A Brief Technical Review of TensorFlow and its applications of Natural Language Processing

Jiayi Chen (jiayic15)
CS410: University of Illinois at Urbana-Champaign
October 30, 2020

Overview

TensorFlow is a famous open source software library for machine learning, text retrieval and text classification. It provides a vast number of packages and able to conduct range of tasks focus on training and inference of neural networks.

TensorFlow was developed by Google Brain team and now it's released under the Apache License in 2015. TensorFlow enables us to utilize and develop our own machine learning toolkits for text retrieval and text mining along with other libraries such as keras, caffe and pytorch.

In this technical review, we are going to review 2 state-of-the-art text classification methods (BERT and RNN) implemented on TensorFlow and look into their performances. Also, we aim to analyze how TensorFlow distinguished among other Machine Learning libraries.

Text classification with BERT

BERT stands for Bidirectional Encoder Representations and it's applicable to Word Embeddings and text representations. BERT was developed by researchers at Google in 2018 and has been proven to be state-of-the-art for a variety of natural language processing tasks such text classification, text summarization, text generation, etc.

One reason why BERT is so efficient is because of its Pre-training phase, which allows BERT to learn from and familiar with the context and training environment. Then it will be faster in the Fine-Tuning phase, since it's already learned some of the context and variables in Pre-training phase.

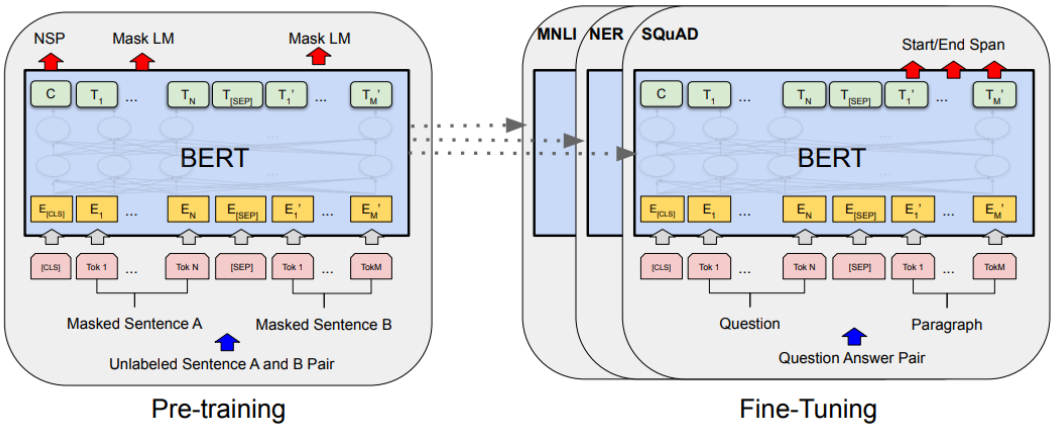

Figure 1. Overall pre-training and fine-tuning procedures for BERT (BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding)

When using TensorFlow to implement BRET, we used IMDB Dataset. We intend to use BERT tokenizer to create word embeddings that can be used to perform text classification and finally conduct sentiment analysis.

During the process, TensorFlow provides sophisticated packages and pre-defined functions which make us easy to set up the input/output pipelines, create BERT Tokenizer, and create the Model.

```
BertTokenizer = bert.bert_tokenization.FullTokenizer
bert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/1",
                            trainable=False)
vocabulary_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
to_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
tokenizer = BertTokenizer(vocabulary_file, to_lower_case)
```

Figure 2. Creating a BERT Tokenizer

```
class TEXT_MODEL(tf.keras.Model):

    def __init__(self,
                 vocabulary_size,
                 embedding_dimensions=128,
                 cnn_filters=50,
                 dnn_units=512,
                 model_output_classes=2,
                 dropout_rate=0.1,
                 training=False,
                 name="text_model"):
        super(TEXT_MODEL, self).__init__(name=name)

        self.embedding = layers.Embedding(vocabulary_size,
                                          embedding_dimensions)
        self.cnn_layer1 = layers.Conv1D(filters=cnn_filters,
                                        kernel_size=2,
                                        padding="valid",
                                        activation="relu")
        self.cnn_layer2 = layers.Conv1D(filters=cnn_filters,
                                        kernel_size=3,
                                        padding="valid",
                                        activation="relu")
        self.cnn_layer3 = layers.Conv1D(filters=cnn_filters,
                                        kernel_size=4,
                                        padding="valid",
                                        activation="relu")
        self.pool = layers.GlobalMaxPool1D()

        self.dense_1 = layers.Dense(units=dnn_units, activation="relu")
        self.dropout = layers.Dropout(rate=dropout_rate)
        if model_output_classes == 2:
            self.last_dense = layers.Dense(units=1,
                                           activation="sigmoid")
        else:
            self.last_dense = layers.Dense(units=model_output_classes,
                                           activation="softmax")
```

Figure3. Creating the model

Finally, we performed sentimental analysis of IMDB movie reviews and achieved an accuracy of 89.26%. All the packages of TensorFlow helped us delivered the tasks.

Text classification with RNN

Using RNN (Recurrent Neural Networks) to conduct text classification is efficient and robust. By using the same dataset to do the sentimental analysis, we are going to review how TensorFlow help us to conduct tasks.

The one most signature feature of RNN is: as a class of neural network, it allows previous outputs to be used as inputs while having hidden states.

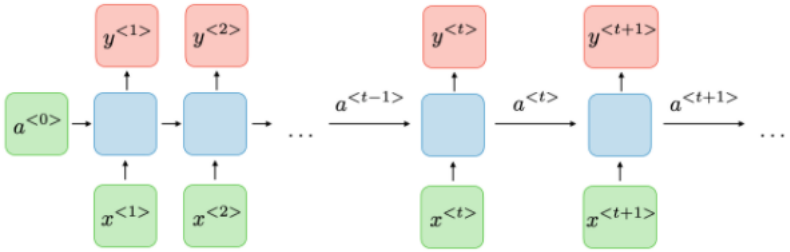

Figure4. Architecture of a traditional RNN

RNN models are mostly used in the fields of natural language processing. In this case, we trained a RNN on the IMDB large movie review dataset for sentiment analysis using TensorFlow.

As mentioned, TensorFlow provides many dedicated pre-defined functions and packages for us to accelerated the process of setting up pipelines, training datasets, and building models.

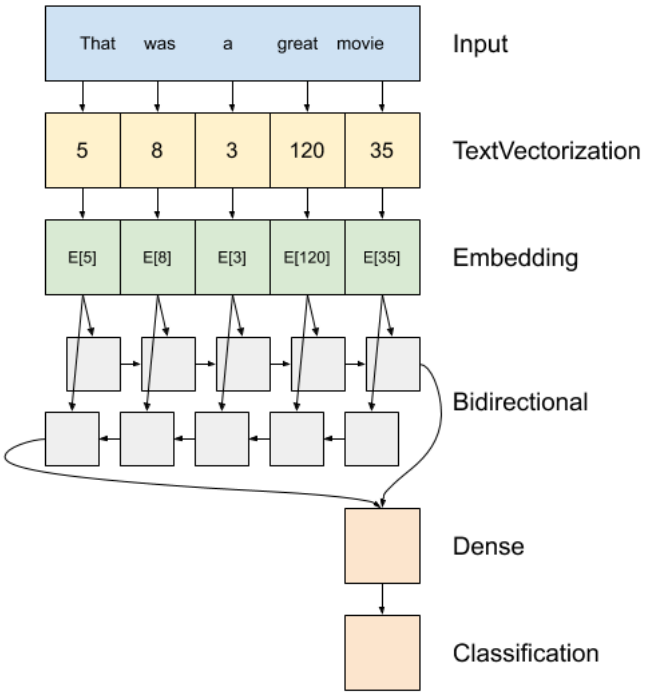

Figure5. Diagram of the RNN model

```
model = tf.keras.Sequential([
    encoder,
    tf.keras.layers.Embedding(
        input_dim=len(encoder.get_vocabulary()),
        output_dim=64,
        # Use masking to handle the variable sequence lengths
        mask_zero=True),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
])
```

Figure6. Building model using TensorFlow.Keras function

As a result, the accuracy of text classification using RNN is 86.35%. Similar with the one using BERT. Again, the implementation of 2 methods shows the robustness and efficiency of TensorFlow.

How TensorFlow distinguished

Among so many famous Machine Leaning Libraries, how TensorFlow wins the competition? It might due to below key functionalities:

- TensorFlow provides an accessible and readable syntax which is essential for making these programming resources easier to use
- TensorFlow provides excellent functionalities and services when compared to other popular deep learning frameworks
- TensorFlow is a low-level library which provides more flexibility
- TensorFlow provides more network control which allowing developers and researchers to understand how operations are implemented across the network

Library	Rank	Overall	Github	Stack Overflow	Google Results
tensorflow	1	10.87	4.25	4.37	2.24
keras	2	1.93	0.61	0.83	0.48
caffe	3	1.86	1.00	0.30	0.55
theano	4	0.76	-0.16	0.36	0.55
pytorch	5	0.48	-0.20	-0.30	0.98
sonnet	6	0.43	-0.33	-0.36	1.12
mxnet	7	0.10	0.12	-0.31	0.28
torch	8	0.01	-0.15	-0.01	0.17
cntk	9	-0.02	0.10	-0.28	0.17
dlib	10	-0.60	-0.40	-0.22	0.02
caffe2	11	-0.67	-0.27	-0.36	-0.04
chainer	12	-0.70	-0.40	-0.23	-0.07
paddlepaddle	13	-0.83	-0.27	-0.37	-0.20
deeplearning4j	14	-0.89	-0.06	-0.32	-0.51
lasagne	15	-1.11	-0.38	-0.29	-0.44
bigdl	16	-1.13	-0.46	-0.37	-0.30
dynet	17	-1.25	-0.47	-0.37	-0.42
apache singa	18	-1.34	-0.50	-0.37	-0.47
nvidia digits	19	-1.39	-0.41	-0.35	-0.64
matconvnet	20	-1.41	-0.49	-0.35	-0.58
tflearn	21	-1.45	-0.23	-0.28	-0.94
nervana neon	22	-1.65	-0.39	-0.37	-0.89
opennn	23	-1.97	-0.53	-0.37	-1.07

Table1. Rankings of 23 deep learning libraries

Conclusion

In this technical review, we basically walk through the implementations of 2 famous text classification methods: BERT and RNN. We examined the feasibility and robustness of TensorFlow as a Machine Learning library; we also compare it with other data science libraries and draw conclusions regarding how and why TensorFlow proved to be one of the best open source Machine Learning Library: accessible and readable syntax, excellent functionalities and services, low-level library, and more network control.

References

Text classification with an RNN https://www.tensorflow.org/tutorials/text/text_classification_rnn

Fine-tuning a BERT model https://www.tensorflow.org/official_models/fine_tuning_bert

Why TensorFlow always tops machine learning and artificial intelligence tool surveys https://hub.packtpub.com/tensorflow-always-tops-machine-learning-artificial-intelligence-tool-surveys/#:~:text=TensorFlow%20is%20an%20open%20source,%2C%20mobiles%2C%20and%20edge%20devices

Text Classification with BERT Tokenizer and TF 2.0 in Python https://stackabuse.com/text-classification-with-bert-tokenizer-and-tf-2-0-in-python/

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding https://arxiv.org/pdf/1810.04805.pdf