The Language Classification Problem

Devon Stone

devon12stone@gmail.com

+ 27 72343 2779

ABSTRACT

This report details the methods used to create a Naïve Bayes classifier capable of determining whether a phrase was written in English, Afrikaans, or Dutch. The classifier was trained and tested on a set of 2671 independent phrases. The classifier was trained on 80 percent (2208) of the total phrases. The accuracy of the classifier was then tested using the remaining 463 phrases. The classifier achieved an overall accuracy of 97,6%.

EXTERNAL LIBRAIRES USED

The solution to this problem was written using Python version 3.6.3. The external libraries used to solve the problem were pandas, numpy and sklearn. The pandas library was used to read in and analyze the data provided in the problem. The numpy library was used to store and manipulate data. While the sklearn library provided a method to tokenize the words in the training and test phrases as well as a method used to create the Naïve Bayes classifier.

DATA CLEANING

Before any feature extraction could take place, the data had to be cleaned. This was completed in the following four ways

- Rows in the data that contained blank phrases were removed from the data set using pandas' *dropna()* function
- All phrases were reduced to lower case using Python's *lower()* function.
- All potential punctuation was removed from the phrases using a for loop.
- Additional whitespaces in the phrases were removed using a combination of Python's join and split functions.

TRAINING AND TESTING SETS

The data provided had to be divided into training and testing data sets. The data was divided into a training set containing 80% of the phrases (2208) and a testing set containing 20% of the phrases (463).

The training and testing sets were created by first reading the provided data into a Python module using pandas and then converting the pandas data frame into two numpy arrays. The first numpy array contained all the phrases while the second contained each phrase's associated language. The two arrays where then sliced into training and testing sets.

FEATURE EXTRACTION

The process of extracting from the set of phrases was completed using the bag of words approach. This was done using sklearn's CountVectorizer() function. The function is found in sklearn's feature extraction module.

The CountVectorizer() function tokenizes each phrase into individual words and assigns a unique identifier, in this case an integer, to each word in the phrase. This process generates a vector containing all the unique words in the entire data set/feature space. From this vector a matrix is generated where the rows of the matrix represent the phrases in the data set and the columns represent each unique word. The matrix is a sparse matrix as each phrase only contains a few of the unique words from the entire feature space of words.

The matrix generated for this solution had the shape 2208 (all phrases contained in the training set) by 3773 (all unique words contained within the phrases in the training set). The table below hypothetical represents how the generated matrix appears…

	word_1	word_2	…	word_3772	word_3773
phrase_1	1	0	…	0	0
phrase_2	0	1	…	1	0
…	…	…	…	…	…
phrase_2207	2	0	…	1	0
phrase_2208	0	1	…	0	1

CLASSIFIER

Naïve Bayes classifiers are simple, efficient, and accurate when used for text classification. Therefore, a Naïve Bayes classifier was used in this solution, to be more specific, sklearn's multinomial classifier.

A Naïve Bayes classifier uses Bayes theorem as an underlying algorithm. Bayes theorem appears as follows…

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

The above equation is used to calculate the conditional probability of event A occurring given that event B has occurred. For this to be calculated the probability of event B occurring must be known, the probability of event A occurring must be known and the conditional probability of event B occurring given that event A has occurred must be known.

In the case of this solution, the classifier attaches a language label (either English, Afrikaans, or Nederlands) to each row in the bag of words matrix created during feature extraction. The classifier then creates a look up table of the conditional probabilities for every word in the training feature space. The conditional probabilities for every word appear as follows…

- *P(word | Enlgish)*
- *P(word | Afrikaans)*
- *P(word | Nederlands)*

Phrases in the testing data set were then classified (language labels were attached to the phrases) based upon the conditional probabilities of each word in the look up table generated from the words in the training data set. For example, the conditional probability of the phrase "how are you" being English would appear as follows….

P("how are you" | English) = P("how" | Enlgish) * P("are" | English) * P("you" | English)

The phrase "how are you" would further have conditional probabilities for it being Afrikaans or Dutch. The classifier would then compare the three conditional probabilities calculated and then attach a language label to the language with the highest conditional probability.

RESULTS

After the model was trained the test data set was fed into the model. The accuracy of the model was tested by comparing the predicted language labels of the test set to the actual language labels of test set.

The overall accuracy and accuracy for each language of the classifier can be seen below…

	Accuracy (%)
English Phrases	99,3
Afrikaans Phrases	99,2
Dutch Phrases	30,8
All Phrases	97,6

The overall accuracy of the classifier is very good, even though the Dutch phrases were not classified to a high accuracy. The reason behind the poor accuracy when classifying Dutch phrases is a simple case of class imbalance. There are not enough Dutch words in the training data relative to English or Afrikaans words. A method to overcome the class imbalance would be to use a Synthetic Minority Over-Sampling Technique (SMOTE). This would involve generating similar instances of the Dutch phrases that would be based from the Dutch phrases in the training data set.

BONUS QUESTIONS

OTHER APPROCHES

The Naïve Bayes classifier was used as it has been found to be efficient, not computationally complex and accurate when classifying text. The issue with a Naïve Bayes approach is that the method does not have the ability to classify words that are not contained with the training set.

A second option would be to use a neural network. A neural network could be trained on the same bag of words as the solution. However, a neural network would have the ability to learn new words as it could be trained to search for specific patterns (groupings of ASCII numbers) within words that belong to each specific language.

Another option would be to use a Support Vector Machine (SVM) algorithm. An SVM algorithm could be used to generate a hyperplane that could ideally separate each of the three languages from one another. An SVM algorithm would deal with the class imbalance issue within the data better than the solution used, however it is more computationally complex.

SUPERVISED VS UNSUPERVISED CLASSIFICATION

Supervised classification deals with the classification of labelled data, while unsupervised learning deals with the classification of unlabeled data. Supervised classification algorithms are more commonly used in problems where patterns in the training data is used to classify test data, for example in image classification. While unsupervised classification algorithms are more commonly used for clustering analysis.

CLASSIFICATION VS REGRESSION

Classification algorithms are used to determine the class of an input value (predicting a categorical variable). For example, a classification algorithm would be used to determine whether a feature in an image was a car, a dog or if it belonged to another possible class from a finite set of classes.

A regression algorithm is used to predict a continuous variable. For example a simple linear regression of a straight line uses an x coordinate to predict the value of a y coordinate.

CLASS IMBALANCE

A class imbalance is found in supervised learning and is when one class contains significantly less data in the training data set when compared to the other classes in the data set. The classification rules developed by the classification model are often weaker and not as successful for the smaller class when compared to the prevalent classes.

MITIGATING THE EFFECTS OF CLASS IMABLANCE

The solution to this problem contained a class imbalance and it was explained previously (in the Results Chapter) how SMOTE could be used to overcome the imbalance problem.

NON-DEEP VS DEEP LEARNING

Machine learning is the study of using data to analyze problems and make informed decisions. Non-deep learning is when there is human input in the process of classify or clustering features. Therefore, human error can affect the decision. Deep learning models require no human input and perform feature extraction autonomously.