Erika Fox

Provide some example applications of your function in both deterministic and stochastic modes, for a few sets of seed words and a few different n.

Case 1 (provided case):

Input:

```
words = finish_sentence(
     ['she', 'was', 'not'],
      3,
      corpus,
      True,
  )
```

Output: ['she', 'was', 'not', 'in', 'the', 'world', '.']

After accomplishing the result of this provided case, I was curious to see what the stochastic mode would output. My function made a surprising amount of sense most of the time:

Case 2:

Input:

```
words = finish_sentence(
     ['she', 'was', 'not'],
      3,
      corpus,
      False,
  )
```

Output: ['she', 'was', 'not', 'to', 'take', 'advantage', 'of', 'any', 'thing', 'like', 'it', '.']

Next, I wanted to try giving my function less information, so I reduced the set of words to “she was” and took n down to 2 and tried it in both the deterministic and stochastic modes. I found that these cases made significantly less sense, as expected.

Case 3:

Input:

```
words = finish_sentence(
     ['she', 'was'],
      2,
```

```
        corpus,
        True,
    )
```

Output: ['she', 'was', 'not', 'be', 'a', 'very', 'well', ',', 'and', 'the', 'same', 'time', ',', 'and', 'the']

Case 4:

Input:

```
words = finish_sentence(
     ['she', 'was'],
      2,
      corpus,
      False,
    )
```

Output: ['she', 'was', 'deeply', 'interested', 'and', 'housekeeper', ',', 'and', 'idleness', 'was', 'at', 'present', 'between', 'fanny', 'is']

I tried reducing n to 1 with the original case to see if it would work properly…

Case 5:

Input:

```
words = finish_sentence(
     ['she', 'was', 'not'],
      1,
      corpus,
      True,
    )
```

Output: ['she', 'was', 'not', ',', ',', ',', ',', ',', ',', ',', ',', ',', ',', ',', ',', ',', ',', ',']

Even though this result isn't true English, it makes sense that we would get this as it is not surprising that a comma would be the most common token.

Case 6:

Input:

```
words = finish_sentence(
      ['she', 'was', 'not'],
       1,
       corpus,
       False,
   )
```

Output: [['she', 'was', 'not', 'but', 'soon', 'you', 'after', 'mother', 'music', 'by', 'miss', 'expected', ',', 'with', '.']

The function appears to be as random as I expected for n=1.

Case 7:

Input:

```
words = finish_sentence(
      ['I','think','that'],
       3,
       corpus,
       True,
   )
```

Output: ['I', 'think', 'that', 'he', 'had', 'been', 'in', 'the', 'world', '.']

Case 8:

Input:

```
words = finish_sentence(
      ['I','think','that'],
       3,
       corpus,
       False,
   )
```

Output: ['I', 'think', 'that', 'he', 'meant', 'nothing', 'farther', 'of', 'those', 'fine', 'bold', 'hills', 'that', 'we', 'did']

Next I tried keeping the set of words the same, but upping n to 4. I was pleased to see how much sense the results made with this input.

Case 9:

Input:

```
words = finish_sentence(
     ['I','think','that'],
      4,
      corpus,
      True,
  )
```

Output: ['I', 'think', 'that', 'he', 'had', 'been', 'staying', 'a', 'fortnight', 'with', 'us', '.']

Case 10:

Input:

```
words = finish_sentence(
     ['I','think','that'],
      4,
      corpus,
      False,
  )
```

Output: ['I', 'think', 'that', 'even', 'john', 'and', 'fanny', 'are', 'not', 'entirely', 'without', 'merit', '.']

Finally, I tried another short set of words and set n to 2, so it only had one word to go off of. I found that in the deterministic mode, the result did not make sense, but I got better results in the stochastic mode.

Case 11:

Input:

```
words = finish_sentence(
     ['Erika', 'said'],
      2,
      corpus,
      True,
  )
```

Output: ['Erika', 'said', 'elinor', ',', 'and', 'the', 'same', 'time', ',', 'and', 'the', 'same', 'time', ',', 'and']

Case 12:

Input:

```
words = finish_sentence(
     ['Erika', 'said'],
      2,
      corpus,
      False,
  )
```

Output: ['Erika', 'said', 'elinor', 'was', 'quiet', '.']