HAP/LAP Master
Euskal Herriko Unibersitatea
Facultad de Informática

PoS Tagging Exersice

January 31, 2022

Student: David Cabestany Manen
Course: Computational Syntax
dcabestany001@ikasle.ehu.eus

Contents

List of Tables

List of Figures

Abstract

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

1 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?

The domain with the best accuracy is "Learned" with an accuracy of 0.92 score.

Which on has the worst?

The domain with the worst accuracy is "Humor" with an accuracy of 0.83 score.

Why do you think this is the case?

I think that one explanation for that is the size of the corpora. Learned is the bigger domain. Humor is not the smaller but is a tricky domain, because humor uses the language in a pragmatic way with a lot of irony and funny collocations.

2. Which domain has the largest difference between dev and test?

The domain with the biggest gap between dev and test is "Science-fiction" with a score of 0.05.

Why do you think this might be?

This is the smallest corpus. I think this might be the point.

2 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 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.

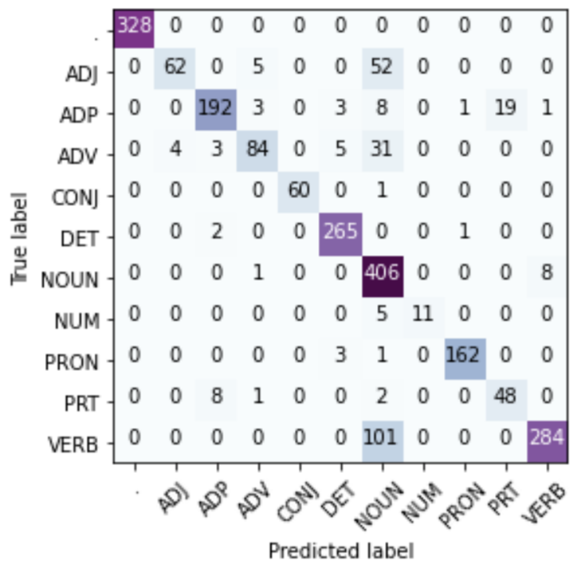

Figure 1: Confusion Matrix

1. Which labels are most often predicted correctly?

As we can see in Figure 1, the most frequent correct predictions are Nouns, Pronouns, and Verbs.

Why do you think this could be?

Those tags are the most frequent in language and normally the most rigid collocations in English.

2. What are the most common errors?

Nouns predicted as Verbs.

Why do you think this is?

Because the verbs can be used as nouns, also.

3 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.

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.*

3. *Hypothesize what phenomena might be difficult for the tagger.*

The errors found, as we can see in Figure 1 are mainly confusions among nouns and other categories, that is due to the fact that some verbs works as nouns, and some adjectives works as well as nouns. Some examples to illustrate that verbs are used as verbs are gerunds; and examples of nouns working as adjectives are those nouns followed by another noun, that is "ticket office" for give an example.