Lenning_JM8

October 21, 2021

1 Chapter 8

1.1 1

1. I/PRP need/VBP a/DT flight/NN from/IN Atlanta/NN Atlanta should be a NNP because it is a proper noun
2. Does/VBZ this/DT flight/NN serve/VB dinner/NNS Diiner should be a NN because it is singular
3. I/PRP have/VB a/DT friend/NN living/VBG in/IN Denver/NNP Have should be a VBP because has is the base
4. Can/VBP you/PRP list/VB the/DT nonstop/JJ afternoon/NN flights/NNS Afternoon should be a JJ

1.2 2

Use the Penn Treebank tagset to tag each word in the following sentences from Damon Runyon's short stories. You may ignore punctuation. Some of these are quite difficult; do your best.

[ ]:
```
import nltk
nltk.download('averaged_perceptron_tagger')
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/alenning/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

[ ]:
```
text1 = word_tokenize("It is a nice night")
text2 = word_tokenize("This crap game is over a garage in Fifty-second Street")
text3 = word_tokenize("Nobody ever takes the newspapers she sells")
text4 = word_tokenize("He is a tall skinny guy with a long sad mean-looking␣
 ↪kisser and a mournful voice")
text5 = word_tokenize("I am sitting in Mindys restaurant putting on the␣
 ↪gefillte fish which is a dish I am very fond of")
text6 = word_tokenize("When a guy and a doll get to taking peeks back and forth␣
 ↪at each other why there you are indeed")
```

```
print(nltk.pos_tag(text1))
print(nltk.pos_tag(text2))
print(nltk.pos_tag(text3))
print(nltk.pos_tag(text4))
print(nltk.pos_tag(text5))
print(nltk.pos_tag(text6))
```

```
[('It', 'PRP'), ('is', 'VBZ'), ('a', 'DT'), ('nice', 'JJ'), ('night', 'NN')]
[('This', 'DT'), ('crap', 'NN'), ('game', 'NN'), ('is', 'VBZ'), ('over', 'RP'),
('a', 'DT'), ('garage', 'NN'), ('in', 'IN'), ('Fifty-second', 'NNP'), ('Street',
'NNP')]
[('Nobody', 'NN'), ('ever', 'RB'), ('takes', 'VBZ'), ('the', 'DT'),
('newspapers', 'NNS'), ('she', 'PRP'), ('sells', 'VBZ')]
[('He', 'PRP'), ('is', 'VBZ'), ('a', 'DT'), ('tall', 'JJ'), ('skinny', 'NN'),
('guy', 'NN'), ('with', 'IN'), ('a', 'DT'), ('long', 'JJ'), ('sad', 'JJ'),
('mean-looking', 'NN'), ('kisser', 'NN'), ('and', 'CC'), ('a', 'DT'),
('mournful', 'JJ'), ('voice', 'NN')]
[('I', 'PRP'), ('am', 'VBP'), ('sitting', 'VBG'), ('in', 'IN'), ('Mindys',
'NNP'), ('restaurant', 'NN'), ('putting', 'VBG'), ('on', 'IN'), ('the', 'DT'),
('gefillte', 'NN'), ('fish', 'NN'), ('which', 'WDT'), ('is', 'VBZ'), ('a',
'DT'), ('dish', 'JJ'), ('I', 'PRP'), ('am', 'VBP'), ('very', 'RB'), ('fond',
'NN'), ('of', 'IN')]
[('When', 'WRB'), ('a', 'DT'), ('guy', 'NN'), ('and', 'CC'), ('a', 'DT'),
('doll', 'NN'), ('get', 'NN'), ('to', 'TO'), ('taking', 'VBG'), ('peeks',
'NNS'), ('back', 'RB'), ('and', 'CC'), ('forth', 'NN'), ('at', 'IN'), ('each',
'DT'), ('other', 'JJ'), ('why', 'WRB'), ('there', 'EX'), ('you', 'PRP'), ('are',
'VBP'), ('indeed', 'RB')]
```

1.3 3

Now compare your tags from the previous exercise with one or two friend's answers. On which words did you disagree the most? Why?

Lowry and Ammon both categorized these by hand. It was interesting to compare them to NLTK's answers. It seemed that NLTK mixed up NN and JJ much more frequently than both Lowry and Ammon. For example 'fond' in sentence 5 was categorized by NLTK as a NN but Lowry and Ammon both used JJ.

1.4 4

Implement the "most likely tag" baseline. Find a POS-tagged training set, and use it to compute for each word the tag that maximizes p(t|w). You will need to implement a simple tokenizer to deal with sentence boundaries. Start by assuming that all unknown words are NN and compute your error rate on known and unknown words. Now write at least five rules to do a better job of tagging unknown words, and show the difference in error rates.

[ ]:
```
from nltk.corpus import brown
from sklearn.model_selection import train_test_split

brown_news_tagged = brown.tagged_words(categories='news')
brown_train, brown_test = train_test_split(brown_news_tagged, train_size=0.8)

def baseline(document):
    most_common = set()

    tag_fd = nltk.ConditionalFreqDist((word.lower(), tag) for (word, tag) in␣
 ↪document)

    for word in tag_fd.conditions():
        tag = tag_fd[word].max()
        most_common.add((word, tag))

    return list(most_common)

base = baseline(brown_train)
```

[ ]:
```
import pandas as pd

def get_acc(train, test):
    train = pd.DataFrame(train, columns=['word', 'tag_pred'])
    test = pd.DataFrame(test, columns=['word', 'tag_act'])
    merged = test.merge(train, left_on="word", right_on="word")
    merged['score'] = (merged['tag_act'] == merged['tag_pred'])

    return 1 - len(merged[merged['score'] == False]) / len(merged)


print("correctly assigned tags:", get_acc(base, brown_test))
```

```
correctly assigned tags: 0.9097648261758691
```

1.5 5

Build a bigram HMM tagger. You will need a part-of-speech-tagged corpus. First split the corpus into a training set and test set. From the labeled training set, train the transition and observation probabilities of the HMM tagger directly on the hand-tagged data. Then implement the Viterbi algorithm so you can decode a test sentence. Now run your algorithm on the test set. Report its error rate and compare its performance to the most frequent tag baseline.

[ ]:
```

```

1.6 6

Do an error analysis of your tagger. Build a confusion matrix and investigate the most frequent errors. Propose some features for improving the performance of your tagger on these errors.