Jonathan McDowell
jcmcdow2@illinois.edu

Tech Review

Natural language processing has been receiving increased attention as a problem that can addressed by recent machine learning improvements. Processing language has long been a difficult task for computers due to the data structure, complexity, ambiguity, and computational resources required. Organizations typically generate a significant amount of text data through surveys and other user reviews. Being able to quickly mine the topic and determine the sentiment can save a large amount of manual work. There are many frameworks out there that allow users to create text machine learning models. Three of the most popular are TensorFlow, PyTorch and the Natural Language Toolkit.

TensorFlow was released in 2015 by Google as a successor to DistBelief. TensorFlow's initial release proved a flexible interface to allow users to create deep learning models. While TensorFlow provided flexibility it did not provide the end-to-end pipelines needed to create production ready models. TensorFlow has continued to evolve. There are currently 4 major versions of TensorFlow, TensorFlow core, TensorFlow.js, and TensorFlow Lite. TensorFlow has also pulled in Keras, another machine learning framework.

TensorFlow has been proven useful for many NLP tasks. TensorFlow has several methods for creating word embeddings. You can define your own algorithm for creating word embeddings. You can use Keras or TensorFlow layers to create a word embedding. It's also possible to pull in pretrained word embeddings like GloVe and Word2Vec using TensorFlow Hub.

TensorFlow can be used to create neural networks capable of text generation or text classification. Recurrent Neural Networks models can be easily created for text generation or classification. It exposes several methods that allow you to link as many neural network layers together as you need. By stacking these layers, you can create a custom neural network model

that fits your needs. By choosing different layers you can create CNN, LSTM or RNN models. TensorFlow also makes it easy to create transformer models in the same way.

Another big part of TensorFlow is TFX. TFX is used for creating production machine learning pipelines. The goal is to make it easier for users to develop and deploy their machine learning models automatically. By automating as much of the process as possible teams are able to work more efficiently and focus on the important tasks. TFX also makes it easier for models to be updated as new data is added. TFX offers several benefits such as standardization, growth, and reproducibility/repeatability. These three benefits help to ensure consistent reliability when serving TensorFlow models.

PyTorch is another popular machine learning framework. Like TensorFlow, it offers significant flexibility that allows it to be useful for natural language processing. It is currently considered the standard for researching new models. PyTorch was created by Meta in 2016. It was based on the Torch library which was created in 2002. PyTorch takes advantage of Tensors, which are essentially the same as NumPy arrays, but can be ran on both CPUs and GPUs. It gained a significant amount of popularity because of its flexibility, speed and ease of use.

It has proven useful for many machine learning tasks such as natural language processing, image classification, and data predictions. For specifically NLP problems, PyTorch has been used for language modeling, part of speech tagging, text generation, topic mining, and sentiment analysis.

Both frameworks have multiple outside packages that expand the core functionality, but PyTorch has one specifically for NLP tasks called TorchText. TorchText has many common datasets designed for NLP tasks such as text classification, language modeling, machine translation, sequence tagging, question answering and unsupervised learning. Using these datasets can be useful when learning or exploring different text methods, as it saves the user the time of searching for and cleaning datasets. TorchText also contains several methods for transforming text data into appropriate input such as SentencePieceTokenizer, GPT2BPETokenizer, BERTTokenizer and several others. This can allow users to focus on the

model building aspect. This package also adds additional text specific neural network layer options such as MultiHeadAttentionContainer, InProjContainer, and ScaledDotProduct.

There are a few key differences between the two frameworks. One area where PyTorch has an advantage on TensorFlow is the availability of state-of-the-art models. For NLP tasks the current state-of-the-art models are impossible for individuals or small companies with a limited budget to create. GPT-3 has more than 175 billion parameters making it difficult to train, and then use for predictions. Because of this, providers such as HuggingFace has made a large pre-trained models available. Looking at the total models PyTorch has significantly more models available around 85% of their models working exclusively with PyTorch, 5% work exclusively with TensorFlow, and around 10% work with both frameworks.

One key difference between PyTorch and TensorFlow is how the models are served. While TensorFlow's TRX has many featuresforautomating the analysis and deployment of TensorFlow models, PyTorch's version, TorchServe, is much more limited.
PyTorch excels in many of the same tasks as TensorFlow. Many tasks could be accomplished with the same result using either framework.

Another key difference between the two frameworks is how they can be deployed. As discussed earlier, TensorFlow's TRX can be used to manage the full machine learning lifecycle. TensorFlow also provides TensorFlow Lite, that allows models to be used on mobile or embedded devices with limited computational power. PyTorch released TorchServe in 2020 which allows some deployment activities, but it isn't as robust as TRX. For mobile devices,

In conclusion, there are many similarities between TensorFlow and PyTorch. Both frameworks can take advantage of GPUs and offer substantial flexibility. Either would be a good choice for NLP tasks when building models from scratch. If the latest researched models are needed, PyTorch will be easier to utilize and likely easier to recreate the model. If the models will be deployed and used in a production environment TensorFlow's TRX will be a more robust choice. The user's needs will determine which framework makes the most sense, but either framework will work more often than not.

References

https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/

https://alexmoltzau.medium.com/pytorch-governance-and-history-2e5889b79dc1

https://medium.com/@saitejaponugoti/nlp-natural-language-processing-with-tensorflow-b2751aa8c460

https://blog.tensorflow.org/2020/09/brief-history-of-tensorflow-extended-tfx.html

https://blog.tensorflow.org/2022/10/how-startups-can-benefit-from-tfx.html

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

https://towardsdatascience.com/how-to-create-word-embedding-in-tensorflow-ed0a61507dd0

https://pytorch.org/text/stable/index.html

https://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html

https://www.techleer.com/articles/591-know-the-basic-idea-behind-torchserve-and-torchelastic/

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