UDACITY

Logout

‹ Return to "Natural Language Processing Nanodegree" in the classroom

Part of Speech Tagging

REVIEW HISTORY

Meets Specifications

Sensational Learner,
Congratulations!👏
The hard work put into the project is commendable. The work shows that you have a good understanding of the concepts covered in the classroom. Continue with this spirit of hard work in all that you do and nothing will ever stop you from accomplishing your dreams. It was my pleasure reviewing this project. Have a splendid day and good luck with the next project.

Recommendations.

Here are some resources that can give you an even deeper understanding of the project and for further learning.

- Part of Speech Tagging;
- Part-Of-Speech Tagging for Social Media Texts;
- AI in Practice: Identifying Parts of Speech in Python:
- Learning POS Tagging & Chunking in NLP.

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**

You did well by running all the code cells in the submitted `HMM Tagger.ipynb` notebook and it's HTML copy.✅

✓ **Submitted notebook has made no changes to test case assertions**

Good job avoiding changes to the test case assertions. They are meant to evaluate the code in their respective sections of the project and they let you know when the code is correct or wrong.

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**

The emission count dictionary has 12 keys and `time` is tagged as a NOUN. This is excellent!✅

Pro Tips

In order to avoid explicit initialization of dictionary keys, we can use python's defaultdict. Check this out as well.

✓ **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**

Training and Testing accuracy were both perfectly obtained. MFC tagger passes all test case assertions and produces the expected accuracy using the universal tagset.

```
training accuracy mfc_model: 95.72%
testing accuracy mfc_model: 93.02%
```

Pro Tips

itertools.chain can also be used here to merge tuples of words and sentences.

Calculating Tag Counts

✓ **All unigram test case assertions pass**

Unigram test case assertions passed. Well done!

Pro Tips

Here is another pythonic approach to calculating the unigrams.

```
def unigram_counts(sequences):

    for tag in sequences:
        if tag in dict_tag.keys():
            dict_tag[tag] += 1
        else: dict_tag[tag] = 1

    return dict_tag

tag_unigrams = unigram_counts(tags)
```

✓ **All bigram test case assertions pass**

All the bigram test assertions passed. Good job with the implementation!👍

Pro Tips

Here is another pythonic approach to calculating the bigrams.

```
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**

All the start and end count test case assertions passed as expected. ✔

Basic HMM Tagger Implementation

✓ **All model topology test case assertions pass**

Good job implementing the correct model topology for the Basic HMM Tagger. All its test case assertions passed to confirm its correctness. Well done!👏

✓ **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**

The Basic HMM tagger successfully was correctly implemented to get impressive accuracies on the training and testing sentences. Excellent results!👏

```
training accuracy basic hmm model: 97.53%
testing accuracy basic hmm model: 95.83%
```

DOWNLOAD PROJECT

RETURN TO PATH

Rate this project
☆☆☆☆☆