María José Vilella Sánchez

Practical exercise: POS tagging

You will need to turn in two documents: the completed pos_exercise.py and a word document with the answers to the questions here.

Point distribution:

Part 1: 2.5 points

Part 2: 2.5 points

Part 3: 5 points

PART 1.: TRIGRAM TAGGER WITH BACKOFF

You will need to create train/dev/test splits for each domain in the brown corpus (be sure to use tagset="universal" when you call brown.tagged_sents). Afterwards, you will train a trigram tagger which backs off to a bigram tagger which backs off to a unigram tagger which finally backs off to a default tagger with "NOUN" (see NLTK book: Chapter 5.4 for details).
For each domain test the tagger on the dev and test data.

1. Which domain has the best test accuracy? Which on has the worst? Why do you think this is the case?

The domain with the best test accuracy: learned (Test Accuracy for learned domain: 92.42341022824363)

The domain with the worst test accuracy: humour (Test Accuracy for humor domain: 83.55899419729207)

Perhaps this is due to the difference between the length of the corpus of each domain. Since learned is the largest and humor is the smallest, as can be seen below:

```
Train len for adventure domain:  3245
Dev len for adventure domain:  464
Test len for adventure domain:  928
Train len for belles_lettres domain:  5046
Dev len for belles_lettres domain:  721
Test len for belles_lettres domain:  1442
Train len for editorial domain:  2097
Dev len for editorial domain:  300
Test len for editorial domain:  600
Train len for fiction domain:  2974
Dev len for fiction domain:  425
Test len for fiction domain:  850
Train len for government domain:  2122
```

```
Dev len for government domain:  303
Test len for government domain:  607
Train len for hobbies domain:  2935
Dev len for hobbies domain:  419
Test len for hobbies domain:  839
Train len for humor domain:  737
Dev len for humor domain:  105
Test len for humor domain:  211
Train len for learned domain:  5413
Dev len for learned domain:  774
Test len for learned domain:  1547
Train len for lore domain:  3416
Dev len for lore domain:  488
Test len for lore domain:  977
Train len for mystery domain:  2720
Dev len for mystery domain:  388
Test len for mystery domain:  778
Train len for news domain:  3236
Dev len for news domain:  462
Test len for news domain:  925
Train len for religion domain:  1201
Dev len for religion domain:  171
Test len for religion domain:  344
Train len for reviews domain:  1225
Dev len for reviews domain:  175
Test len for reviews domain:  351
Train len for romance domain:  3101
Dev len for romance domain:  443
Test len for romance domain:  887
Train len for science_fiction domain:  663
Dev len for science_fiction domain:  95
Test len for science_fiction domain:  190
```

2. Which domain has the largest difference between dev and test? Why do you think this might be?

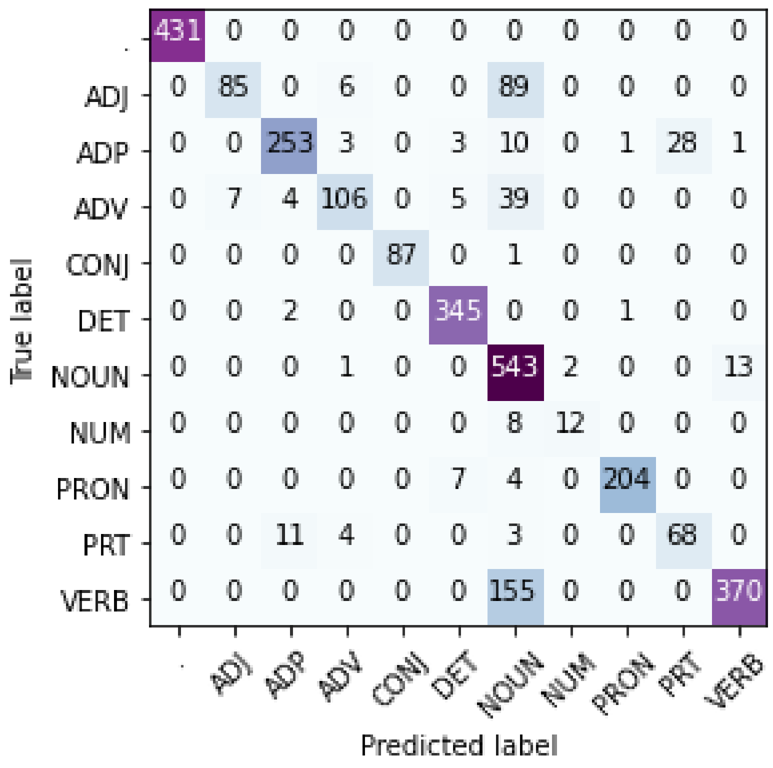
The domain which shows the biggest different between the dev and the test is “science_fiction” (dev: 95 and test: 190). My conclusion is that the difference in the length of the corpus of each domain is the principal reason for these results.

PART 2: CONFUSION MATRIX

Next, you will use the trigram tagger you trained for the final domain and collect the predictions on the test set using tagger.tag_sents(). Note that the tagger expects UNTAGGED sentences, so you will need to remove the tags from the test set. Now you will create a confusion matrix using the code from sklearn.metrics.confusion_matrix.

This function takes two arguments: 1) the true labels and 2) the predicted labels. Note that these should be flat lists containing only the tags, i.e., test_labels = ['ADP', 'DET', 'NOUN',…]. Therefore, you will need to manipulate both the test and predictions to get them into the correct format. Finally, use plot_confusion_matrix() to plot the errors.

We can appreciate the confusion matrix below:



True label \ Predicted label	.	ADJ	ADP	ADV	CONJ	DET	NOUN	NUM	PRON	PRT	VERB
.	431	0	0	0	0	0	0	0	0	0	0
ADJ	0	85	0	6	0	0	89	0	0	0	0
ADP	0	0	253	3	0	3	10	0	1	28	1
ADV	0	7	4	106	0	5	39	0	0	0	0
CONJ	0	0	0	0	87	0	1	0	0	0	0
DET	0	0	2	0	0	345	0	0	1	0	0
NOUN	0	0	0	1	0	0	543	2	0	0	13
NUM	0	0	0	0	0	0	8	12	0	0	0
PRON	0	0	0	0	0	7	4	0	204	0	0
PRT	0	0	11	4	0	0	3	0	0	68	0
VERB	0	0	0	0	0	0	155	0	0	0	370

1. Which labels are most often predicted correctly? Why do you think this could be?

The labels most predicted correctly are nouns (543), dots (431) and verbs (370). As syntactic functions are essential in every sentence, this seems to be the reason for the result of this experiment.

2. What are the most common errors? Why do you think this is?

The most common errors are that both verbs and adjectives are wrongly classified as nouns. In the sphere of the wrongly classified verbs (as nouns) may be due to the fact that in English some verbs and nouns have the same written form, for example “balance” or “fight”. Moreover, we can see this confusion with the -ing form, that is *running* could be either a verb in a continuous form or a noun.
As for adjectives classified as nouns, this may be due to the fact that there are adjectives that are used as nouns in sentences, such as “the dead” or “the rich”.

PART 3:

In this section, you will need to choose ONE of the following: a) implementing viterbi in a simple HMM-based tagger or performing in-depth error analysis on the previous results.

a) **Complete the code in simple_hmm.py**

b) Error analysis:

1. Based on the code already available in pos_exercise.py, create a function to find sentences with errors. Use this function to collect all the sentences with errors in the dev set of the science_fiction domain of the brown corpus.

2. Now, you will need to categorize what type of errors are found.
   - You can use the following information to help you:
     - a) whether the word is found in the training set or not
     - b) the relative frequency of the word in the training set
     - c) the number of tags associated with the word in train
     - etc.
3. Hypothesize what phenomena might be difficult for the tagger.