AI Challenge
Song Language Predictor
Artificial Intelligence

Eduardo Vaca
A01207563

Introduction

For the course Artificial Intelligence, we had to develop a challenge, which basically it's a project related to any field of AI. I decided to create an application capable of predicting the language of any song which I called *Ich connais esa lied, I credo* (meaning" I know that song, I believe") each word on a different language (from the languages my model is capable of predicting).

I decided to do this project firstly because I love music, and I wanted to do something interesting related to it. I found first a dataset that maps release decade and timbre covariance values of a song and I wanted to predict the decade of a song by analyzing the timbre values. However, after some data analysis I realize that the relation wasn't that strong to create an accurate predictor of a song's decade. I only managed to obtained a model with 49% accuracy, which it's better than just guessing between 7 decades, but I wanted to have a more accurate project. Thus, I decided to change the idea of the project aiming for something more precise. Still with a musical project in mind I found an API which provides the lyrics of any song, and that's when I came up with the idea. I found interesting analyzing lyrics and predicting its language because songs contain informal expressions of the language which makes this problem harder than just predicting the language of a perfectly written paragraph.

My solution for this project was a Machine Learning model with 98% in accuracy (more analysis latter) and I believe it's very useful because the training was made with short phrases of each language instead of big chunk of words in paragraphs. Also and more important, to predict the language I do not go over the whole lyrics, I just analyze a maximum of 5 snippets (small sentences) and predict each one deciding the final result by obtaining the most common value. This allows a real fast and reliable prediction.

With this AI solution, you could save yourself of firstly having a person determining the language of a song and updating a database. Also you wouldn't need to have a gigantic dictionary of words from each language with an expensive algorithm that checks word by word from the lyrics in each dictionary. Finally, supporting more languages it's very easy in my solution, specially because I create my own dataset from Wikipedia articles, you would only need to add new searches in the new language to my *generate_data.py* script and train the model again to have a fully supported new language.

After some research, I've found there've been other projects related to language identification. For example, Twitter has a huge amount of data considering all the Tweets posted and they have specific teams for analyzing the data and for understanding the content of Tweets for reasons like grasping user's interests, improving search and fighting spam. In their engineering blog, they posted an article called *Evaluating Language Identification Performance* where they expose how they analyze Tweets and the steps involved in natural language processing defining *language identification* as the first and most important one. They state it's a hard problem due to the shortness and informality of a tweet, they can't just use a dictionary of common words, they had to build an intelligent model.

Evaluating language identification performance. (2016, November 15). Retrieved November 17, 2017, from https://blog.twitter.com/engineering/en_us/a/2015/evaluating-language-identification-performance.html

Process

For developing this project, I needed to do the following:

1. Create dataset

I looked for datasets of languages for training my model. However, I only found limited ones, with few languages, and with not so many records. Thus, I decided to create my own dataset to enhance more the experience of the challenge and also to provide the feature of being able to support any desired language in my model.
I created my dataset by sending requests to Wikipedia in the languages I wanted to support. I searched for:

- Love
    - Amour in French
    - Amor in Spanish
    - Liebe in German
    - Amore in Italian
    - Liefde in Dutch
- Person
    - Nom in French
    - Persona in Spanish
    - Person in German
    - Persona un Italian
    - Mens in Dutch
- World
    - Monde in French
    - Mundo in Spanish
    - Welt in Dutch
    - Mondo in Italian
    - Aarde in Dutch

I decided to use this words because after some research I found out those were the most used words in songs.
I had to clean the data by removing images, links and useless sentences like dates or references in order to use the data for training.

2. Extract Features from Text

In order to perform Machine Learning on text we first need to transform the text into numerical features.
I first found a strategy called *Bag of Words*, which consists on assigning an integer to a word frequency. But there are two problems with this approach:

- Some languages will have repeated words
- We cannot expect that the training dataset contains all of the words and their various forms in a language detection problem

Thus, I decided to follow an approach using a *Char Analyzer* which extract features on the character level instead of the word level. This helps a lot to consider suffixes and prefixes like *–able, -ness, -tion, a-, co-, pro-, etc.* Achieving that the features do not depend on the words themselves but on the parts specific to each language.

3. Build ML Model

Choosing the model required some research as well, I needed to find which model works best when working with text data. I found that Naive Bayes and Logistic Regression were a good fit.

Naïve Bayes Classifier

It uses Bayes Theorem to predict membership probabilities for each class such as the probability that a given record belongs to a particular class. It assumes that all features are unrelated to each other (that's why it's called Naïve). For example: Presence or absence of a feature doesn't influence in the presence or absence of another one. Specifically, I used Multinomial Naïve Bayes which is one of the variants used for text classification.

Logistic Regression

From Wikipedia: *Logistic Regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function.*

In other words, this model uses a black box function to understand a relation between dependent variable (target class wanted to predict) and some independent variables (attributes used to predict). The black box function is also known as Softmax function (popular function to calculate probabilities of an event).

4. Connect to lyrics API

To obtain the lyrics I'm using a free library from Python called PyLyrics. Because it's free it's very limited but you can find the song if it´s famous enough. I created a class responsible of fetching lyrics and choosing randomly the snippets to be analyzed for prediction. The name of the class is *lyrics_fetcher.py*.

5. Build the UI App

The project goal was to develop something using AI, thus my UI app is really simple. I developed it using Python's library called Tkinter.

Analysis

While developing the project I found out the following:

Because I created my own dataset that involved a lot of responsibility. My first approach was to use a whole paragraph as a single record, for example:

En → *"Love is a variety of different emotional and mental states, typically strongly and positively experienced, that ranges from deepest interpersonal affection to simple pleasure. An example of this range of meanings is that the love of a mother differs from the love of a spouse differs from the love of food. Most commonly, love refers to a feeling of strong attraction and personal attachment. "*

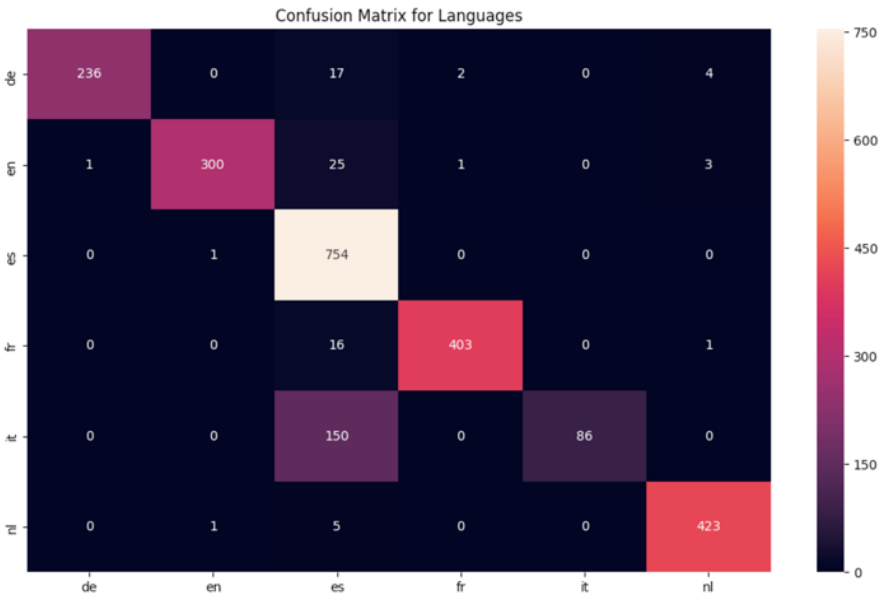
I ended up having around 2000 records to train my model and after training I obtained an accuracy of 100%. I was impressed but also with suspicious of over-fit. I tested it with lyrics snippets and precision wasn't 100%, it was 40% at the best scenario. After some analysis I found out that I was indeed over-fitting my model, I was using huge paragraphs as records and the test dataset had paragraphs as records too while snippets were only small sentences. Also the paragraphs were really well written in comparison of the song lyrics (as I previously mention, songs use informal language) so I was basically training my model to obtain a perfect performance on my test data but not on my real data which were lyrics snippets. So what I did was to split each paragraphs into small groups of 6 words (which was the average length of snippets) and with this training data I was training my model to distinguish languages with less words. My accuracy decreased to 93-98% but now the real predictions were on this range of precision too.

In addition, due to the fact that I used two Machine Learning algorithms I was able to compare both.
I obtained the following precision chart and confusion matrix by using Multinomial Naïve Bayes:

```
Evaluating...
             precision    recall  f1-score   support

         de       1.00      0.91      0.95       259
         en       0.99      0.91      0.95       330
         es       0.78      1.00      0.88       755
         fr       0.99      0.96      0.98       420
         it       1.00      0.36      0.53       236
         nl       0.98      0.99      0.98       429

avg / total       0.93      0.91      0.90      2429
```

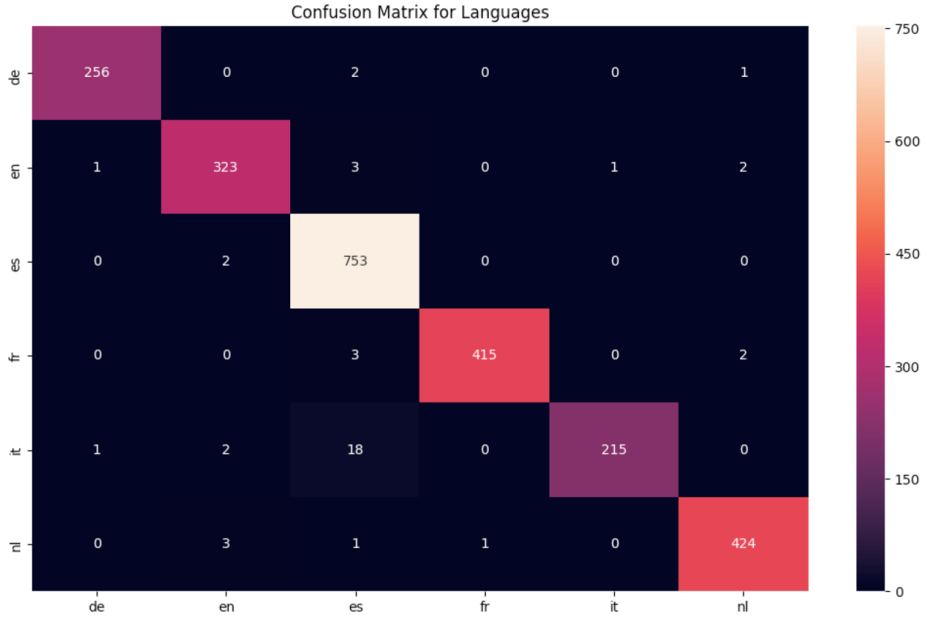


While with Logistic Regression I obtained the following:

```
Evaluating...
             precision    recall  f1-score   support

         de       0.99      0.99      0.99       259
         en       0.98      0.98      0.98       330
         es       0.97      1.00      0.98       755
         fr       1.00      0.99      0.99       420
         it       1.00      0.91      0.95       236
         nl       0.99      0.99      0.99       429

avg / total       0.98      0.98      0.98      2429
```



Conclusion

As you can see from the confusion matrix, the model using Logistic Regression is more accurate. There's still some error with Spanish-Italian (which is expected due to their similarity) but using Logistic Regression is better in this problem. I believe this result is because of the model assumptions. Naïve Bayes assumes all features are conditionally independent so if some features are dependent then the prediction might be poor. On the other hand, Logistic Regression splits feature space linearly, it still works if some variables are correlated. This affects due to the char analyzer, the use of determined prefixes depends on the language causing these features to be related between them. Also because of the separation by sentences, they are correlated. Naïve Bayes is generative, modeling distribution of individual classes, but those classes are sentences and are correlated so Logistic Regression is better for this problem by being discriminative and learning the boundary between classes.

User/Installation Guide

You can find the User Guide of the project in the README file at the repo.

References

1. Evaluating language identification performance. (2016, November 15). Retrieved November 17, 2017, from https://blog.twitter.com/engineering/en_us/a/2015/evaluating-language-identification-performance.html
2. http://scikit-learn.org/stable/modules/naive_bayes.html
3. http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
4. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
5. https://en.wikipedia.org/wiki/Logistic_regression