Embeddings

Use TensorFlow's built-in functions for creating embeddings.

Chapter Goals:

- Use TensorFlow feature columns to create input embeddings

A. Feature columns

So far, we've dealt with pre-training an embedding model as well as training an embedding model in tandem with an LSTM. In both of these cases, we had to write our own embedding matrix and handle its initialization.

A third option for incorporating word embeddings into your LSTM training model is by using TensorFlow's feature column API. Most of these feature columns won't be discussed in this course, since they aren't directly relevant to NLP.

The feature column function we will focus on is `tf.feature_column.embedding_column`. It lets you incorporate an embedding matrix automatically into your model, which will be trained alongside the LSTM.

B. Sequential categorical column

The `embedding_column` function takes in two required arguments. The second required argument is just the embedding size, which we normally set to the $ 4^{th}$ root of the vocabulary size. The first required argument is something called a categorical column.

The feature columns API provides functions for creating categorical columns, but none of these functions work for sequential data (i.e. data with time steps). For functions that work with sequential data, we need to use the extended feature column API, found in `tf.contrib.feature_column`.

Note: Any code found in the `tf.contrib` submodule is volatile, meaning that it may be moved to the main TensorFlow module in future updates.

the main TensorFlow module in future updates.

There are four functions for creating categorical columns with sequential data:

- `sequence_categorical_column_with_identity`
- `sequence_categorical_column_with_hash_bucket`
- `sequence_categorical_column_with_vocabulary_file`
- `sequence_categorical_column_with_vocabulary_list`

Since each of the vocabulary words in the text corpus are converted to unique integer IDs, we only need to focus on the first function. The other three functions all involve some form of text sequence processing, such as hashing words into IDs or using a specified vocabulary list/file.

The categorical column itself is a placeholder for a categorical feature. Categorical columns don't contain any data until the computation graph is run, similar to `tf.placeholder` objects.

The `sequence_categorical_column_with_identity` function takes in two required arguments: a string name for the categorical column and the vocabulary size of the text corpus. The string name will be used when creating the input dictionary for the main conversion function.

The example below creates an embedding column (`embed_col`) using a categorical column (`input_col`) as input.

```
import tensorflow as tf
vocab_size = 10000
input_col = tf.contrib.feature_column \
              .sequence_categorical_column_with_identity(
                  'input', vocab_size)
embed_size = int(10000**0.25)
embed_col = tf.feature_column.embedding_column(
                  input_col, embed_size)
```

C. Converting to embeddings

The main conversion function used to create the embedded input sequences is `sequence_input_layer`. Like the other functions, it takes in two required

arguments as well. The first is a dictionary of input data, where each key is the name of a categorical column. The second is a list of feature columns corresponding to data in the input dictionary.

In our case, since the input data is just a batch of tokenized sequences, the dictionary will only contain a single key-value pair. The key will be the same string we set in the `sequence_categorical_column_with_identity` function. Furthermore, the second argument will just be a list containing the embedding column.

Below, we show an example using `sequence_input_layer` with tokenized sequences (`input_seqs`) and `embed_col` from the previous example.

```
import tensorflow as tf
# Input batch of tokenized sequences (30 time steps)
input_seqs = tf.placeholder(tf.int64, shape=(None, 30))

input_dict = {'input': input_seqs}
embed_seqs, sequence_lengths = tf.contrib.feature_column \
                                  .sequence_input_layer(
                                      input_dict, [embed_col])
```

The output of `sequence_input_layer` is a tuple containing the embedded sequences and the sequence lengths. The sequence lengths output is used for calculating the sequence lengths for variable length sequence inputs.