LE GUEN Eflamm

Write-up : Wikipedia Language Classification

Introduction :

The goal of this project is to classify two languages by using either a Decision Tree algorithm or Adaboost algorithm (boosted decision stump). In our use case we classify "English" and "Dutch" and we provide the dataset for the training and for the test step. We can also classify other languages by changing the data and the features of the sentence class.

Decision tree Algorithm :

We first use a decision tree to classify the data. We also use the metric of information gain to split our data by features and build our decision tree. We use the entropy in order to measure the disorder in our dataset and then compute our information gain. We can change the depth of the decision tree, it's assigned to 10 (10 features) but we can decrease it and the error is still acceptable (until 4).

Adaboost :

We create a set of boosted stumps (decision tree of depth equals to 1) in order to classify the data. Each sentence of the dataset has a weight. We adjust this weight with the result of the classification, the sum of the weight has to be equals to 1 because we have to form a distribution. We also boost each stump with a weight by using the amount of say. The size of the hypothesis set can change and with a size of 6 I have the best results.

Dataset :

We have a train set and a set in order to run our classifier. The train set is composed of 111 sentences with 56 in English and 55 in Dutch each sentence is labeled with either 'dt' for Dutch or 'en' for English. Each sentence is picked up from wikipedia using random articles.

The test set is composed of 50 sentences with 25 in Dutch and 25 in English. Each sentence is written without a label and the program will display the prediction of each sentence line by line ('en' for English or 'dt' for Dutch).

Utilisation :

python sort.py train <examples> <hypothesisOut> <learning-type>

- **<examples>** is a file containing labeled examples.
- **<hypothesisOut>** specifies the filename to write your model to.
- **<learning-type>** specifies the type of learning algorithm you will run, it is either "dt" or "ada".

python sort.py predict <hypothesis> <file>

- **<hypothesis> :** is a trained decision tree or ensemble created by your train program
- **<file>** : is a file containing lines of 15 word sentence fragments in either English or Dutch.

Features choice :

In order to classify our data and to compute entropy we selected binaries features. The features will depend on the language you want to classify. In our case, I choose to see which words are more frequently used in each language. Secondly, I use the function endswith() in order to differentiate data with frequent suffixes in each language.

- **Has-XXX :**

These features will indicate us if we have the word XXX in the sentence and we will put it in a dictionary ({“Has-XXX” : True} for example). I tried to find frequent words in both languages but also not frequent in the two languages.

For the **English** features I picked : **“has-the”, “has-be”, “has-to”** and **“has-of”** which are the 4 most frequent words in English.

For the **Dutch** features I picked : **“has-het”, “has-een”, “has-en”** and **“has-de”** which are the 4 most frequent words in English.

- **Ends-with-XX:**

These features will indicate us if we have the letters XX in the sentence and we will store it in a dictionary as before.

For the **English** language I picked : **“ends-with-ed”** for the conjugation.

For the Dutch language I picked : **“ends-with-en”** which is one of the most frequent suffixes used in the Dutch language. This suffix is also used in English so when I debugged the code It never really appeared after the information gain selection.

PS : I have 0 error of prediction when I test (on my test set) my program using “dt” and I have 1 error when I test it with “ada”. But when I decrease the number of features the error rate increases with “dt” and secondly I achieve boosting it and decrease the error rate with “ada”.