**Justyn Lao**

# CS410 Technology Review - TensorFlow NLP

## Overview

TensorFlow is an open-source machine learning library created by Google. A key benefit of TensorFlow is how it caters to both beginners and experts. In fact, TensorFlow provides curriculums of varying levels to learn the foundations of machine learning. Additionally, TensorFlow has support for Python, JavaScript, Mobile & IoT, as well as production. This wide spectrum of capabilities makes TensorFlow a great option for a variety of machine learning needs, for hobbyist developers and large corporations alike.

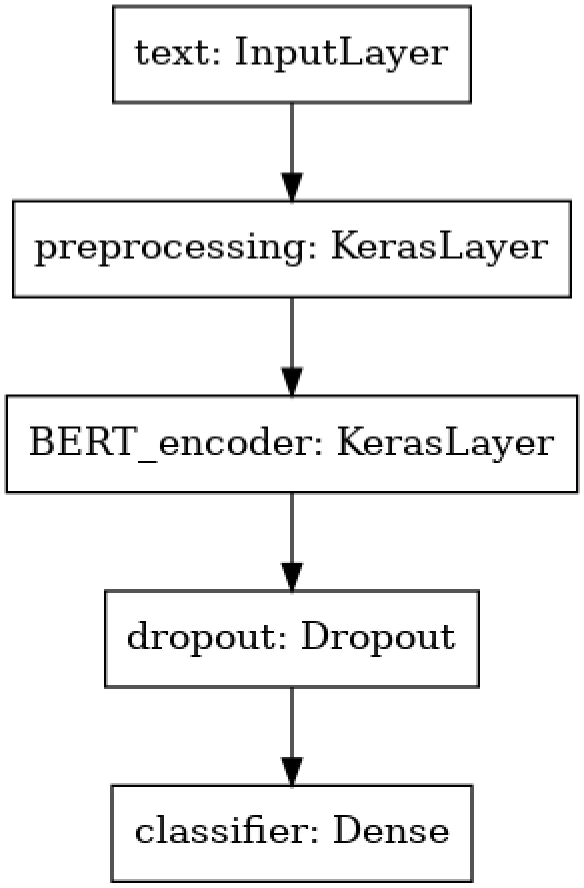
TensorFlow offers many collections and models for performing Natural Language Processing (NLP) tasks. A key part of NLP is text preprocessing, and TensorFlow features a Text collection that allows users to perform many standard preprocessing operations on text. A major part of text preprocessing is tokenizing text; TensorFlow provides several tokenizers, including Whitespace, Unicode, Wordpiece, Sentencepiece, UnicodeChar, and more.

TensorFlow's primary type of model for text classification is Bidirectional Encoder Representations from Transformers, known as BERT. BERT uses the transformer encoder architecture to process tokens of text in the full context of all tokens. BERT models can then be used and fine-tuned for several tasks, such as sentiment analysis.

Overall, TensorFlow seems like a solid resource for NLP, but in the eyes of a beginner to machine learning, the sheer number of options when it comes to models and collections can be somewhat overwhelming.

## Sentiment Analysis with TensorFlow

BERT models come pre-trained on a large corpus of text, so performing specific tasks such as sentiment analysis can be done with just some fine-tuning. First, a preprocessing model needs to be selected, as text needs to be tokenized and arranged into Tensors before they can be input to the BERT model. The preprocessor divides text into a mask, word IDs, and type IDs, which are necessary for BERT models. The BERT model itself then needs to be selected, and given input text would output a map with three keys: pooled output, sequence output, and encoder outputs. Given the preprocessor and BERT models, one can then add layers and create a model. A depiction of such a model can be seen in the diagram below.



After the model is established, it needs to be trained for sentiment analysis. By incorporating built-in TensorFlow functions to calculate loss and optimize, the model can be trained and then finally evaluated based on accuracy. Lastly, the model can be exported and used to test any sentence.

This entire process was documented through one of TensorFlow's tutorials. Although the process seems fairly streamlined due to all the built-in models and functions provided by

TensorFlow, there is still a lot to digest in terms of deciding what functions and models to choose for what situations.

**Comparison with NLTK Sentiment Analysis**

Natural Language Toolkit (NLTK) is a Python library specifically for working with human language data. NLTK is far smaller when compared to TensorFlow, but when it comes to NLP, it still has a comprehensive library of utilities.

Contrary to the vast amount of offerings that TensorFlow has, NLTK just has two sentiment analysis tools; there is one general sentiment analyzer that needs to be trained, and one using “Valence Aware Dictionary and sEntiment Reasoner“ (VADER) that comes pre-trained. With fewer options, the whole sentiment analysis process becomes a lot less intimidating, especially for beginners. A simple Naive Bayes Sentiment Model can be created very quickly. Given documents of text data, unigram word features can be created and applied in order to obtain feature-value representations. These can then be used to train the classifier and subsequently output evaluation results.

The VADER sentiment analyzer is even simpler; after tokenizing some text, the polarity scores utility function can just be called on each sentence to get scores for positive, neutral, negative, and compound.

**Conclusion**

If one only requires smaller, more specific NLP needs, NLTK appears to be the better choice in terms of not getting lost in the realm of machine learning. It would likely be ideal for those looking to get started with NLP programming faster. For larger-scale projects, TensorFlow is probably the better choice; further studying and reading documentation might be necessary, however. TensorFlow also has more resources in case help is needed, as is expected with such a popular library.

## References

https://www.tensorflow.org/text/tutorials/classify_text_with_bert

https://www.tensorflow.org/text/guide/bert_preprocessing_guide

https://www.tensorflow.org/text/guide/tf_text_intro

https://www.nltk.org/howto/sentiment.html

https://www.nltk.org/api/nltk.sentiment.vader.html