Homework 3 – Naïve Bayes Classifier

- Submitted by Pragya Shah (SC15B107)

Problem Statement: Implement a Naïve Bayes classifier using the given reference model.

$$P(k,d) = p(k) \times \prod_{w \in d} p(w \mid k)$$

$$P(k \mid d) = \frac{p(k,d)}{\sum_{k'} p(k',d)}$$

a. Write code to read in the training documents and collect counts $c(k)$ (number of documents in class k) and $c(k,w)$ for all speakers k and all words w.
   Report $c(k)$ and $c(k,w)$ for $k \in \{clinton, trump\}$ and $w \in \{country, president\}$
   Write code to compute the probabilities $p(k)$ and $p(w|k)$ for all k,w. Train on the training set. Report $p(k)$ and $p(w|k)$ for $k \in \{clinton, trump\}$ and $w \in \{country, president\}$
b. Write code to read in a test document and compute the probabilities $P(k|d)$ for each k. Report these probabilities for the first line of `dev`. Note that these probabilities should sum to one! Avoid underflow.
c. Run the classifier on the test set and report your accuracy, which should be at least 50%.
d. Describe any implementation choices you had to make (e.g., smoothing, log-probabilities).

Solution:

The code consists of several modules which perform different tasks. The function **training()** takes the test files which constitute the corpus and counts up different features of it like 'docs' (total number of documents in the corpus) etc. Both the dev and train files have been combined to form a larger corpus.

The function **compute_p()** computes the various probabilities like p(k) etc. Laplace's add – 1 smoothing has been used to prevent the probabilities to go to zero if a word not spoken by the speaker in corpus is met with in the test case

The function **testsample()** takes 1 document and evaluates the probabilities p(k,d) and p(k/d) for each speaker w.r.t that document. Then it takes the argmax of p(k/d) and compares it with the original speaker of the document. If they match, it returns a score 1 else a 0.

While evaluating p(k,d) we may come across underflow, which is taken care by multiplying each p(w/k) with 1000. This, however, will not affect the final result of the function.

The condition that sum of all p(k/d) for a particular document must be 1 is verified by testsample()

Note: the program considers words of length 1 to be stop words. Hence it ignores them

The function **finaltest()** sends the documents of the test file, one by one for evaluation to the testsample() function. It also maintains a record of the overall score obtained by the classifier for the test file and calculates its accuracy.

Function **main()** is used to call the various functions making up the classifier and printing the desired outputs as described in the problem statement.

The last part of the program is the boilerplate code which gives us the flexibility to give paths to any other test/dev/training files we might want to use from the command prompt.

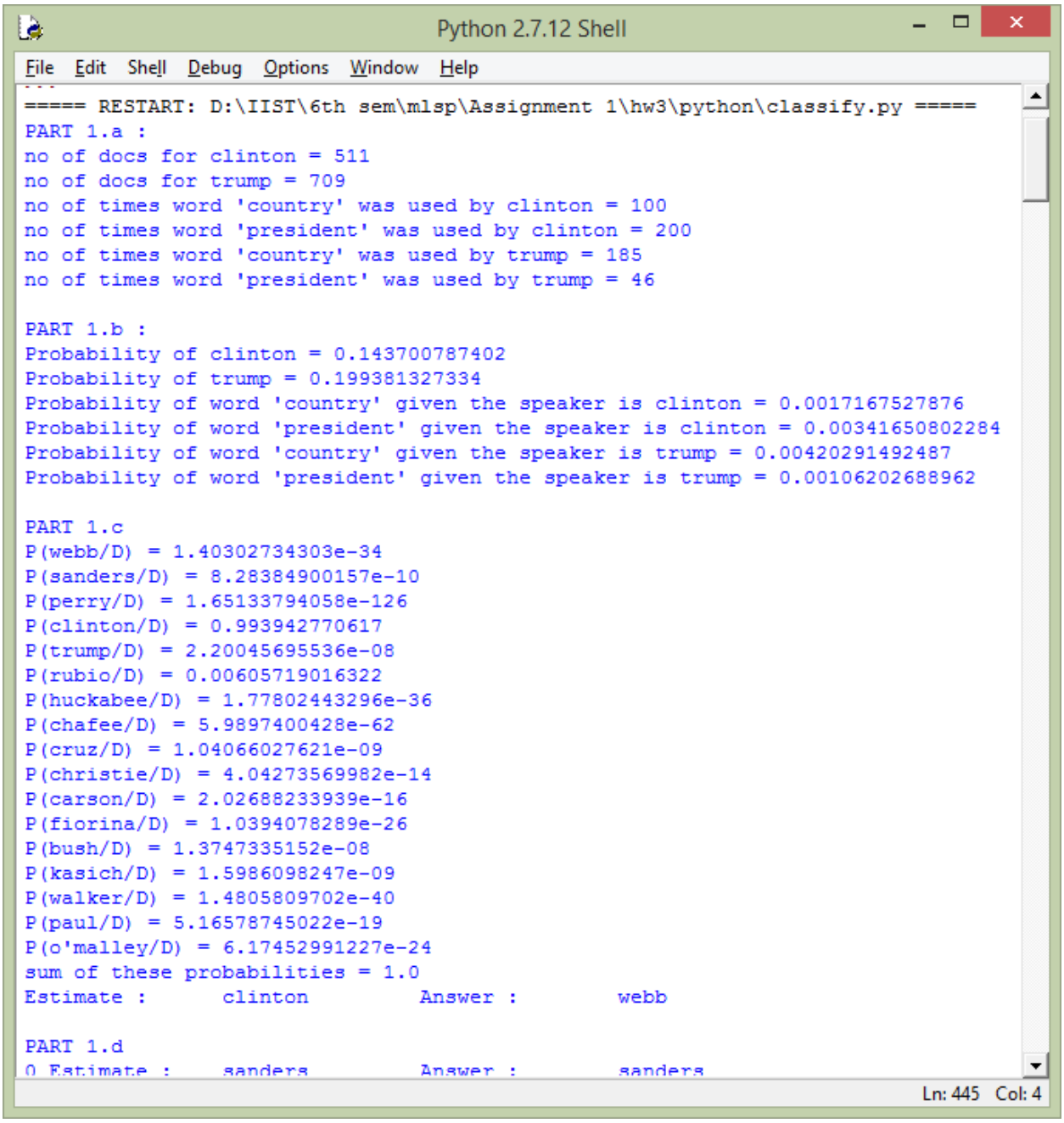
Source code:

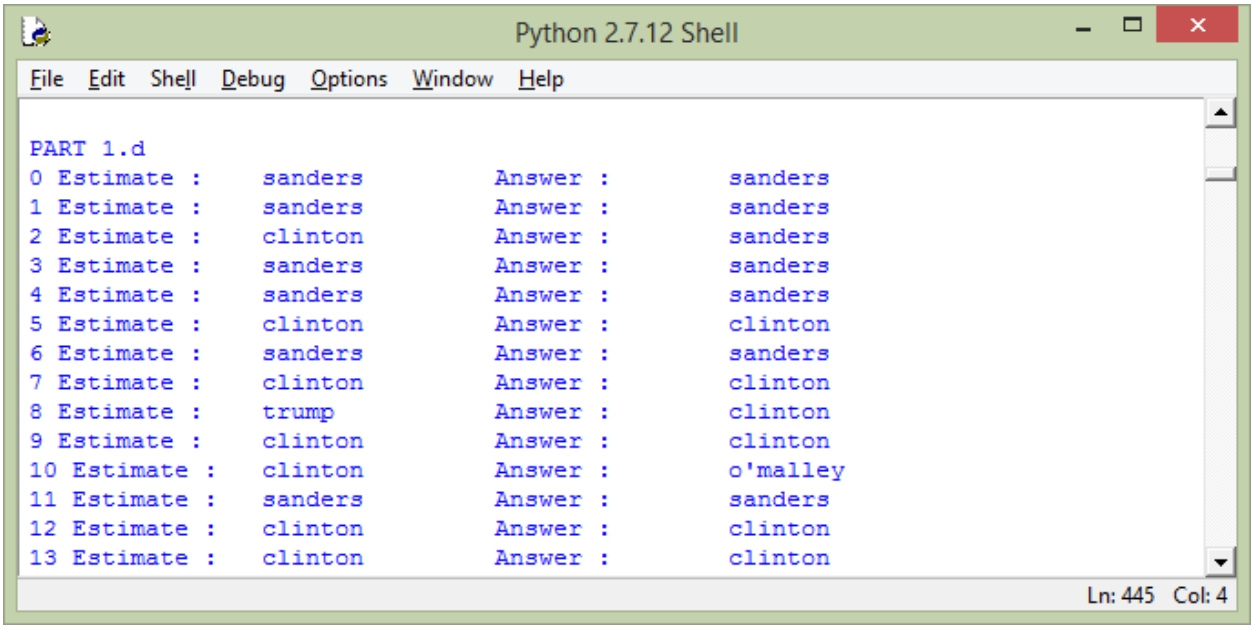
Look at classify.py in the python directory.

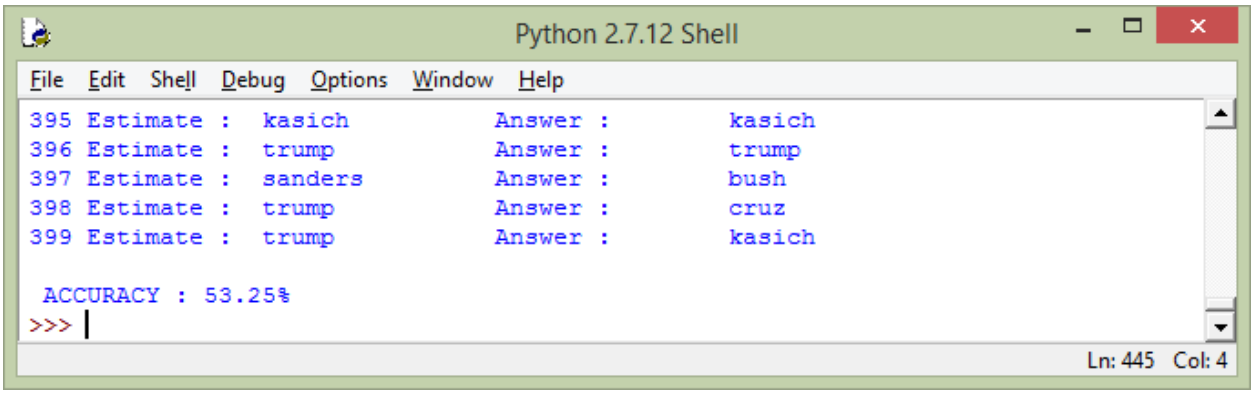
NOTE: The directory “python” contains –

The code uses Python 2.7.12 and might not be compatible with Python 3

Output:


Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
===== RESTART: D:\IIST\6th sem\mlsp\Assignment 1\hw3\python\classify.py =====
PART 1.a :
no of docs for clinton = 511
no of docs for trump = 709
no of times word 'country' was used by clinton = 100
no of times word 'president' was used by clinton = 200
no of times word 'country' was used by trump = 185
no of times word 'president' was used by trump = 46
PART 1.b :
Probability of clinton = 0.143700787402
Probability of trump = 0.199381327334
Probability of word 'country' given the speaker is clinton = 0.0017167527876
Probability of word 'president' given the speaker is clinton = 0.00341650802284
Probability of word 'country' given the speaker is trump = 0.00420291492487
Probability of word 'president' given the speaker is trump = 0.00106202688962
PART 1.c
P(webb/D) = 1.40302734303e-34
P(sanders/D) = 8.28384900157e-10
P(perry/D) = 1.65133794058e-126
P(clinton/D) = 0.993942770617
P(trump/D) = 2.20045695536e-08
P(rubio/D) = 0.00605719016322
P(huckabee/D) = 1.77802443296e-36
P(chafee/D) = 5.9897400428e-62
P(cruz/D) = 1.04066027621e-09
P(christie/D) = 4.04273569982e-14
P(carson/D) = 2.02688233939e-16
P(fiorina/D) = 1.0394078289e-26
P(bush/D) = 1.3747335152e-08
P(kasich/D) = 1.5986098247e-09
P(walker/D) = 1.4805809702e-40
P(paul/D) = 5.16578745022e-19
P(o'malley/D) = 6.17452991227e-24
sum of these probabilities = 1.0
Estimate : clinton Answer : webb
PART 1.d
0 Estimate : sanders Answer : sanders
Ln: 445 Col: 4





Summary:

The outputs of parts 1.a through 1.c can be seen above.

% Accuracy = 53.25%

Part 1.d – The implementation choices

- Smoothing – Laplace's add – 1 smoothing
- Avoiding underflow of p(k,d) - While evaluating p(k,d) we multiplied each p(w/k) with 1000.
- Stop words – all words of length 1 (eg : '.' ',' 'a' 'i' '1' '2')

Inference:

Given were 17 speakers. Had we taken a random guess, for example a classifier that always says that the document belongs to just 1 speaker, the accuracy could have been expected to be about 5.88%.

This classifier proves to be about 10 times better than a random guess.

Still there is scope for improvement in this very model of classifier.

Considering the way a human would do the above classification:

1. He reads the corpus, going through the statements made by various speakers
2. Each speaker, in a debate/campaign, speaks about a few particular themes/topics/ideas he believes in. The human classifier has the ability to identify such ideas/themes for each speaker.
3. When given a new statement and asked to classify, the human classifier analyzes the key ideas of the statement and compares it to those of the speakers. Considering this he then decides on the person who is most probable to make that statement
4. Thus, all the words spoken do not have the same priority.

As all AIs are built up taking inspiration from human behavior, it would be a good idea to model the above behavior in this classifier in order to improve its accuracy. This can be done using the tf-dtf concept.

Through this assignment, I came to know that though AI can be intimidating, it really boils down to skillful and practical use of probability. Also the need to follow a modular approach for writing large complex programs to avoid errors became more obvious. There is always scope of improving algorithms; all we need is to look at the problems from a different angle.