Lab 3: Wikipedia Language Classification

Anishya Thinesh
CSCI 331

1 Features

I brainstormed possible features by browsing different Dutch and English Wikipedia pages and observing certain patterns that were unique to both languages. Through this and additional internet research I was able to derive the following features.

- Dutch Function Words: Does the sentence contain popular Dutch function words like 'en', 'maar', 'als', etc.? These are not English words that would be in any sentence so I thought it would be good at differentiating the two languages.
- Dutch Frequent Substrings: Does the sentence contain substrings of length 2 that are common to the Dutch language? Like 'ij', 'aa', 'oo', etc.? I saw these substrings occur a lot in the Dutch Wikipedia pages and not so much in the English ones so I thought it would be a good separator.
- Average Word Length: When considering non-function words, Dutch words seemed to be a little longer than English words so I thought this would be a good feature. After looking at actual data for average length, I picked a word length of 6 to differentiate between returning True to indicate Dutch and False for English.
- Common Substrings of Two Words: checking for strings consisting of two words that were common in the Dutch language, like 'de het' and 'was van'. Unlikely for these to be in English excerpts so it's a effective decider for classification.
- Common Substrings of Three Words: checking for strings consisting of three words that were common in the Dutch language, like 'op de hoogte' and 'was van de'. Unlikely for these to be in English excerpts so it's a effective decider for classification.

2 Decision Tree Learning

The decision tree learning algorithm accepts the feature sets of the input, the max depth of the tree, and the weights associated with the feature sets. When the DT algorithm is used by itself, it is given 'dummy weights' of 1 for each example. The DT algorithm first finds which feature to split the data on to provide the most information gain. This is aided by functions that calculate remainder and entropy. The algorithm makes groups based on the boolean answer to what feature is the most profitable. Then it creates each branch of the tree recursively. When the max depth is reached, then it determines the majority language represented in the remaining data. I came up with the best parameters for max tree depth of the tree by iterating through depths 1-20 and calculating the percent that the model accurately classified from the test data. I found the highest percentage and went with that depth, 4. Other popular percentages were 80, 70 percent.

```
The highest accuracy rate of 90.0was MAX_TREE_DEPTH=4
```

Figure 1: result for configuration with highest accuracy

3 Boosted Learning

The adaptive boosting algorithm takes in the feature sets of the input, a learning algorithm, and the number of hypotheses in the ensemble. The algorithm initializes an array for example weights, hypotheses, and hypotheses weights. The algorithm uses the DT algorithm to return the desired amount of hypotheses, and for each hypothesis, misclassified examples are given more weight. The algorithm then

returns the list of hypotheses and their hypothesis weights. To determine the number of useful stumps, I again iterated through two values for number of stumps and max depth to find the combination that yielded the highest accuracy rate.

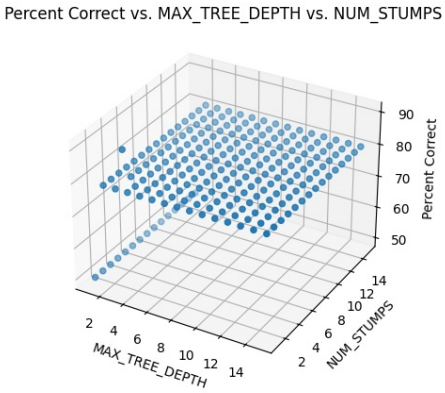

Figure 2: all accuracy rates for various configurations of max depth and num stumps

```
The highest accuracy rate of 90.0was with NUM_STUMPS=2 and MAX_TREE_DEPTH=3
```

The optimal configuration for adaboost was 2 stumps and a tree depth of 3.

4 Documentation

1. **Training Mode**: `python script.py train <example_file> <hypothesisOut_file> <learning_type>`
   - `<example_file>`: Path to the labeled training data.
   - `<hypothesisOut_file>`: File to save the trained model.
   - `<learning_type>`: Choose between "dt" (Decision Tree) or "ada" (AdaBoost).
2. **Prediction Mode**: `python script.py predict <hypothesis> <file>`
   - `<hypothesis>`: Path to the trained decision tree or AdaBoost model.
   - `<file>`: Path to the file containing data to predict.