NLP Modeling Library

This library provides a set of Keras primitives (`tf.keras.Layer` and `tf.keras.Model`) that can be assembled into transformer-based models. They are flexible, validated, interoperable, and both TF1 and TF2 compatible.

- `layers` are the fundamental building blocks for NLP models. They can be used to assemble new `tf.keras` layers or models.
- `networks` are combinations of `tf.keras` layers (and possibly other networks). They are `tf.keras` models that would not be trained alone. It encapsulates common network structures like a transformer encoder into an easily handled object with a standardized configuration.
- `models` are combinations of `tf.keras` layers and models that can be trained. Several pre-built canned models are provided to train encoder networks. These models are intended as both convenience functions and canonical examples.
- `losses` contains common loss computation used in NLP tasks.

Please see the colab nlp_modeling_library_intro.ipynb for how to build transformer-based NLP models using above primitives.

Besides the pre-defined primitives, it also provides scaffold classes to allow easy experimentation with noval achitectures, e.g., you don't need to fork a whole Transformer object to try a different kind of attention primitive, for instance.

- `TransformerScaffold` implements the Transformer from ["Attention Is All You Need"] (https://arxiv.org/abs/1706.03762), with a customizable attention layer option. Users can pass a class to `attention_cls` and associated config to `attention_cfg`, in which case the scaffold will instantiate the class with the config, or pass a class instance to `attention_cls`.
- `EncoderScaffold` implements the transformer encoder from "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", with customizable embedding subnetwork (which will replace the standard embedding logic) and/or a custom hidden layer (which will replace the Transformer instantiation in the encoder).

Please see the colab customize_encoder.ipynb for how to use scaffold classes to build noval achitectures.

BERT and ALBERT models in this repo are implemented using this library. Code examples can be found in the corresponding model folder.