

[Return to Classroom](#)[DISCUSS ON STUDENT HUB](#)

Part of Speech Tagging

REVIEW

HISTORY

Meets Specifications

FURTHER READING

Below are links to some material you might find interesting for more insight on the subject matter:

1. [AI in Practice: Identifying Parts of Speech in Python](#)
2. [POS Tagging and Chunking in NLP](#)
3. [POS for Social Media](#)
4. [Parts Of Speech Tagging With HMM and Viterbi Algorithm in NLP](#)
5. [Part-of-Speech Tagging](#)
6. [An introduction to part-of-speech tagging and the Hidden Markov Model](#)

OVERALL COMMENTS

Great work! Congratulations on meeting all requirements of Rubric and your work shows your effort and understanding of concepts 🎉🎉

You did a great job and should be proud of yourself. After reviewing this submission, I am impressed and satisfied with the effort and understanding put in to make this project a success.

All the requirements have been met successfully 100 %

I want you to get the best out of this review and hence I have tried to provide you a holistic experience in this review by adding :-

- Few Suggestions which you can try and improve your python skills.
- Appreciation where you did great
- Some learning opportunities to learn POS/HMM beyond coursework

I hope you find these suggestions informative 😊
Keep doing the great work and all the best for future projects 👍

General Requirements



- Includes `HMM_Tagger.ipynb` displaying output for all executed cells
- Includes `HMM_Tagger.html`, which is an HTML copy of the notebook showing the output from executing all cells

Both notebook and HTML are included with submission 👍



Submitted notebook has made no changes to test case assertions

No changes were made to test case assertions.

Baseline Tagger Implementation



Emission count test case assertions all pass.

- The emission counts dictionary has 12 keys, one for each of the tags in the universal tagset
- "time" is the most common word tagged as a NOUN

emission counts look good 🙌

We could also use `defaultdict` to avoid explicit initialization of dictionary keys. We could then use `Counter` as the default value for `defaultdict`. Next, we could use python `zip` function to create an iterator that could in turn be used to implement the actual counter.

```
def pair_counts(sequences_A, sequences_B):  
    # initialize dictionary with default value set to Counter  
    dictionary = defaultdict(Counter)  
    for i in range(len(sequences_A)):  
        for key, value in zip(sequences_A[i], sequences_B[i]):  
            dictionary[key][value] += 1  
  
    return dictionary
```

```
emission_counts = pair_counts(data.training_set.Y, data.training_set.X)
```



Baseline MFC tagger passes all test case assertions and produces the expected accuracy using the universal tagset.

- >95.5% accuracy on the training sentences
- 93% accuracy the test sentences

MFC tagger accuracy looks good. 🍷

You could use [itertools.chain](#) to merge different tuples of words and sequences

Calculating Tag Counts



All unigram test case assertions pass

Good use of `Counter` to implement `unigram_counts` 🙌

Tag unigrams look good!

Please also take note of this pythonic way of calculating unigram using `chain`. Also, [here](#) is a thread about using

* syntax tokens in a function call.

```
def unigram_counts(sequences):  
    return Counter(chain(*sequences))
```

```
tag_unigrams = unigram_counts(data.training_set.Y)
```



All bigram test case assertions pass

Tag bigrams look good!

Please also take note of the following pythonic way of calculating bigrams:

```
def bigram_counts(sequences):  
    return Counter([pair for sequence in sequences for pair in zip(sequence,  
sequence[1:])])
```

```
tag_bigrams = bigram_counts(data.training_set.Y)
```

```
def bigram counts(sequences):
```

```
counts = Counter()  
counts.update(chain(*(zip(s[:-1], s[1:]) for s in sequences)))  
return counts
```

```
tag_bigrams = bigram_counts(data.training_set.Y)
```



All start and end count test case assertions pass

Well Done, Starting and ending counts are correctly calculated and testcase assertions are passing 🙌

Please also take a note of the following pythonic way of computing start and end counts:

```
def starting_counts(sequences):  
    return Counter(next(zip(*sequences)))
```

```
tag_starts = starting_counts(data.training_set.Y)
```

```
def ending_counts(sequences):  
    reversed_sequences = (reversed(sequence) for sequence in sequences)  
    return starting_counts(reversed_sequences)
```

```
tag_ends = ending_counts(data.training_set.Y)
```

Basic HMM Tagger Implementation



All model topology test case assertions pass

Great job implementing Basic HMM network topology 🙌



Basic HMM tagger passes all assertion test cases and produces the expected accuracy using the universal tagset.

- >97% accuracy on the training sentences
- >95.5% accuracy the test sentences

Great! Accuracy on both training and testing data sets are above threshold 🙌

 [DOWNLOAD PROJECT](#)

[RETURN TO PATH](#)