NLP Lab 7

Dixitha Kasturi
dkasturi@syr.edu

Topic: Text Classification

Part 1

a) Using the gender_features function defined in Lab1 (Part 1) as a reference, define a new feature extraction function that includes features for two-letter suffixes,
b) Make a new gender_features function that keeps three suffix letters and report the accuracy. Be sure to make allowances if any names that are only two characters long. Alternatively, make a gender_features function that uses the first letter and the last two letters.

For this question I took the 2 cases with 2 and 3 features respectively.

a) For 2 features the accuracy was 80.2%
The features function was modifed as follows:

```
def gender_features2(word):
    return{'last_letter': word[-1],'secondlast_letter':word[-2]}
```

```
print(gender_features2('Shrek'))
```

```
{'last_letter': 'k', 'secondlast_letter': 'e'}
```

The accuracy was as follows:

```
# classify accuracy function runs the classifier on the test set and reports
#   comparisons between predicted labels and actual/gold labels
print("Accuracy for 2 features classifier = ", nltk.classify.accuracy(classifier2, test_set2))
```

```
Accuracy for 2 features classifier =  0.802
```

b) When I tried to run the classifier, I initially got the following error, I suspected that the out of bounds index meant that there were names which were only of length 2

```
#for 3 features
train_set3 = [(gender_features3(n), g) for (n, g) in train_names]
test_set3 = [(gender_features3(n), g) for (n, g) in test_names]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-69-44509f2c106a> in <module>
      1 #for 3 features
----> 2 train_set3 = [(gender_features3(n), g) for (n, g) in train_names]
      3 test_set3 = [(gender_features3(n), g) for (n, g) in test_names]

<ipython-input-69-44509f2c106a> in <listcomp>(.0)
      1 #for 3 features
----> 2 train_set3 = [(gender_features3(n), g) for (n, g) in train_names]
      3 test_set3 = [(gender_features3(n), g) for (n, g) in test_names]

<ipython-input-49-786bb8496896> in gender_features3(word)
      3
      4 def gender_features3(word):
----> 5     return{'last_letter': word[-1],'secondlast_letter':word[-2],'thirdlast_letter':word[-3]}
      6

IndexError: string index out of range
```

To verify if my suspicion is correct, I calculated the lengths of each word and put them in a list and got the frequency of each of those lengths, unsurprisingly, there were 19 words with length = 2

```
lengths =[len(x[0])for x in namesgender] #getting length of each name
lengths.sort() # sorting the lengths

#Getting frequency of each name length
f = {}
for i in lengths:
    if i in f:
        f[i] += 1
    else:
        f[i] = 1

print(f)
```

```
{2: 19, 3: 272, 4: 926, 5: 1878, 6: 2049, 7: 1447, 8: 846, 9: 351, 10: 116, 11: 24, 12: 10, 13: 3, 14: 1, 15: 2}
```

I printed out the list of those names :

```
#List of names with length 2
names_len2 = [a for a in namesgender if len(a[0])==2]
names_len2
```

```
[('Hy', 'male'),
 ('Jo', 'female'),
 ('Jo', 'male'),
 ('Em', 'female'),
 ('Bo', 'male'),
 ('Di', 'female'),
 ('Cy', 'male'),
 ('Er', 'male'),
 ('Ev', 'male'),
 ('Ez', 'male'),
 ('Si', 'male'),
 ('Bo', 'female'),
 ('La', 'female'),
 ('Ki', 'female'),
 ('Ag', 'female'),
 ('Ed', 'male'),
 ('Al', 'male'),
 ('Vi', 'female'),
 ('Ty', 'male')]
```

To train my classifier model with 3 features, I excluded out the above 2 letter words/names and further confirmed that only 3 and above length words are present in the new list to give to the model:

```
# If we get rid of these names, our feature engineering for last 3 letters will work

#removing len 2 names
namesgender3 = [x for x in namesgender if len(x[0])>2]
lengths3 =[len(x[0])for x in namesgender3] #getting length of each name
lengths3.sort() # sorting the lengths

#Getting frequency of each name length
f3 = {}
for i3 in lengths3:
    if i3 in f3:
        f3[i3] += 1
    else:
        f3[i3] = 1

print(f3)

# We now build the training and testing set using gender features on the new names set
# separate the names into training and test
train_names3 = namesgender3[500:]
test_names3 = namesgender3[:500]
train_set3 = [(gender_features3(n), g) for (n, g) in train_names3]
test_set3 = [(gender_features3(n), g) for (n, g) in test_names3]
```

```
{3: 272, 4: 926, 5: 1878, 6: 2049, 7: 1447, 8: 846, 9: 351, 10: 116, 11: 24, 12: 10, 13: 3, 14: 1, 15: 2}
```

The accuracy then was 78%

```
print("Accuracy for 3 features classifier = ", nltk.classify.accuracy(classifier3, test_set3))
```

```
Accuracy for 3 features classifier =  0.78
```

Part 2

I chose 1350 and 5050 as my common words length for this question. I did play around with the numbers and observed that as the number is closer to the 2000 mark, the accuracy did not vary a lot or it was the same. But if random shuffling was performed the accuracies varied.

For 1350 count the accuracy was 73%

For 5050 count the accuracy was 79%

```
# get the 2000 most frequently appearing keywords in the corpus
word_items_less2k = all_words.most_common(1350)
word_features_less2k = [word for (word, freq) in word_items_less2k]   # just the words

word_items_more2k = all_words.most_common(5050)
word_features_more2k = [word for (word, freq) in word_items_more2k]
```

```
# training using naive Baysian classifier with a 95/5 split
train_set_less2k, test_set_less2k = featuresets_less2k[100:], featuresets_less2k[:100]
classifier_less2k = nltk.NaiveBayesClassifier.train(train_set_less2k)

# evaluate the accuracy of the classifier
print (nltk.classify.accuracy(classifier_less2k, test_set_less2k))
# the accuracy result may vary since we randomized the documents
```

```
0.73
```

```
# training using naive Baysian classifier with a 95/5 split
train_set_more2k, test_set_more2k = featuresets_more2k[100:], featuresets_more2k[:100]
classifier_more2k = nltk.NaiveBayesClassifier.train(train_set_more2k)

# evaluate the accuracy of the classifier
print (nltk.classify.accuracy(classifier_more2k, test_set_more2k))
# the accuracy result may vary since we randomized the documents
```

```
0.79
```