# CS 410 Text Information System Tech Review
# Overview of BERT model

Chunjuan Li (NetID: cli111)

## Introduction

Researchers who have been working in the Deep Learning world or Natural Language Processing (NLP) mentioned that the lack of enough training data is one of the biggest challenges when they aimed to make the deep learning based NLP models perform well. Meanwhile, other people may say we have data generated everywhere globally, there is an enormous amount of text data available. However, if we want to create task-specific datasets with annotated labels, we have to divide the data into many different areas which results in limited training data for the NPL model to run accurately. To mitigate the challenge, researchers have developed different kinds of techniques for training general purpose language representation models using web data (known as pre-training). Then the smaller task-specific datasets can be fine-tuned. Compared with the original approach that trains smaller task-special datasets from scratch, this pre-training model approach has significant accuracy improvements. BERT (Bidirectional Encoder Representations from Transformers) was built upon these recent techniques for NLP pre-training, it was created and open-sourced in 2018 by Google, and it presented state-of-the-art results in a wide variety of NLP tasks. A 2020 literature survey concluded that "in a little over a year, BERT has become a ubiquitous baseline in NLP experiments"

In this tech review, I am going to have an overview of BERT to better understand how it works.

## Core idea of BERT

A language model is used to perform the “fill in the blank” task based on the sentence’s context, such as this example, “She likes to go hiking on ___ days”, and “sunny” will be more likely to be filled in the blank than “rainy” by a language model.

When generating the sentence, a general language model (LSTM-based model) uses a one-direction approach that starts from left-to-right or right-to-left or combined left-to-right and right-to-left, and it works well to generate sentences that keep predicting the next word until the last word in the sentence. But for BERT, a bi-directionally (or unidirectional) trained language model can do a deeper sense of the sentence’s context and flow than a single-directional language model, that functionality proceeded by the Masked LM (MLM) technique in BERT. For the “fill in the

blank" example we mentioned above, the model may fill the word "rainy" more likely than "sunny" after the model takes the input not only in a forward direction but also a backward direction, which means "rainy" might be the best fit instead of "sunny" based on the previous and next context of the sentence.

Besides that, BERT is made up of Transformer model architecture instead of LSTMs. The transformer is widely used by most of the latest models, and it does not require sequential data to proceed in order, which means it can process the whole sentence simultaneously. For example, given the sentence, "I arrived at the bank after crossing the river", the Transformer can make the decision on dealing "bank" as the shore of the river instead of the financial institution by its attention to the word "river".

## How does BERT work

As we already knew that BERT is based on Transformer, which is an attention mechanism that learns contextual relations between words in a text. The transformer has 2 separate tasks which are encoder and decoder. BERT model participant in the encoder task to learn the input and context to understand the user's intent. The training of BERT is one in two phases, the first phase is pre-training where the model understands what's the input and context, and the second phase is fine-tuning where the model knows how to solve the problem.

Pre-training phase: For each word user gets the token embedding from the pre-trained, and embedding adds the segment embeddings and position embeddings to build the ordering of the inputs. The inputs will be passed to BERT under the encoder tasks in transformer. BERT training uses 2 strategies which are Masked Language Model (MLM) and Next Sentence Prediction (NSP). The goal of MLM is to fill the word (15%) with masked tokens and output the tokens, this is kind of the "fill in blank" task we talked about previously. This process helps BERT to understand the bi-directional context. And NSP takes sentences and figures out the relationship among different sentences, this helps BERT to understand the context across different sentences. By using those two strategies together, BERT gets a good understanding of the input and context.

Fine-tuning phase: We can further train BERT on very specific tasks, for example, question answering which is a prediction task. BERT can be trained by learning the start vector and end vector from the paragraph to find the best answer to the question. To do fine-tuning, the input data need to be transformed into a specific format, such as tokens marked the sentence with the beginning ([CLS]) and separation/end ([SEP]), and segment IDs which can be used to differentiate the sentences, also comparing with Pre-training, fine-tuning is rather inexpensive.

## How to use BERT

In the real world, it's hard for us to train BERT from scratch with 110M parameters BERT-Base and 340M BERT-Large, and the cost will be extremely high as well.

`BERT-Base` : 12-layer, 768-hidden-nodes, 12-attention-heads, 110M parameters

`BERT-Large` : 24-layer, 1024-hidden-nodes, 16-attention-heads, 340M parameters

Fortunately, there are many implementations of BERT that can help us to just use BERT to fine-tune our model to perform our required tasks with a Cloud TPU (BERT FineTuning with Cloud TPUs), such as Tensorflow (https://www.tensorflow.org/tfmodels/nlp/fine_tune_bert) and PyTorch (https://github.com/huggingface/transformers). Also, there are many good resources and examples we can easily find to help us to get started with BERT.

## Conclusion

As we can see, BERT is definitely a very powerful model in NLP, and in the meanwhile, it's also a very large model that has over a hundred million weights (embedding layer and Transformers). That causes the fine-tuning to be relatively slow, also the referencing is slow when we run BERT on some texts. There are plenty of online materials available for us to deeply explore BERT, and no doubt that there's still a chance for us to improve BERT after we learn and use more about it.

## References

https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/
https://www.tensorflow.org/
https://www.tensorflow.org/tfmodels/nlp/fine_tune_bert