Technology Review -TensorFlow

Utpal Mondal
umondal2@illinois.edu

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs; and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift (early release). Third-party packages are available for C#, Haskell, Julia, MATLAB, R, Scala, Rust, OCaml, and Crystal. New language support should be built on top of the C API. However, not all functionality is available in C yet. Some more functionality is provided by the Python API.

TensorFlow provides multiple official models for Natural language processing such BERT (Bidirectional Encoder Representations from Transformers), ALBERT (A Lite BERT), NHNet (News Headline generation model), Transformer and XLNet. Official models are well optimized for speed/performance, but still they are easy to read and understand. These models are maintained, tested and kept to up to date with latest version of TensorFlow. Also, there are official computer vison models for Image Classification (MNIST, ResNet, EfficientNet) and Object Detection and Segmentation (RetinaNet, Mask R-CNN, ShapeMask and SpineNet).

TensorFlow runs and graphs can be inspected and understood using the TensorBoards, suite of web applications provided by TensorFlow. Some of the common TensorBoards are Scalar Dashboard used to visualize the statistics that vary over the time; Histogram Dashboard displays the statistical distribution varied over the time; Distribution Dashboard, Image Dashboard, Audio Dashboard, Graph Explorer, Embedding Projector, Text Dashboard.

TensorFlow Lite is TensorFlow's lightweight solution for mobile and embed devices. It runs the machine learned model on mobile device with low latency so one can do classification, regression etc. without making roundtrip to the server. It is supported on IOS devices using the C++ API and on android devices through Java Wrapper APIs. To run the model on mobile, model will be built separately and exported to mobile devices.

TensorFlow with Keras has numerous applications like Text Classification, Sentiment Analysis, Image Recognition, Speech Recognition, Object Tagging etc.

Basic Text Analysis/Text Classification using TensorFlow-Keras API involves below steps at a high-level,

Understand the problem statement: Basically, we would like to understand the end goal of the model here for example classify Twits as Sarcastic/Non-Sarcastic, classify reviews as negative/positive etc.

Understand the data-set: Next step is to understand the data set available for us to train and test the data set. This involves length and dimension of the data, number of records available for training etc. Many of the commonly used data sets are already exposed through Keras API.

Pre-process the examples and labels: Next step is to preprocess the data in order to feed the data into model/neural network. This step involves tokenizing the data, padding the data using pad_sequences and converting the text representation into numerical representation. This data preparation step can be performed using the Tokenizer API provided with Keras.

Understand word embedding: A word embedding involves representing words and documents using a dense vector representation. In an embedding, position of the word in embedding is learned by words surrounding it in vector space rather than individual words. Keras offers extensive API for creating embedding layer that can be used for neural network model on text data.

Create a neural network model: Next step we create the model that will be trained using the training data and evaluated using the test data. Keras model API provides 3 ways to create models,

The Sequential model, consists of a simple list of layers. This layer limited to single-input, single-output stacks of layers as the name suggests.

The Functional API, which is an easy-to-use, fully-featured API that supports arbitrary model architectures. This is the model that should fit to most use cases in the real-world scenario.

Model subclassing, this is used when model requirement does not fit into the existing API and need to build custom implementation from scratch. We can obtain the summary of the model created using the Model. Summary method that prints a string summary of the network.

Train the model to fit the data set: Keras network model training involves compiling and fitting the model to training data. Model. Compile method takes multiple input to compile the model. Main inputs to compile method are optimizers, losses, and metrics. Most of the cases you won't have to create optimizers, losses, and metrics, Keras API provides multiple optimizers such as SGD, RMSprop, Adam etc.; Losses such as MeanSquaredError, KLDivergence, CosineSimilarity etc. and Metrics such as AUC, Precision, Recall, etc.

Evaluate the model: Evaluate method provided by Keras API does the computation in batches and returns the loss value & metrics values for the model. Evaluate method takes multiple parameters such as batch size (default to 32 if not specified), set multiprocessing parameter to use process-based threading and also API provides the flexibility to use call back methods.

References:

https://en.wikipedia.org/wiki/TensorFlow
https://www.tensorflow.org/
https://keras.io/
https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf