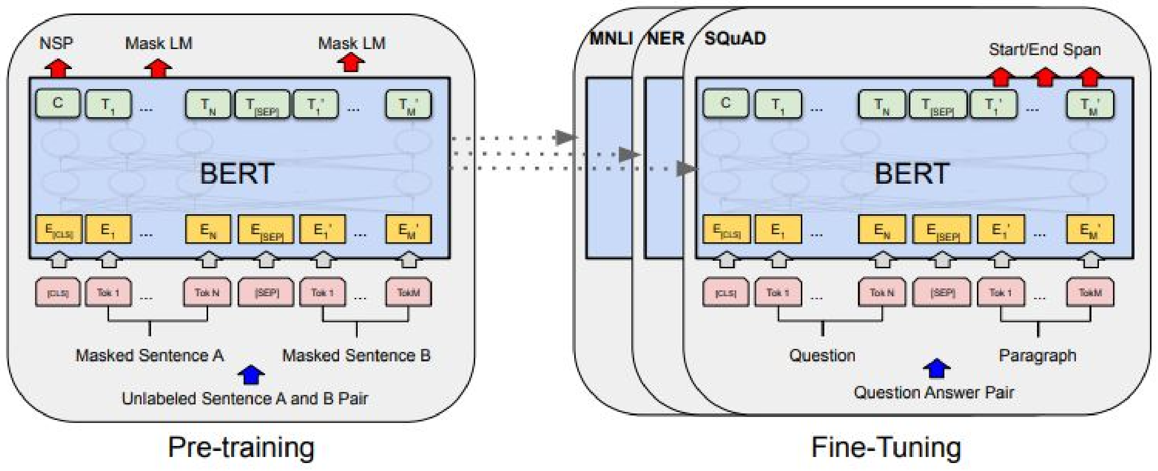
3 Background on the Algorithms

BERT

BERT stands for Bidirectional Encoder Representations from Transformers (Devlin, et al. 2019). With existing models, the model was constrained in that they were unidirectional, where the attention scores of each word was based on the score of the words that appear before that specific word. BERT allows for Bidirectional Encoding, which means that the attention scores of each word are based on both the words before and after the word currently being analyzed. BERT's ability to use surrounding words means that rather than attaching a single vector to each word, BERT can attach multiple vectors to each word based on complex relationships.

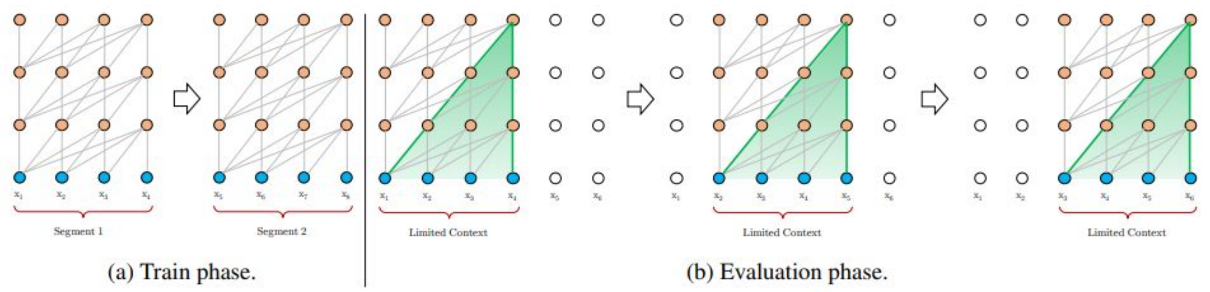
BERT controls its word vectors through its continuous testing process. By randomly masking words in a sentence and predicting them, BERT is constantly reinforcing its correct word scoring and is correcting improper scores that may have been set for previous word contexts. This also allows BERT to utilize the context of the surrounding words in it's assessment of word attention scoring rather than utilizing most common following token algorithms that many unidirectional models are based on.

While the main innovation made by BERT is it's bidirectional attention scoring, there are several other key improvements made by the BERT framework. Among these include positional embedding layers, which provide meta-data on individual tokens, allowing BERT to track which order tokens appear in, a piece of data that previously wasn't available to most Transformer based NLP's. In addition to the positional embedding layer is the segment embedding layer, which handles word meta-data related to sentence structure. This handling of segment tokens is what allows BERT to handle sentence pairs as input for its question answering process. BERT's training also includes Next Sentence Prediction, which is used to pretrain the model for tasks that require an understanding of the relationship between two sentences. In this task BERT is given two related sentences, half the time the second sentence is replaced with an unrelated random sentence. BERT then predicts whether or not the second sentence is related.



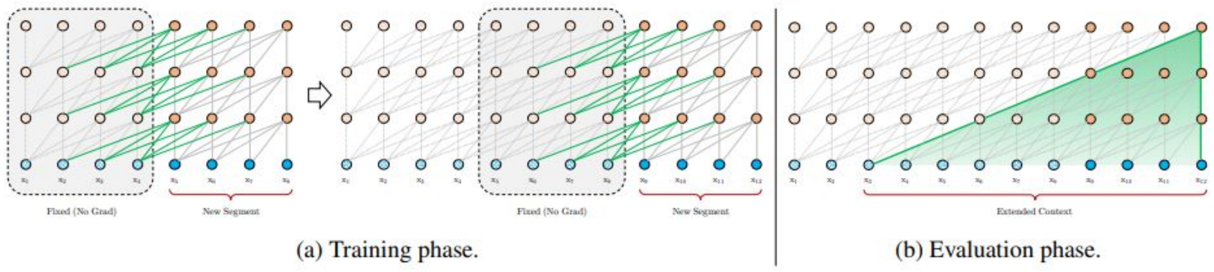
Transformer-XL

The constraint that Transformer-XL worked to eliminate is the segment barrier. Vanilla Transformer language models split a text corpus into independent segments to analyze for context and language modeling loss. Normally, text segments are kept separate, no information flowing across the segment boundaries. That leads to tokens at the beginning of each segment not having sufficient context for proper optimization. Additionally, hidden states need to be recomputated for each segment, which is a costly and time-consuming process (ACL, 00:03:25 - 00:03:49).



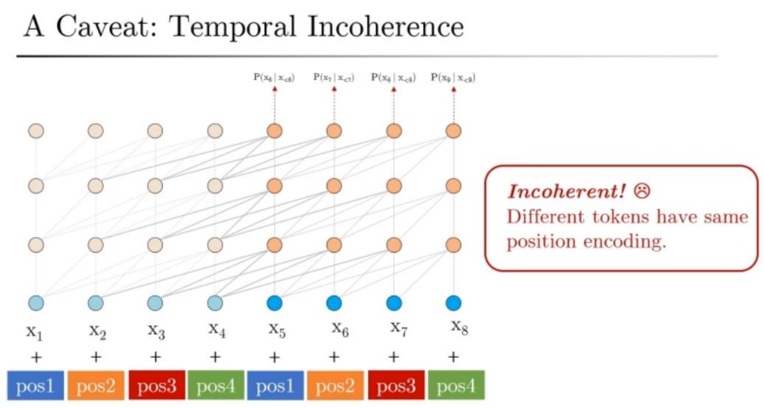
(Dai, et al. page 3)

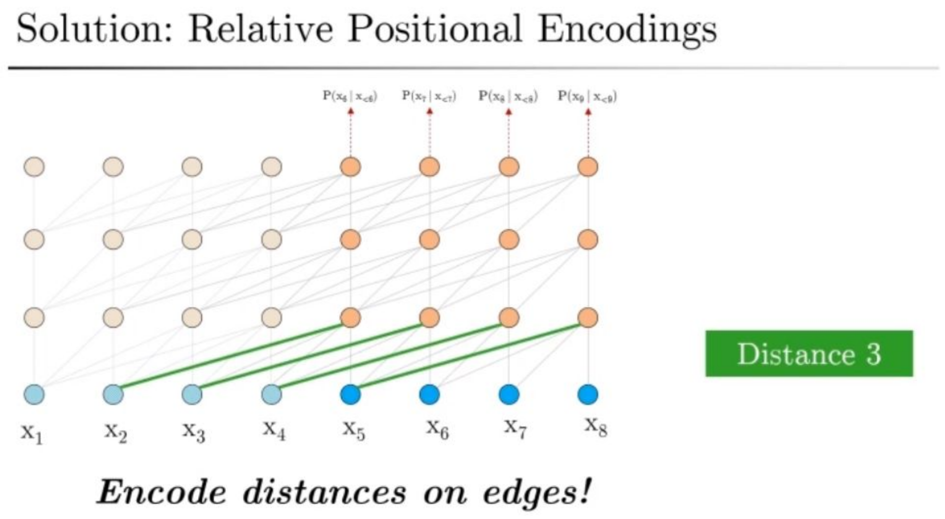
Transformer-XL (the XL stands for Extra Large, referring to segment size) works to accomplish two goals. The first goal it accomplishes is it caches and reuses hidden states from previous segment computation, to apply to the next segment as fixed memory. This increases the effective context length, and also saves time and resources as the hidden state of the next segment doesn't need to be recomputated, only fine-tuned.



(Dai, et al. page 4)

The second goal is regarding temporal information coherence. An issue with larger effective context length is in regards to the positional encoding, what happens is multiple tokens end up with the same position encodings. Transformer-XL takes care of this by encoding distance from other referenced tokens instead of their actual position. This gets rid of any ambiguity regarding which segment a token comes from, and it generalizes even longer sequences.


A Caveat: Temporal Incoherence
Incoherent! ☹
Different tokens have same position encoding.
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
Distance 3
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
Encode distances on edges!

4 Setup

The first part of the setup was getting a platform to run the two frameworks. Ed set up a Tensorflow server using his company's hardware, and gave access to Vincent and Connor for modifying scripts in a certain section. He focused on getting BERT to work, which included getting the dependencies needed, getting the right server configurations, reading up on BERT documentation and developing its structure and basic behavior. The implementations for both models came from huggingface.co/transformers. It was originally given the dataset of the IMDb database, once the structure of BERT was implemented and we could demonstrate it executing and working, we decided to change the dataset to the Wikipedia Database. The Tensorflow BERT program was changed to operate on Tensorflow 2.0 to improve functionality and allow use of Keras, the Python Deep Learning library. From there we all worked on getting testing scripts created for the BERT tests we wanted to run. While scripts were created to run tests for BERT, Ed also installed Transformer-XL onto the server (this turned out to be a significantly easier task than it was for BERT) and started writing scripts for building, fine tuning, and testing our Transformer-XL model with the same Wikipedia dataset.

We had an extraordinary learning curve to overcome just in getting the first few scripts working on the server. There are key pieces of information that need to be understood before working in data science. The first of those being that a model is a combination of nodes that utilize weights stored in checkpoints to produce outputs. Once an input is given to the model, there is very little that can be done to influence how the model will create the output from the input. Because of this, all of our influence on the models are focused in editing the configuration scripts that generate the two models, editing the environment (Pytorch or Tensorflow) that the model operates in, or editing the process used to train or evaluate that model. This removal of direct control to edit was a big stumbling point in our development. However, once we understood this implementing new features became very easy.

The first scripts to be implemented were the entity/token recognition scripts. There was entity-recognition.py for the BERT model and pipeline.py for Transformer. This reads through a prepared corpus to identify entities that are contained within said script and categorize them. The goal for those was to compare the efficiency of the two models against each other. We would then increase the size of the corpus to see how each model would respond to the increased load. This got put at a reduced priority in favor of implementing our QA scripts. Entity-recognition.py used a custom low level handling of BERT's configuration file while pipeline used a high-level huggingface API to handle working with Transformer-XL.

The next script was bert-qa-advanced.py, it was written to take the input question and 512-max-word corpus as parameters and return the answer to the question. Our original plan was to have the script read from a spreadsheet, and increase the corpus size beyond 512 tokens, but after some research into others' attempts we decided it wasn't worth the additional effort, and left the inputs inside the script. Also at this time, we decided to switch our base libraries again from Tensorflow 2.0 to Pytorch.

Pytorch provided us with the SimpleTransformers library which eventually became critical to our success in developing proper handling of the XLNet QA model. Another factor in our reasoning to move from Tensorflow-2 to Pytorch for our testing was due to Tensorflow using separate modules for BERT and Transformer-XL, adding another layer of complexity in our testing process. By using Pytorch, we were able to write similar code for both models and focus on reconfiguring both models to work properly within the SimpleTransformers library.

We gathered 50 test cases to plug into bert-qa-advanced.py, all taken from random Wikipedia pages, and edited the output so that the answers would go into a .json file for organized viewing. It was a couple weeks after that when we made a script to implement QA for Transformer-XL, trans_xl_QA_MODIFIED.py, and made it so that it would accept input in the same format as bert-qa-advanced.py. We had issues with that starting out, as Transformer-XL apparently didn't have any existing QA framework,

At this point in development, we began to realize that the question of 'which NLP is better', isn't exactly the right question to ask. There are many tasks that Transformer-XL has no internal processes to handle, those being summarization, text-generation, QA and next sentence prediction.

Thankfully Sam Xu gave helpful insight in their paper "Applying Transformer-XL to Q&A" that led to Ed looking into QANet, "a machine reading and question answering model ... that exclusively uses convolutions and the self-attention mechanisms based on the Transformer." Ed found a model that utilized the attention management from Transformer-XL and applied that to a pretrained wikipedia edited QA model to properly test against BERT. This model (XLNet) was then installed on a local development machine.

This model was different from others we had tested due to the fact that it did not have an existing pre-trained model available for us to utilize. Because of this, we had to develop the proper training and evaluation stages ourselves utilizing pytorch simple-transformers library. We then had to properly identify the output structure to create a similar output to what we have for BERT. In order to create a fair comparison, we also had to convert our 50 testing questions for BERT to a JSON formatted in the same way as SQuAD 2.0.

After doing this Vincent was able to edit the XLNet config files to run the training process via CUDA on his GPU. This allowed us to properly train the model using a variety of data-sets as well as giving us experience in training and evaluation models from scratch rather than utilizing solely pre-trained models.

During our training process we quickly encountered all of the problems of limited datasets. These issues ranged from incredibly low loss which lead us to conclude our custom dataset had led to overfitting, to more complex issues such as including too high of a percentage of “impossible” questions which lead to too low of a variance in probably answers for XLNet to choose between two possible answers. Because of this XLNet would instead return two possible answers as being equally probable.

After encountering continuous issues with utilizing our small dataset twe then copied training data from SQuAD and utilized that in our XLNet training process. This lowered the amount of loss we experienced and produced higher quality results than our initial testing dataset.

At this point we were finally able to test long-form questions in XLNet. These long-form questions were now possible due to XLNet inheriting the unlimited data-pool size from Transformer-XL. We found that while it was now possible to use large source data as the context for questions, XLNet’s answers to these specific questions were less than satisfactory. Our conclusions to why this must have occurred are included in our results section of this paper, but the most likely reasoning is due to SQuAD not properly training the model for long-form questions.

We attempted to add our own long-form questions to the training dataset, and reduced the percentage of short-form questions to see if an improvement in accuracy and a decrease in loss would occur and we noticed substantially less loss but the answers generated to our questions were still less than satisfactory.

5 Results

BERT and Transformer-XL are not directly comparable. Our comparison is limited by the different structures and purposes of the two models. BERT is specifically trained for QA, where it utilizes both Next Sentence Prediction and Beam Search in both pre-training and evaluation phase to provide strong extractive question answering results on data pools smaller than 512 tokens.

Transformer-XL does not provide the tools necessary to perform QA functions due to its lack of segment encoding that does not provide for the necessary differentiation between the “Question” and “Context” sentences.

Transformer-XL’s model adds Recurrence and Relative Positional Encoding to the general Transformer model, improving on the two largest disadvantages, those being Limited context-dependency and Context fragmentation. Because of this Transformer-XL is able to handle long-form data without being restricted by the limited datapool of 512 that other vanilla Transformer attention modules are limited by, like BERT.

We found during our research that choosing between BERT or Transformer-XL is not completely necessary. Transformer-XL’s attention module can be combined with other models to enhance the size of the data pool the models can process. Additionally, the benefits of context handling can be merged with certain models to improve processes like QA. Specifically, Transformer-XL can be combined with QANet to form a new model, XLNet.

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. This marks a significant advantage in XLNet’s overall comprehension and extractive question answering abilities. This is likely due to the fact that BERT’s strongest asset, it’s bidirectional tokenization and encoding, is not utilized at all in SQuaD’s training process. This levels the playing field even more, as SQuAD’s dataset also eliminates the need for strong segment encoding provided by BERT’s next sentence prediction. This is a major factor in the usefulness of BERT, as pre-training BERT for use on many popular datasets such as SQuAD, will not tune the Next Sentence Prediction or segment encoding skills. Because of this researchers will often need to use Wiki or IMDB datasets followed by fine-tuning utilizing SQuAD in order to reach the highest possible results BERT is capable of.

For general research purposes where large data-pools are not necessary or beneficial, BERT is the best model of choice due to the flexibility and variety of problems it is well suited at answering. For most purposes the character limit of QA and Entity Recognition limited at 512 tokens is plenty for simple chat applications and automated NLP tasks. Considering that larger texts can be pre-processed into 512 tokens and individually fed into the BERT model, the cases where Transformer-XL would provide better research results is in cases where the context of each question relies on a thorough understanding of the document or source text in question. For example, legal document QA would rely on Transformer-XL’s ability to retain context over large data-sources to produce answers of a higher accuracy than those that could be calculated when using bite sized data chunks with BERT. These bite sized data chunks lose context over each data frame, causing even efficient token delivery systems to break critical context each time the attention frame is moved.

There are other reasons a researcher may consider utilizing BERT over Transformer-XL as well, the largest reason behind this being the size of the development community behind BERT. During our research we discovered a separately pre-trained model for each individual NLP task that could be performed by BERT available for use by the BERT community. However, when searching for these pre-trained models for Transformer-XL and XLNet we were unable to find any pre-trained models for Transformer-XL and the base XLNet model is only available pre-trained for QA because of its basis of the QANet QA model.

Additionally, both Tensorflow and Pytorch have high level API's available to support and accelerate development with BERT. One we utilized heavily throughout our BERT testing and research was the Pipeline API. This API allowed us to rapidly prototype several different BERT NLP tasks without having to change critical configurations or go through a custom pre-training process. Transformer-XL however, only had a single pretrained model available, and because this model was trained on the wikipedia dataset this model was only properly configured for General and Named Entity Recognition. Compared to the models provided by the BERT community, development with Transformer-XL and XLNet required us to develop a more thorough understanding of the individual encoding layers used by Transformer-XL, and while the gain in accuracy is definitely significant the cost of development time and the high loss we witnessed on long-form questions allowed us to reach the conclusion that BERT is better suited for the general researcher.

There are other factors to consider when evaluating differences between BERT and Transformer-XL when considering the growing development community. Unlike other Transformer models, BERT is actively being used by Google to improve their indexing of web pages and their presentation of structured data in search results. This likely means that the already high performance of BERT in summarization due to next sentence and prediction, and context understanding gained from bi-directional encoding will be improved as the model is put to the test in generating quality QA/Summarization and Content Generation for consumers. It is unlikely that Transformer-XL or XLNet will experience similar wide-spread use unless the attention handling model is adapted for use with BERT. This would be unlikely to happen, as most use cases for NLPs are related to generated content digestible by humans and summarizing large amounts of data into digestible snippets. It isn't likely that Transformer-XL's attention modeling would improve these interactions, but it is very likely that BERT's very high number of encoding layers will create a large amount of data to be passed along the Transformer-XL, created a very resource intensive model that would likely operate very slowly. Even without the combination of the two models as BERT's input reaches closer to it's 512 token limit the time of completion begins to increase quadratically. It is hard to imagine the time of completion for our 1,000+ token tests we ran on XLNet.

Whether researchers or lawyers, whose use of NLP would require strong context loss reduction over long-form content would benefit from a combination of BERT and Transformer-XL even with this gain in resource-usage and time-to-train is hard to say. Our initial results of QA on long-form content utilizing XLNet produced results that force us to conclude that accurate QA with large data sources may require improvements to Transformer-XL's attention handling, as there is enough context loss to produce very unreliable results with large amounts of content.

There seems to be a growing development community for XLNet as well however, as this model did have several high level pytorch modules such as the simple transformers library that allowed us to set up XLNet on a local development environment. However, because of the lack of availability of pre-trained models for XLNet we did have to develop our own training and testing phases for the XLNet model. There are a few anomalies in the XLNet model that we were unable to record or recreate in our tests, but have been reported heavily throughout the community. The largest of these anomalies are related to XLNet's high accuracy in QA being reliant on padding being added to both the question and the answer in the encoding phase of the QA process. This padding is needed to reach the high accuracy in QA, in tests where padding is not provided researchers often report lower QA accuracy scores across the board.

This brought several other important pieces of data to our attention related to the state of NLP related to QA. The biggest piece of data we discovered is that the industry standard QA dataset, SQuAD (Stanford Question Answering Dataset) is not sufficient for training models designed to handle large data pools. This is due to the fact that all of the training data available in the SQuAD dataset is under 512 tokens each, in order to be usable by the average transformer model which is not capable of processing segments larger than 512 tokens at a time.

This means that despite seeing a large increase in QA accuracy over BERT when using XLNet, we still do not have an accurate estimate of how well a properly trained XLNet model could perform large data QA. The data science community would have to properly mix long and short token QA examples into the SQuAD dataset in order to be sure that models based on the Transformer-XL model properly utilize the attention handling. If this is not done, Transformer-XL will suffer from an issue very similar to the improper utilization of BERT's next sentence prediction layer caused by the SQuAD dataset. Truly, the state of data science seems limited by access to quality datasets moreso than by the state of NLP models and their capabilities. Without properly pre-processed and accurate training data that properly trains every layer of a dataset it's capabilities will never be fully realized.

References

ACL. "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context." *Vimeo*, 14 Jan 2020, https://vimeo.com/384795188

Dai, Zihang, et al. “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context.” *ArXiv.org*, Cornell University, 2 June 2019, https://arxiv.org/abs/1901.02860.

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

Flipkart, Omprakash Sonie. “Deep Natural Language Understanding.” Indian Institute of Technology Patina, 22 Jan 2020, http://www.iitp.ac.in/~ai-nlp-ml/course/dnlp/Deep%20Natural%20Language%20Understanding%20(Om).pdf

Giacaglia, Giuliano. “Transformers.” *Medium*, Towards Data Science, 30 Dec. 2019, towardsdatascience.com/transformers-141e32e69591.

She, Jennifer, and Emma Chen. “Transformer-Based Model for Question and Answering.” *Web.stanford.edu*, Stanford University, https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15791637.pdf.

Su, Dan, et al. “Generalizing Question Answering System with Pre-trained Language Model Fine-tuning.” https://mrqa.github.io/assets/papers/63_Paper.pdf

“Transformer Models in NLP” *Mc.ai*, 11 Jan 2019, https://mc.ai/transformer-models-in-nlp/

Xu, Sam. “Applying Transformer-XL to Q&A.” *Web.stanford.edu*, Stanford University, https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15766157.pdf.

Yang, Zhilin, et al. “XLNet: Generalized Autoregressive Pretraining for Language Understanding.” *ArXiv.org*, Cornell University, 19 June 2019, https://arxiv.org/abs/1906.08237

Yap, Dian Ang, et al. "Faster Transformers for Document Summarization." *Web.stanford.edu*, Stanford University, https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15776950.pdf