**Project 3- Team 3 Analysis for the name gender classifier**

To build the best name gender classifier, we download the name package from nltk by using “nltk.download(name)”. Then we load the male names in “male.txt” and female names in “female.txt”, and print first 10 names out in the data frame form after we shuffle the names randomly. There are total 7944 names our project. From these names, we are going to divide the ***first 500*** names become the **test set**, ***500 to 1000*** names become the **dev-test set**, and the **training set** is the **last 6944 names**. To have the test set is critical for our project because we can use it to test our hypothesis before we jump to the training set.

* **Baseline ( Using the last letter of the names to predict the gender)**

After all the preparations, we used the Text book’s codes to help us build the basic structure for the name gender classifier. It is called the gender\_feature. We changed all the words into lower cases and print out the last letter of the names. With the experiment of dev-test set, we will get 75% accuracy to predict the gender of names by only looking at the last letter of the names. This is a good start.

* **Baseline + Last Trigram ( combing the last letter and the last 3 letters of the name to predict the gender)**

Then we add the last Trigram to the classifier and hope to increase the accuracy. We are adding the Last Trigram to the basic structure. As our example shows that “die” as the last 3 letters will give us the female name. **(Here is the fun fact, I can’t really have an example to show you that name ends with “die” will be a female name. I can only think of “Melodie”. You can also use it for Melody. I guess the name that ends with die is too “dark” for many people. Who does want to give a name to the new born baby that end with “die”? If it happened to you, no offense. )**

Anyway, with the effort we make, it increase to 77% now. With our example, it also makes sense. As I can think of “Melodie”, it also could be “Eddie” that is a male name. Since we can easily find out the counter example for this feature, the accuracy will not increase significantly.

* **Baseline + Last Trigram+ First Trigram + First Four Grams ( this combination really tells you that only 2% increase is not enough. We need more)**

Since the last letter and last 3 letters give us a good estimate, we start to combine them with the first letters. Like the example, clearly, it gives us “Cindie”. As I can think of, “Cin” mostly appear in the female names such as one of the famous princess from Disney—“Cinderella”. Now, the accuracy increases to 85%.

* **Baseline + Last Trigram+ First Trigram + First Four Grams + Entropy**

To add “Entropy” feature to our classifier, it doesn’t make our result better. It decreases to 82%. To find a reason to persuade us for the decrease, we can think about the Type I error and Type II error. Before “Entropy”, we have 85% accuracy that does not mean that it is actually 85% in the reality. Maybe there are “errors” existing in our classifier when it fetches for the gender and name. There is some “off” information contained in our dataset.

For Type I error, for instance, under the category of male names, our classifier identified some female names as male names.

For Type II error, for example, under the category of male names, our classified identified the male names as female names and put them under the female category.

The entropy feature helps us to correct some of our false positive and false negative. Our accuracy level is more reliable.

* **Now we are going to look at Entropy and our model performance**

We use the sample code from text book to test how the entropy works. By the textbook, the formula of Entropy is

For example for the line of entropy(‘male’, ‘female’, ‘male’, ‘male’), to calculate it, we will have entropy = label for males and label for female= = .811278

Then we will do the entropy for input labels of names, and we get about .9510 for our test. It is close to 1 and it is considered high. When entropy is high, the input values have highly varied labels. For our case, entropy of labels in the name gender classifier depends on the ratio of male to female names. When the entropy is high, there are many labels with a medium frequency. Probability of labels and log of probability of labels are not small. It means that our data set has a relatively balanced gender labels for names.

Then we run the classifier to show that how many names misplaced in our classifiers. There are 89 names in total that we are misplaced. Around 18% error in our classifier. Then, we use “most\_informative\_features” to show the ratio of female and male for different last letter and last trigram. It gives us an informative idea about the prediction.

When we did some statistics for male and female, we get the mean for male is 2.32 with standard deviation is 0.35 and the mean for female is 2.27 with standard deviation is 0.37. Standard deviation is relatively small that is good sign for our data set. It means that the data set varies less. For example, some names such as “Alex”, “Francis” and so on, they may be with female labels and male labels, because they are unisex. With mean 2.32 for male, it means that for names with same first trigram/four gram, last trigram and last letter, around 2.32 names under the male category and 2.27 names under the female category. The histograms for both genders also show that they are approximately normal distribution.

* **Testing with real data**

We will test how well the classifier is for predicting the male vs. female with the top 1000 names from the US social Security Administration for each year since 1880. From the plot, it is easy to tell that before 1900, our classifier still a good estimator since it is above 80% but it drops significantly in 1920 and after 2000. Even though it drops, it still above 78% that is still considered as a good estimator.