

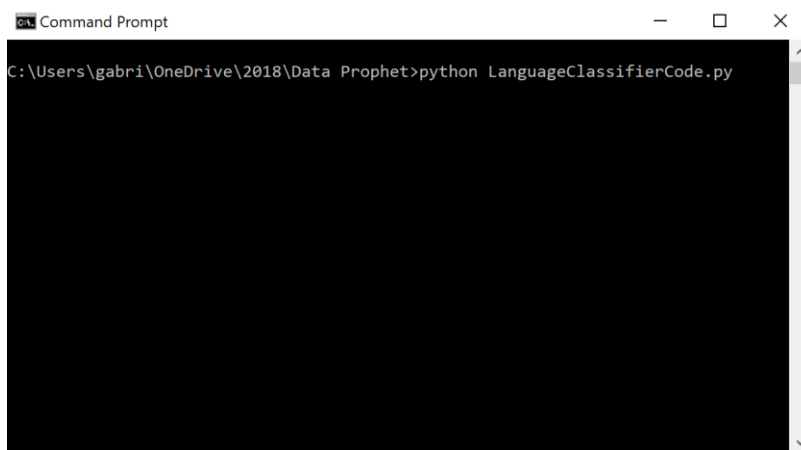
REPORT

The Language Classification Problem

Gabriel Stein

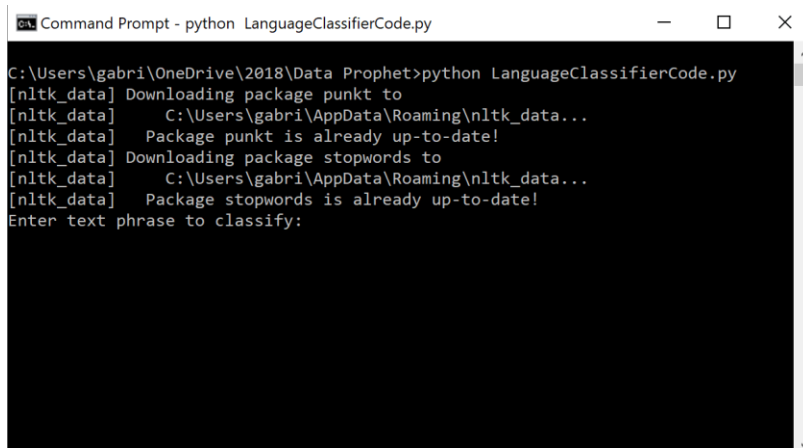
Instructions on how to execute code:

1. Execute the program via the command line.



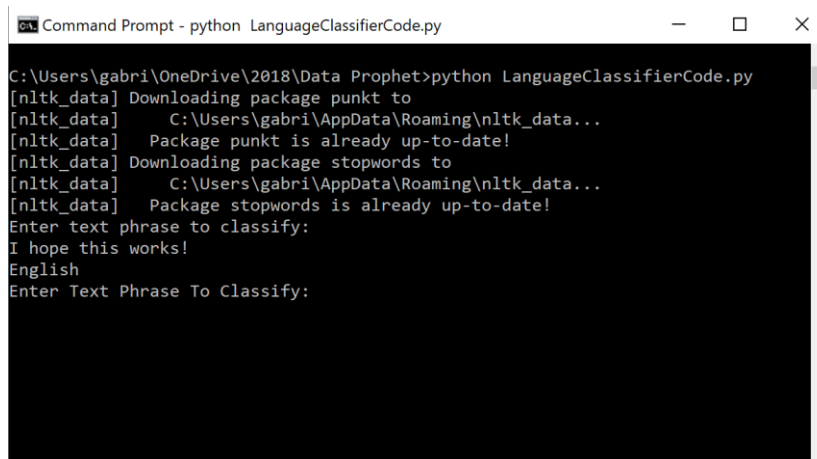
```
Command Prompt
C:\Users\gabri\OneDrive\2018\Data Prophet>python LanguageClassifierCode.py
```

2. If the “punkt” or “stopwords” package is not already installed on your computer, they will be downloaded and installed, you will then be prompted to enter a text phrase to classify.



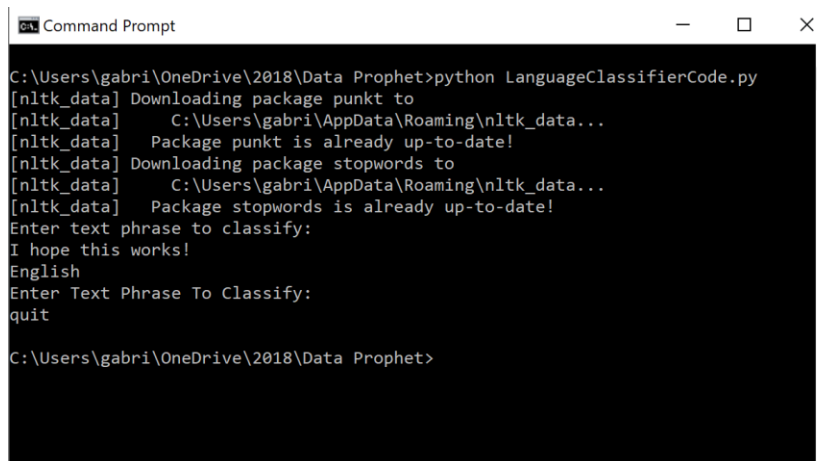
```
Command Prompt - python LanguageClassifierCode.py
C:\Users\gabri\OneDrive\2018\Data Prophet>python LanguageClassifierCode.py
[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\gabri\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]   C:\Users\gabri\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Enter text phrase to classify:
```

3. Type in a sentence (there is no input validation) and the program will then output the predicted language and then let you classify another phrase.



```
Command Prompt - python LanguageClassifierCode.py
C:\Users\gabri\OneDrive\2018\Data Prophet>python LanguageClassifierCode.py
[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\gabri\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]   C:\Users\gabri\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Enter text phrase to classify:
I hope this works!
English
Enter Text Phrase To Classify:
```

4. Type in 'quit' to exit the program



```
Command Prompt
C:\Users\gabri\OneDrive\2018\Data Prophet>python LanguageClassifierCode.py
[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\gabri\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]   C:\Users\gabri\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Enter text phrase to classify:
I hope this works!
English
Enter Text Phrase To Classify:
quit
C:\Users\gabri\OneDrive\2018\Data Prophet>
```

Libraries used:

Natural Language Toolkit (NLTK):

- **Tokenize**
A tokenizer that divides a string into substrings by splitting on the specified string
- **Corpus**
Contains stopwords (common words in a language such as 'and', 'to', 'the') belonging to specific languages, in this case, English and Dutch (Afrikaans not yet supported)
- **Classify**
Contains the Naïve Bayes Classifier that the dataset was trained on

Pandas:

Used to read the dataset from the CSV into a python dictionary. Later used to make a list of the 10 most frequent and least frequent words in the dataset. This list is used to implement dimension reduction for the text feature sets.

Pickle:

Used to serializing the model post-training (save the model in a trained state) and then later used to de-serialize the model in order to use it.

Pipeline:

This data is subjected to two cleaning steps:

- Text is converted to lower case
- All items that contain a blank field are excluded

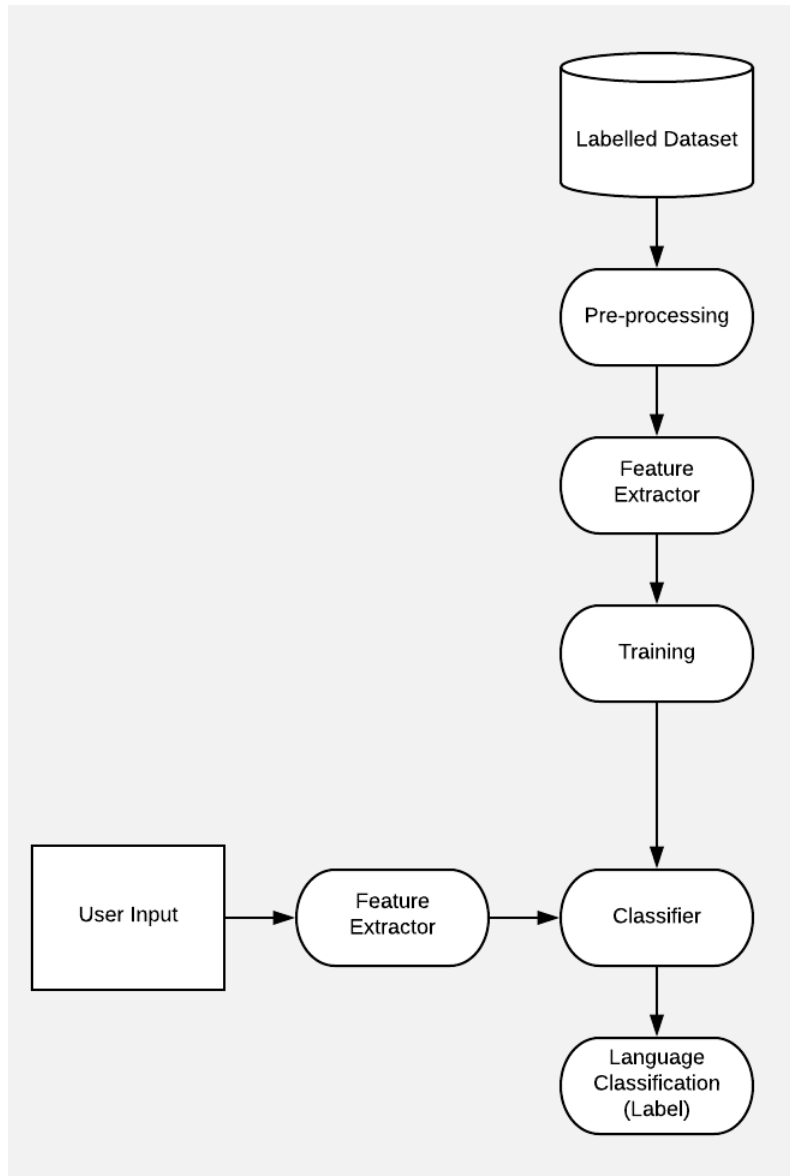
A list consisting of the 10 most frequent words, the 10 least frequent words and English and Dutch stop words is used during feature extraction to reduce the feature set for each item in the dataset.

A list of all words ("vocabulary") in the dataset (minus the words in the list mentioned above) was compiled to be used to create text features for each phrase.

Each phrase is tokenized. Each word is then checked against the "vocabulary". All words in the vocabulary have a set state of false, if a word is contained within the given phrase, then its value is changes from false to true. In other words, each phrase has its own corresponding vocabulary, where only the words in the phrase are marked true. This is known as the Bag-of-Words model.

The Naïve Bayes algorithm then uses these features to predict which language class the phrase belongs.

Model architecture:



Overview of Testing Process:

In order to test the accuracy of the model, I divided the dataset into a train and test feature set. The test feature set contained 500 items whilst the train feature set contained the rest (approximately 2000 items). I then used the built in “accuracy” method within the NLTK library to measure the accuracy of the model using the test feature set.

I tested the following:

1. The classifier’s accuracy over the entire test dataset
2. The classifier’s accuracy for each language

In order to test 2 (from above), I divided the test dataset into three separate datasets for each language. I individually tested the classifier on each of these datasets. It is important to note that the composition of the test dataset was fairly distributed among the classes. The size of each class within the test dataset, makes up on average 18% of the total size of each class within the dataset (e.g. there are 100 phrases of each language within the dataset, then there is 18 phrases of each language in the test dataset).

```
Size Percentage of Each Test Class to the Total Class Size Within the Data Set:
English      17.0
Afrikaans    19.0
Nederlands   18.0
```

Overview and Discussion of Results, Performance on each Language and Potential Improvements:

An important thing to note is the class imbalance within the dataset, as shown below:

```
Size Percentage of Each Class Within the Data Set:
English      73.0
Afrikaans    24.0
Nederlands    3.0
```

It is clear that majority of the dataset contains English phrases with Afrikaans as a far second and the minority being Dutch. We can see below that this has a clear impact on the classifier’s performance in each language.

```
English Proficiency of Naive Bayes Classifier: 100.0
Afrikaans Proficiency of Naive Bayes Classifier: 81.6
Dutch Proficiency of Naive Bayes Classifier: 12.5
```

While the proficiency in Dutch is very low, the classifier’s average performance is very good at 92.6%

```
C:\Users\gabri\AppData\Local\Programs\Python\Python35-32\python.exe C:/Users/gabri/Desktop/LanguageClassifierCode.py
Accuracy of Naive Bayes Classifier: 92.60000000000001%
```

Bonus Questions:

1. Two machine learning approaches that would also be able to perform the task:

There are many different algorithms and approaches to text classification. Two of the more common approaches besides Naïve Bayes (and multinomial Naïve Bayes) are the implementation of Support Vector Machines (SVM) or Random Forests (RF). However, most cases still use the bag-of-words model for feature extraction. Therefore, I would say that my approach would still be similar for both of these approaches.

The RF approach is primarily suited to multiclass problems, while the SVM is intrinsically two-class (the same as Naïve Bayes). If I were to use the RF approach, for the multiclass problem, I would need to reduce it into multiple binary classification problems.

For classification problems, a Random Forest will give a probability of belonging to a class. An SVM gives you the distance to the boundary, therefore, I would still need to convert it to a probability.

2. The difference between supervised and unsupervised learning:

If we think of a model as a classroom of students, and our dataset as the teacher then we can explain supervised and unsupervised learning in the following way:

Suppose you are taking zoology as a subject and your teacher is having you learn what family (i.e. Cat or Dog) every breed of animal is (i.e. A golden retriever belongs to the dog family). Supervised learning refers to the teacher showing you 10 pictures and telling you that they are dogs. In the class test, there is a picture you haven't seen before of an animal you don't exactly recognise and you're asked what family it belongs to, because you have seen 10 pictures of what dogs look like, you can pretty much guess it's a dog.

In technical terms, Supervised learning is where you have input variables (The teacher showing you pictures) and an output variable (The teacher telling you they are dogs) and you use an algorithm to learn the mapping function from the input to the output. The goal is to approximate the mapping function so well that when you have some new input data (The picture in the test) that you can predict the output variables (Guessing that it was a dog) for that data.

Now if let's go back to when you were learning. The teacher now shows you 10 images of animals but doesn't tell you what type of animal they are, so instead of classifying each picture, you put the images in groups based on some similarities. Unsupervised learning is when you only have input data (The 10 pictures) and no corresponding output variables (no direct answer for each picture). The goal for unsupervised learning is to model the underlying structure in the data in order to learn more about the data.

3. The difference between classification and regression.

While classification and regression are both supervised learning techniques, there is a difference. A classification model tries to approximate a mapping function that predicts a label or class for a given input variable.

For example, when shopping for a car at a car dealership, one can classify a car being sold as either: expensive; cheap or average price. Classification models commonly predict a continuous value as the probability of a given input variable (car price) belonging to each output class (expensive, cheap, average). The probabilities can be interpreted as the likelihood of a given input variable belonging to each class. A predicted probability can be converted into a class value by selecting the class label that has the highest probability. For example, a specific car price may be assigned the probabilities of 0.1 as being "expensive", 0.7 as being "cheap" and 0.2 as being "average". We convert these probabilities to a class label by selecting the "cheap" label as it has the highest predicted likelihood. A regression model tries to approximate a mapping function that predicts a continuous output variable for a given input variable. A continuous output variable is a real-value, such as an integer. These are often quantities, such as amounts and sizes. For example, a car may be predicted to sell for a specific rand value based on certain properties (car make, engine type etc.).

4. The consequences of class imbalance can be best shown with the following common example:

Consider a dataset of transaction data belonging to a certain company. We would like to find out which transactions are fraudulent and which are genuine. It is very costly to the company if a fraudulent transaction gets processed as genuine as this negatively impacts our customers trust in us and costs the company money. Therefore, we want to catch as many fraudulent transactions as possible.

If the dataset consists of 10000 genuine and 10 fraudulent transactions, the classifier will tend to classify fraudulent transactions as genuine transactions.

Suppose the machine learning algorithm has two possible outputs:

- Model 1: Classified 8 out of 10 fraudulent transactions as genuine transactions and 10 out of 10000 genuine transactions as fraudulent transactions.
- Model 2: Classified 2 out of 10 fraudulent transactions as genuine transactions and 100 out of 10000 genuine transactions as fraudulent transactions.

If we determine the classifier's performance by the number of mistakes it makes, then Model 1 is better as it made only a total of 18 mistakes while Model 2 made 102 mistakes. However, since we want to minimize the number of fraudulent transactions happening, Model 2 is the better option as it only made 2 mistakes classifying the fraudulent transactions. Model 2 may classify more genuine transactions as fraudulent, but this is a price worth paying for now.

A general machine learning algorithm will pick Model 1 over Model 2, which is a problem. In practice, it would mean letting a lot of fraudulent transactions go through even though we could have stopped majority of them by using Model 2.

5. Methods to reduce the negative consequences of the class imbalance problem:

- Resampling Techniques (Improve the balance between the data):
Over-sampling: You can add copy instances from the under-represented class
Under-sampling: You can delete instances from the over-represented class
- Try different algorithms/models

While I am aware of changing the performance metric, I am not confident in my ability to apply this technique amongst others, and therefore have not formally listed them here.

6. The difference between using an Artificial Neural Network (ANN) vs. Naïve Bayes:

In a nutshell, Naive Bayes is far simpler to use than an ANN. However, when considering which method to use, one should take into account the amount of training data available.

Naive Bayes and ANNs have perform differently depending on the amount of training data they receive. The Naive Bayes classifier has been shown to perform well even with very small amounts of training data that ANNs would find insufficient. Therefore, if you have a small amount of training data, Naive Bayes is the better option. However, when provided with large amounts of training data, the story changes.

As the amount of training data being fed increases, due to the algorithms simplicity, the performance of the Naive Bayes classifier plateaus. In contrast, an ANNs' complexity allows its performance to improve as large amounts of training data are accumulated.

Another point to consider when dealing with large datasets is that the two models' complexities impact their tendencies to overfit. The Naive Bayes' simplicity prevents it from fitting its training data too closely. However, due to their complexity, ANN's can very easily over fit training data, especially when provided with large data sets.