Programming for Computational Linguistics 2018/2019

Final Project Checkpoint 2 — n-grams

These checkpoint documents provide a detailed recommendation for how to proceed with the project. They will break the requirements into small, well-defined steps. It is not necessary to follow this plan — you can choose to implement the requirements however makes the most sense to you. However, if you are unsure of how to proceed, this should provide you a good starting point.

In this checkpoint, we will start working on the module `lm.py`. Since `corpus.py` handles converting between strings and token sequences, in this file, we can only deal with token sequences, and never have to deal with full text strings. We need to define our class `LanguageModel`.

Start by making a constructor for it. The constructor accepts as a parameter n. Our instance will need to remember what n is, so store this as an instance variable. As we implement more methods, we might find that we will want our instances to have more instance variables, and we might add stuff to the constructor later.

Next, let's implement a function that isn't required directly in the specification, but which will make our lives easier going forward: `get_ngrams(tokens, n)`. We will define this as a function outside of our class (although we could also implement it as a class method if we want to). As input, `get_ngrams` accepts `tokens`, a list of strings, and `n`, the n in n-gram. This function should return a list of n-grams present in a token sequence. Each n-gram can be represented as a tuple of tokens, where each token is a string. As explained in the Theory section, it is useful to "pad" our input sequence with PAD symbols, so we will do this, using `None` as our PAD symbol. As a concrete example, we want a function that behaves as follows:

```
>>>get_ngrams(['the', 'cat', 'in', 'the', 'hat'], 4)
[(None, None, None, 'the'), (None, None, 'the', 'cat'), (
    None, 'the', 'cat', 'in'), ('the', 'cat', 'in', 'the')
    , ('cat', 'in', 'the', 'hat'), ('in', 'the', 'hat',
    None), ('the', 'hat', None, None), ('hat', None, None,
     None)]
```

Implement such a function, and implement a unit test to make sure it works. Make sure you remember to test for edge-cases, such as short (and empty) token sequences.