DeepLearning.AI Tensorflow Developer

Natural Language Processing in TensorFlow: Week 1

Word Based Encoding

Word Encoding (Tokenization): Assign numeric value to each word in vocabulary

TextVectorization layer's adapt method generate vocab and create encoding vector from sentences, get_vocabulary method returns the vocabulary list, setting include_special_token as True gives empty space (' ') and UNK token.

```
vectorization_layer = tf.keras.layers.TextVectorization()

vectorization_layer.adapt(sentences)

vocab = vectorization_layer.get_vocabulary(include_special_tokens=False) [Exclude
punctuation, Default True]

print(vocab)
```

Text to Sequence

TextVectorization layer adapted to given sentences, transforms input sentence into tensor of token values.

```
sequence = vectorization_layer(input_sentence)

print(sequence)
```

For multiple sentences padding (Zero Value) are added at the end of the sentences. Padding added to make all token list of sentences equal size. Max sequence length is the length of the longest sentence.

In order to create pre-padding, transform input sentences into tensoflow dataset. Then apply vectorization_layer on sentences as map function to create token length of original sentence length. From Keras utils pad_sequences method apply pre-padding on sentences.

```
sentences_dataset = tf.data.Dataset.from_tensor_slices(input_sentences)

mapped_sequences = sentences_dataset.map(vectorization_layer)

sequences_pre = tf.keras.utils.pad_sequences(mapped_sequences, padding='pre')
```

TextVectorization layer with ragged set to True gives token sequences of original length of the sentences. Apply pad_sequences method on ragged token vectors for pre-padding. Also the maxlen parameter of pad_sequences method apply max length of tokens for input sentences.

```
ragged_layer = tf.keras.layers.TextVectorization(ragged=True)

ragged_layer.adapt(sentences)

ragged_sequences = ragged_layer(input_sentences)

sequences_pre = tf.keras.utils.pad_sequences(ragged_sequences.numpy())

print(sequences_pre)
```

TextVectorization maps unknown words from input sentences into [UNK] token value.

Loading Dataset

Load dataset in json format from public sarcasm dataset.

```
import json

with open("./sarcasm.json", 'r') as f:

        datastore = json.load(f)

headlines = []

for item in datastore:

        headlines.append(item['headline'])
```