CSCI 630

LAB 3

Wikipedia Language Classification

Abhishek Shah, as5553

FILES INCLUDED

Following files are included in the zipped folder.

driver.py – This is the driver code for the program. This is where the main() is.

adaBoost.py – adaBoost implementation for classification.

decisionTree.py – Decision Tree implementation for classification.

checkFeatures.py – Contains features described and used for splitting the data.

treeNode.py – Contains class tree. Useful for recursing and for keeping track of left and right child.

dtModel.obj – Pretrained Decision tree model (depth: 1)

adaModel.obj – Pretrained adaBoost model (totalDecisionStumps: 70)

train.dat – training data for both the models.

test.dat – testing data for both the models.

RUNNING THE PROGRAM

The program has **two entry points**

For training

python3 driver.py train <examples> <hypothesisOut> <learning-type(dt or ada)>

- **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.*
  Either "dt" or "ada".

For testing
python3 driver.py predict <hypothesis> <file> <testing-type(dt or ada)>

- **hypothesis** is a trained decision tree or ensemble created by your train program
- **file** is a file containing sentences in either English or Dutch.
- ***testing-type*** *specifies the type of learning algorithm you will run.*
  Either "dt" or "ada".

RUNNING FLOW OF THE PROGRAM

Let us consider one example.

Say, we are running a “dt” model.
The flow of the program along with the output will be as followed.

python3 driver.py train train.dat dtModel.obj dt

After running,

You will get your model saved by the name **dtModel.**

Using this model, we run the test command,

python3 driver.py predict dtModel.obj test.dat dt

After running this command, we get the following output.

en
nl
en
nl
en
en
nl
en
en
en

These are nothing but the languageLabel for all the lines that are there in test.dat.

That is, “en” on line 1 means that the first statement in test.dat is a english sentence.

Features

The strength of a model is heavily dependent on the strength of the features used in training. While some were generic, a vast majority of the features used were geared towards English and Dutch.
To split up the data, I selected 5 features based on frequent patterns (common words, etc.) in the training set.

Feature:
commonEnglishWords
Explanation:
Checking for the presence of common english words

commonWords = ['to', 'be', 'I', 'it', 'for', 'not', 'with', 'he', 'his', 'they', 'we', 'she', 'there', 'their', 'so', 'about', 'me', 'the', 'of', 'have', 'him', 'know', 'your', 'than', 'then', 'also', 'our', 'these', 'us', 'most', 'but', 'and']

Feature:
commonDutchWords
Explanation:
Checking for the presence of common dutch words

commonWords = ['naar', 'be', 'en', 'ik', 'het', 'voor', 'niet', 'met', 'hij', 'zijn', 'ze', 'wij', 'ze', 'er', 'hun', 'zo', 'over', 'hem', 'weten', 'jouw', 'dan', 'ook', 'onze', 'deze', 'ons', 'meest', 'niet','wat', ze', 'zijn', 'maar', 'die', 'heb','voor', 'ben','mijn','dit','hem','hebben','heeft','nu', 'allemaal', 'gedaan', 'huis', 'zij', 'jaar', 'vader', 'doet', 'vrouw', 'geld', 'hun', 'anders', 'zitten', 'niemand', 'binnen', 'spijt', 'maak', 'staat', 'werk', 'moeder', 'gezien', 'waren', 'wilde','praten', 'genoeg', 'meneer', 'klaar', 'ziet', 'een']

Feature:
englishArticles
Explanation:
“a”, “and”, “the” are very commonly used in english sentences.

Feature:
stringVan
Explanation:
Van is a very commonly occurring string in Dutch.

Feature:
stringDeHet
Explanation:
De, Het are two very commonly occurring words in Dutch.

Each statement in our input file runs through all our features to make sure the model understands which language the sentence is written in.

Models

Two classification models are used in the program.

Decision Tree

Entropy is used as a measure of impurity in a given set, and the Information gain algorithm is used to split the data by features. The decision tree can be assigned a maximum depth to restrict its growth.

The data consisted of few sentences with labels of English and Dutch for training and few sentences of both languages for testing. The decision tree accepts a parameter "depth" which is an upper bound for the depth of the tree.

I experimented with trees of varying heights and checked their error rates. A minimum depth of 0 works absolutely fine. The model for some reason fails to predict when the depth is set to 2 but works flawlessly when depth is increased. Personally, I decided to go with a tree that was 1 levels deep.

With depth set to 1, of all the examples only one example was classified wrongly. It can be easily fixed by just tuning in the features.

- Decision tree parameter settings

For changing depth, assign new value to depth variable in **dtTrain** function.

AdaBoost

A boosted ensemble of decision stumps (decision trees with a depth of 1) is built using the training data. Every instance of data is assigned a weight (forming a distribution)

and the AdaBoost algorithm is used to adjust the weight of each instance before the next stump in the ensemble is created.

The data used for the decision tree was also used to train and test the adaBoost model. In this case, however, a weight was assigned to each example of the data, forming an even distribution of weights. The train method requires a single parameter "totalDecisionStumps" which is the total number of decision stumps to be used in the ensemble. I experimented with various ensemble sizes and checked the error rates. An ensemble of 20 decision stumps was enough to accurately classify the training and test data. I went with 70 to account for larger test sets.

With totalDecisionStumps set to 70, of all the examples only one examples was classified wrongly. It can be easily fixed by just tuning in the features.

- AdaBoost parameter settings

For changing totalDecisionStumps, assign new value to in **adaDataCollection** function.