Wikipedia Language Classification

Aswathi Kunnmubrath Manden

1. Features chosen for description of language

English and Dutch have Germanic roots. They have lot of similarities. But along with that there are lot of differences as well. I have identified few based on my observation to come with the below list of attributes or features for each language:

["de", "van", "een", "ij", "aa", "and", "the", "of", "to", "avg"]

1. "de" – checking if the sentence has "de"
   "de" is the Dutch for "the" in English, which is a common word in the language. Since "de" is not a word in English we can safely use this to classify.

2. "van" – checking if the sentence has "van"
   "van" is the Dutch for "from" in English, which is also a common word in the language. Since "van" is used very frequently in Dutch compared to English we can use the frequency of this word as a feature to classify.

3. "een" – checking if the sentence has "een"
   "een" is the Dutch for "a" in English, which is also a common word in the language. Since "een" is not a word in English we can safely use this to classify.

4. "ij" – checking if the sentence has words with "ij"
   As per my observation, "ij" is a frequently found two letters combination in Dutch. This is less common in English. So, the frequency of this combination can be used as a feature.

5. "aa" – checking if the sentence has words with "aa"
   I have found that "aa" is another frequently found two letters combination in Dutch. There are very few words with this combination in English. Again, the frequency of this combination can be used as a feature.

6. "and" – checking if the sentence has "and"
   "and" is a common word in the language. Since "and" is not a word in Dutch we can safely use this to classify.

7. "the" – checking if the sentence has "the"
   "the" is a common word in the language. Since "the" is not a word in Dutch we can safely use this to classify.

8. "of" – checking if the sentence has "of"
   "of" is a common word in the language. Since "of" is not a word in Dutch we can safely use this to classify.

9. "to" – checking if the sentence has "to"
   "to" is a common word in the language. Since "to" is not a word in Dutch we can safely use this to classify.
10. "to" – checking the average length of words
    Average length of words in English is 4.7 -5. So, if the average is less than 5, the probability of the sentence being English is high compared to Dutch. This can used as a feature.

2. Decision tree learning

Decision tree algorithm builds the classification models in the form of tree Data structure. The main algorithm is called recursively to build the tree based on True or False values of an attribute.
I have followed the decision tree learning algorithm below.

function DECISION-TREE-LEARNING($examples$, $attributes$, $parent\_examples$) **returns** a tree

if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$)
else if all $examples$ have the same classification **then return** the classification
else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, $examples$)
  $tree \leftarrow$ a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
    $subtree \leftarrow$ DECISION-TREE-LEARNING(exs, $attributes - A$, $examples$)
    add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
  return $tree$

I have used the information gain to build the tree based on the best attribute selected.

- Information Gain
  The information gain is based on the decrease in entropy after a dataset is split on an attribute.
  $$Gain(T, X) = Entropy(T) - Entropy(T, X)$$
  T is the Target or goal. X is the attribute.
- Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.
- A branch with entropy of 0 is a leaf node
- The algorithm is run recursively on the non-leaf branches, until all data is classified.

3. Adaboost

Training:
- Adaboost builds ensemble using a learning algorithm. It uses weak classifiers to build the decision.
- The algorithm that I used is given below.

- The value of K or the number of stumps that is used is 10. Initially I have created an array to hold the weights for each line of rows (N) in the training data file and all the weights are initialized to be 1/ N. Then iterate over the example list which has the attributes.
- Error is calculated for each attribute based on the correctness of the values.
- Based on the error, weight of each row is updated.
- The weight is normalized after each iteration. The weight of each stump is calculated using:

$$\mathbf{z}[k] \leftarrow \log\,(1 - error)/error$$

Each stump along with weight is added to a list. This list object is saved to the file given by the user.

Predicition:

- To predict the user data, first we read the data from the saved file.
- Then the data is processed and evaluated by the stumps to generate the classification value.
- If the value is positive the language is Dutch and "nl" is printed to the console.
- If the value is negative the language is English and "en" is printed to the console.

function ADABOOST($examples, L, K$) **returns** a weighted-majority hypothesis
  inputs: $examples$, set of N labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$
    L, a learning algorithm
    K, the number of hypotheses in the ensemble
  local variables: $\mathbf{w}$, a vector of N example weights, initially $1/N$
    $\mathbf{h}$, a vector of K hypotheses
    $\mathbf{z}$, a vector of K hypothesis weights

  for $k = 1$ **to** K **do**
    $\mathbf{h}[k] \leftarrow L(examples, \mathbf{w})$
    $error \leftarrow 0$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) \neq y_j$ **then** $error \leftarrow error + \mathbf{w}[j]$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) = y_j$ **then** $\mathbf{w}[j] \leftarrow \mathbf{w}[j] \cdot error/(1 - error)$
    $\mathbf{w} \leftarrow$ NORMALIZE($\mathbf{w}$)
    $\mathbf{z}[k] \leftarrow \log\,(1 - error)/error$
  return WEIGHTED-MAJORITY($\mathbf{h}, \mathbf{z}$)

1. **Test of code**

The code to classify the language is in languageLearnAndPredict.py
When you run the code user is prompted with 2 options or with the 2 entry points to the program.

The two entry points to the code:

1. Training part

```
train <examples> <hypothesisOut> <learning-type>
```

2. Predict part

```
predict <hypothesis> <file>
```