CE314 Assignment 2
Text Classification and POS Tagging

Annie Louis

20 November, 2016

Introduction

In this assignment, you will build a Naive Bayes classifier and use it to do topic classification. You will also solve some POS tagging exercises using NLTK.

- Read the **What to submit** section before you submit your work.
- Carefully read the instructions and details about the data.
- You can write the code in any programming language you are comfortable with.
- For the report, be concise. You **do not need to write more than 2 to 5 sentences** for each answer.
- This assignment is worth 20% of your module marks

Important note on plagiarism

The assignment should be completed **individually**. Collaboration is not allowed. You may discuss the lecture material and concepts, and clarify the questions asked in the assignment with other students. But you may not discuss the solutions or approaches. Regarding code, it is okay to help someone with a language's syntax or library functions but you may not give them the actual program statements or show them your code.

Carefully read the guidelines here. `http://www.essex.ac.uk/ldev/resources/plagiarism/default.aspx`

All plagiarism cases will be referred to an academic offenses procedure. No allowance regardless of whether the act was intended or unintended.

IMPORTANT: For Q1 to Q3, you only need to write the answers in your report. You do not need to submit the code.

You may find it useful to go over Lab03 before doing this assignment. Questions Q1-Q3 should be completed using NLTK.

The meaning of the Brown POS tags can be referred to either in NLTK (using the commands introduced in lab) or at this link `https://en.wikipedia.org/wiki/Brown_Corpus#Part-of-speech_tags_used`.

Q1. Training POS taggers on the Brown corpus (15 marks)

Use the first 10,000 tagged **news domain** sentences from the Brown corpus as training data. Build a UnigramTagger and a HMMTagger on this data using NLTK libraries. Restrict training sentences to the **news domain** by using the *categories* argument while accessing the Brown corpus.

Tag the following sentence with each tagger.

S1. *I walked 5 miles.*

a) Copy and paste the output POS-tagged sentence from the two models (10 marks)

b) Did the models produce the right POS tags? (5 marks)

Q2. Comparing taggers (25 marks)

Consider the sentence:

S1. *The prices will drop in the short term.*
S2. *The extent of the drop was very great.*

a) What is the correct POS for the word "drop" in S1 and S2? (You can write your answer using a general word such as adjective, noun etc, no need to write an exact tag from a tagset). (5 marks)

b) Run the UnigramTagger from NLTK on the two sentences. What is the output of the tagger? (Copy and paste only the sentence with the tags). Does the UnigramTagger produce the right POS for the word "drop" in S1 and S2. Explain why the tagger is correct or incorrect. (10 marks)

c) Run the HMMTagger from NLTK on the two sentences and give the output. Is the HMMTagger correct on the word "drop" in S1 and S2? Explain why the tagger is correct or incorrect. (10 marks)

Q3. POS Ambiguity (20 marks)

S1. *Eye drops off shelf.*
S2. *Squad helps dog bite victim.*

a) Do S1 and S2 have more than one interpretation? If so, briefly describe the interpretations. (10 marks)

b) Run the UnigramTagger on these sentences? Did the tagger resolve the ambiguity if any for S1 and S2? (10 marks)

Q4. Text classification (40 marks)

Write a computer program that performs Naive Bayes Classification.

Data: Download the data `a02_data_ug.zip`. The folder has 3 files: `sampleTrain.txt`, `sampleTrain.vocab.txt` and `sampleTest.txt`.

`sampleTrain.txt` is the training data for building a classifier. The vocabulary of the training data is in `sampleTrain_vocab.txt`. The classifier should be finally run and evaluated on test data in `sampleTest.txt`. The second column in the `sampleTrain.txt` and `sampleTest.txt` files gives the gold standard true class for each document. The first column of these files is the document id, the third column gives the words in the document. The columns are separated by tab spaces.

There are 2 classes in the data 0 and 1.

Task: Build a Naive Bayes classifier using the document words as features. It should compute a model given some training data and be able to predict classes on a new test set. For this assignment, use `sampleTrain.txt` for training a model and the model should be used to predict classes for documents in `sampleTest.txt`. Use Laplace smoothing for feature likelihoods. There is **no** need for UNK token. The dataset has been simplified so that the test corpus only contains words seen during training (so no need for UNK). There is also no need to smooth the prior probabilities.

The program when run should print the following:

```
Prior probabilities
class 0 =
class 1 =


Feature likelihoods
          great   sad   boring   ...
 class 0
 class 1

Predictions on test data
d5 =
d6 =
d7 =
d8 =
d9 =
d10 =

Accuracy on test data =
```

The features in the feature likelihood table (great, sad, boring,...) can be printed in any order.

Copy and paste the output into your report as the answer to Q4.

What to submit: Report and Code

A single `registration_number.zip` file. The uncompressed folder should contain a report and a code directory.

IMPORTANT: Write your name and registration number in the report and also as comments in your code.

Report. A file containing the answers for Q1, Q2, Q3, Q4. For Q4, paste the output of your code into the report.

Code. For Q4 only. Your code should run without any arguments. It should read files in the same directory. Absolute paths must not be used. When downloaded, you code should run with a simple command such as `java NaiveBayes` or `python NaiveBayes.py`. A README.txt file should have a single line giving the command to run your code. Check your code by downloading your .zip file into a different machine and testing that it runs without modification. When your code is run, it should print values in the format specified in Q4. Negative marks will be applied if code does not run out of the box.

Assessment criteria

What we are looking for in your answers:

Clear understanding of the concepts demonstrated by taking the right approach, correct substitution and accurate answers

Ability to use concepts learned in class demonstrated by clear answers to questions which ask to analyze numbers and output

Delivering the requested software solutions demonstrated by code which satisfies the criteria, outputs the required solutions cleanly, runs without dependencies, contains proper comments. You will not be evaluated on the efficiency of your code or algorithms as long as they run in reasonable time.