**Overview:**

We are given a supervised classification problem. Given a muliple examples of text from afrikaans, english and dutch with the corresponding labels , we need to train a algorithm to correctly classify a unseen text as belonging to one of the languages

**Data description:**

The data consists of 2761 observations. A single observation consists of raw text representing a pharse and a label representing the corresponding language associated with the text.The are 3 labels. English, Afrikaans and Dutch(or Nederlands). Associated with the raw text file are non language symbols such as punctuation and other miscellaneous symbols for example "@".

**Data Prep.**

**Missing obersvations** There are several observations labels but no raw text data. All such observations were removed from the dataset.

**Non-Alphabet Symbols and Font** All non alphabets symbols were removed. All words were converted to lower case and punctuation marks were removed.

**Packages used:** The following packages were used in learning the classification problem:

The main package library used was the sklearn library. The following packages were used from the sklearn library:

- CountVectorizer from the feature extraction library used to tolkenize raw text data and turn text data into numerical features appropriate to apply to machine learning algorithms
- svm This is the Support Vector Machine package which willed be used for classification problems. The linear svm was used.
- The cross val score package from model selection in sklearn library This used for paramter esitmation for the learning algoritms. We wish to use this library to select the parameters for the SVM which lead

to the lowest cross validation misclassification error rate/highest CV accuracy rate.

Other packages used include:

- Pandas Pandas package used to read and manipulate data sets.
- Numpy The numpy package is used to manage mathematical operations and arrays.
- pickle The pickle package is used to save/store python objects.

**Model Architecture:**

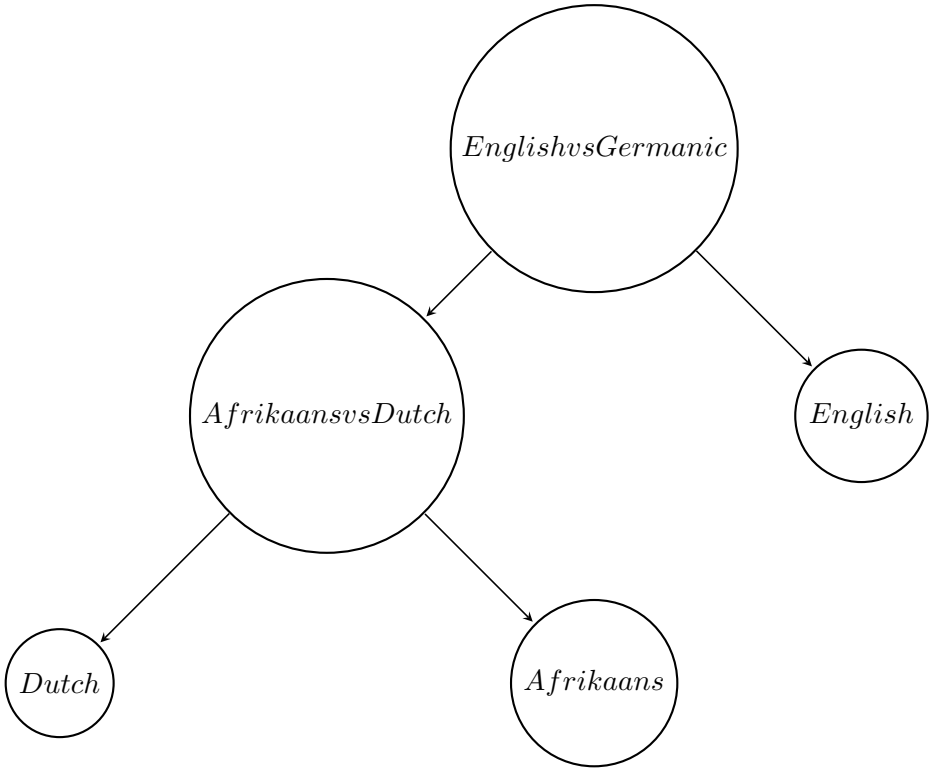
The support vector machine classifcation alogrithm has been used for classifying the which language corresponds to the raw text phrase.

**Multiples Labels** The original support vector machine is a binary classifier. In this problem we have 3 labels. The following strategy was implemented in order classifiy between the 3 labels

1. We use the fact that Dutch and Afrikaans are extremmely similar in terms of linguistic structure (called them Germanic Languages)and that these two languages are quite dissimilar from Engish. We therefore first train a SVM to classify whether a raw text is English or Germanic .
2. If a text has been classified as Germanic we further classify this text as either Afrikaans or Dutch.

The above procedure is a combination of two multi-class classification methodologies one vs all and DAGSVM(Directed Ayclic Graph Support Vector Machine).

One vs All has used to classify English vs Germanic Languages. The DAGSVM corresponds to the structure used to classify a new text data observation as shown the below graph.




**Feature Extraction**

The SVM needs numeric features and so we need to convert the text data to corresponding numeric features. The most popular way of doing this is the "bag of words" procedure. Let $t_1, t_2....t_n$ represent the different n raw text files we have. Each text file consists of words from a unique language $L_1, L_2, L_3$. We form a dictionary of the all the unique words appearing in the **training set** $w_1, w_2, ....w_p$. This set of unique words or vocabulary represents the features. observation $t_i$ corresponds of the vector $\langle w_1, w_2, ....w_p \rangle$ and label $L_i$.

For example if $t_i = \textit{This is is a a english sentence}$ then $w_1 = \textit{This}, w_2 = \textit{is}, w_3 = \textit{a}, w_4 = \textit{senstence}, w_5 = \textit{english}$ and the corresponding values $t_i$ would take is $\langle 1, 2, 2, 1, 1, 1 \rangle$

The above is an example of **word tolkenization**. It may be useful to tolkenize using different permutations of letters that occur in words that make up the text. Such sequences of letters are referred to as n-grams. For

example a 1-gram corresponds to single letter occurrences in a text. A 2-gram to all two letter occurrences and so and so forth. For example some of the 2-grams of the sentence: "Hy lag baie" are "ag,ai,ie,hy,ba" . Feature vectors are constructed the same way but now n-grams are used instead of words.

I used two word tolkenization when comparing between English and Germanic languages. Since the vocabualry/spelling structure between is quite diffreent keeping only dictionary of the words produced quite good classifcation rates.Finer partitions such as the n-gram approach adds too much complexity for a small improvement in the classification rate. Howvever since Afrikaans and Dutch are extremely similar(one is a derivative of the other) finer partioning of the text structure has to be employed. I decided to use 1,2,3-gram letter sequences as my feature for classifying between afrikaans and dutch. I choose smaller ngrams since the difference between afrikaans consists of slight deviations in spelling. For example a difference between dutch and afrikaans spelling is the replacing of cc with kk in spelling various words. E.g "accommodatie" in dutch corresponds to "akkommodasie" in afrikaans. Here the difference is in the 2-gram cc and kk.

**Class Imbalance**

After cleaning up the data data it is noticed that there quite a large class imbalance between the groups.I used 0.7 of the data(traing set) to train the SVMs and 0.3(test set) to evaluate their performance. In the training data there were 1442 observations with the english label, 47 with the Nederlands label and the remainder was the Afrikaans label. Although the SVM is more robust to class imbalance since only a few points observations are necessary to maxmimum margins(the support vectors), large class imbalance can still lead to a degradation in performance due to disproportionate ratio support vectors either side and due the large penalty values required to account for the unequal classes.

To overcome I used decided to use a **ensemble of balanced SVMs**. To do this I constructed $K = \lfloor \frac{n_{excess}}{n_{under}} \rfloor$ linear SVM classifiers where each consist of $n_{under}$(where $n_{under}$ is the number observations with the minority label) observations from each label. Given a test observation each of the K classifers decide of which label to assign to the test observation and the label with highest confirming votes is decided as the final label. For English vs Germanic Languages $n_{under} = 490$ and I constructed 3 SVMs classfiers.

For Dutch vs Afrikaans $n_{under} = 47$ and I constructed a ensemble with 9 SVMs to vote.

**Parameter Estimation**

Since we are using a linear SVM the only parameter to be estimated was C the penalty on misclassifications. 1-dimensional Grid search together with cross validation was performed on each classifier in the ensemble to calculate the best C value(C corresponding to the best CV accuracy.)3-fold cross validation was performed on English vs Germanic classification and 2-fold cross validation was performed on Dutch vs Afrikaans due the small sample sizes for each of the 9 SVMs.

The overall accuracy was at about 97%. On English vs Germanic classification where was a 97% accuracy .The accuracy between Dutch and Afrikaans was poorer when word tolkenization was used as compared to n-gram tolkenization. The Dutch -Afrikaans accuracy using n-gram tolkenization was 95%. Out of 20 observations Nederlands observations in test set, 14 were correctly identified as Dutch .

**Room for improvement**

- I used a linear SVC for classification, however when the decison boundary is not linear it is better to use the kernel SVC. The rfb or polynomial kernels could have been used and may lead to better test accuracy.
- I could've used ngram instead of word tolkenization for english and a wider range n-gram lengths. This allows for more complxity in the learning algorithm which leads to a more flexible/robust classifier
- Many features I included were noisy since no feature selection was performed. I could've removed irrelevant features(using recursive feature elimination or $L_1$ norm pentalities for example ) to reach a similar accuracy which far less features leading to a less computation effort and training time