Language Model

Learn how language models are trained to calculate word probabilities.

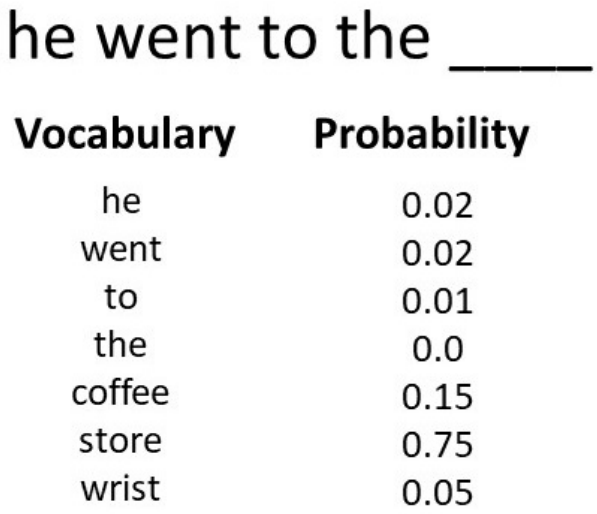
Chapter Goals:

- Understand how a language model works
- Learn how to set up the training data for a language model

A. Word probabilities

As mentioned in the introduction, the purpose of a language model is to assign probabilities to words in sequences of text. The probability for each word is conditioned on the words that appear before it in the sequence. Though it may not seem like it at first glance, the task of calculating a word probability based on the previous words in a sequence is essentially multiclass classification.

Since each word in a sequence must come from the text corpus' vocabulary, we consider each vocabulary word as a separate class. We then use the previous sequence words as input into our language model to calculate probabilities for each vocabulary class.



Multiclass probabilities over the 7 possible vocabulary classes. In this case, the sequence ["he", "went", "to", "the"] is the input to the language model.

B. Inputs & targets

The main idea behind training a language model is to predict each word in a sequence based on the words that come before it. However, rather than making training pairs where the target is a single word, we instead make training pairs consisting of equal length input and target sequences.

Original Sequence: ["she", "bought", "a", "book", "from", "me"]

Input Sequence: ["she", "bought", "a", "book", "from"]

Target Sequence: ["bought", "a", "book", "from", "me"]

Example training pair from a corpus text sequence. Notice that the target sequence is just the input sequence shifted to the right by one word.

The language model attempts to correctly predict each word in the target sequence based on its corresponding prefix in the input sequence. In the example above, the input prefix-target word pairs are:

- (["she"], "bought")
- (["she", "bought"], "a")
- (["she", "bought", "a"], "book")
- (["she", "bought", "a", "book"], "from")
- (["she", "bought", "a", "book", "from"], "me")

For each pair, we'd ideally want the language model to calculate a very high probability for the target word based on the input prefix. This is the same goal as training a multiclass classification model.

C. Maximum length

When creating the input-target sequences for training a language model, a big factor to consider is the length of the sequences. Depending on the text corpus, we can choose to limit our training pair sequences to a fixed maximum length. Using a fixed max sequence length can increase the training speed and also help the language model avoid overfitting uncommon text dependencies (which are sometimes found in long, run-on sentences).

Original Sequence: ["they", "went", "to", "the", "big", "red", "truck"]

Input Sequence: ["they", "went", "to"]

Target Sequence: ["went", "to", "the"]

Example training pair with a fixed max length of 4. Notice that only the first 4 words from the original sequence are considered for the input-target pair.

Time to Code!

In this section of the course, you'll be creating a `LanguageModel` object.

Specifically in this chapter, you'll be completing a helper function for the `get_input_target_sequence` function, which converts a sequence into an input-target pair.

The function you'll complete is the `truncate_sequences` function. This function is used when the length of `sequence` is not less than `self.max_length`.

In this case, we focus on the prefix of `sequence` with length `self.max_length`. The input sequence will be equal to that prefix minus the final token, while the target sequence will be equal to the prefix minus the first token.

Inside the `truncate_sequences` function, set `input_sequence` equal to `sequence` containing every token up to `max_length - 1` (exclusive).

Inside the `truncate_sequences` function, set `target_sequence` equal to `sequence` containing every token from index `1` (inclusive) up to `max_length` (exclusive).

Return a tuple with `input_sequence` as the first element and `target_sequence` as the second element.

```
import tensorflow as tf

def truncate_sequences(sequence, max_length):
    # CODE HERE
    pass

# LSTM Language Model
class LanguageModel(object):
    # Model Initialization
    def __init__(self, vocab_size, max_length, num_lstm_units, num_lstm_layers):
        self.vocab_size = vocab_size
        self.max_length = max_length
        self.num_lstm_units = num_lstm_units
        self.num_lstm_layers = num_lstm_layers
        self.tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=vocab_size)

    def get_input_target_sequence(self, sequence):
        seq_len = len(sequence)
        if seq_len >= self.max_length:
            input_sequence, target_sequence = truncate_sequences(
                sequence, self.max_length
            )
        else:
            # Next chapter
```

```
        input_sequence, target_sequence = pad_sequences(
            sequence, self.max_length


        )
    return input_sequence, target_sequence
```