FOUNDATIONS OF ARTIFICIAL INTELLIGENCE
LAB - 3

Done by : DHRUV SHARMA
ds7042@rit.edu

This is the writeup for the lab 3 submission, instructions and documentation is in the README.txt file submitted.

1) a description of your features and how you chose them

   I choose the following features :
   Presence of ij , double vowels (2 repeating vowels) and dutch diphthongs : These 3 features I got based on my research of the Dutch language.

   2 features representing the presence of dutch and english frequent words : words selected from wikipedia's random article generator manually

   2 features representing the presence of Dutch and English stop words : words got from nltk's function (code for download in script.py for reference). I hardcoded the output as sets.

2) a description of the decision tree learning, how you came up with the best parameters (max depth, etc.) and your own testing results

   The decision tree learning algorithm gets the maximal information gain of the attributes that can be used as split attributes as a particular node. Based on various base cases, one of them being max_depth , we have leaf nodes that are created and have a particular classification. The learning code then serializes this tree to be used.

   I used my examples (about 2500) in a ratio of 90 : 10 train : test , to check which depth is the best. And I observed no significant improvement after a max_depth of 8. Which I have set in my global variable MAX_TREE_DEPTH in my train code.

   I used pandas to efficiently manipulate the examples and make my debugging tasks easier. As referenced previously, since I did not want the files to be dependent on downloading nltk, I ran the script separately and hardcoded the output to be used as examples by both the training methods.

3) a description of the boosting, how many trees turned out to be useful, and your own testing

   For boosting , I used max_depth = 1 , trees and used their misclassifications to weigh my test cases properly and continue the process. The example and ratio of train : test was the same as the previous one , and I used the concept that an error of > 50%

means the weak learning model is weaker than random guessing, so adaptive boosting is of no use after this point. So the code has that break condition. Based on my examples and what ultimately I have submitted as my best model, I used an adaboost ensemble with DECISION_STUMPS (This can also be controlled by changing this global variable) = 8. In my training process greater than 8 stumps made the error > 50, hence I choose this.

4) anything else you think we might like to know

   The program running method is in README.txt
   I have included reference code in script.py to get stop words
   And data_collection.py as a web - scraping code that was used to get examples.
   However I wrote it in a file example in the format required