Open in Colab

Project 3 - Name Gender Classifier

by Gracie Hui Han and Don Padmaperuma

Assignment

Using any of the three classifiers described in chapter 6 of Natural Language Processing with Python, and any features you can think of, build the best name gender classifier you can.

Begin by splitting the Names Corpus into three subsets: 500 words for the test set, 500 words for the dev-test set, and the remaining 6900 words for the training set.

Then, starting with the example name gender classifier, make incremental improvements.

Use the dev-test set to check your progress. Once you are satisfied with your classifier, check its final performance on the test set.

How does the performance on the test set compare to the performance on the dev-test set? Is this what you'd expect?

In [1]:
```
import collections
import nltk
nltk.download('names')
from nltk.corpus import names
from nltk.classify import apply_features
from nltk.metrics import ConfusionMatrix, accuracy, precision, recall, f_measure
import random
```

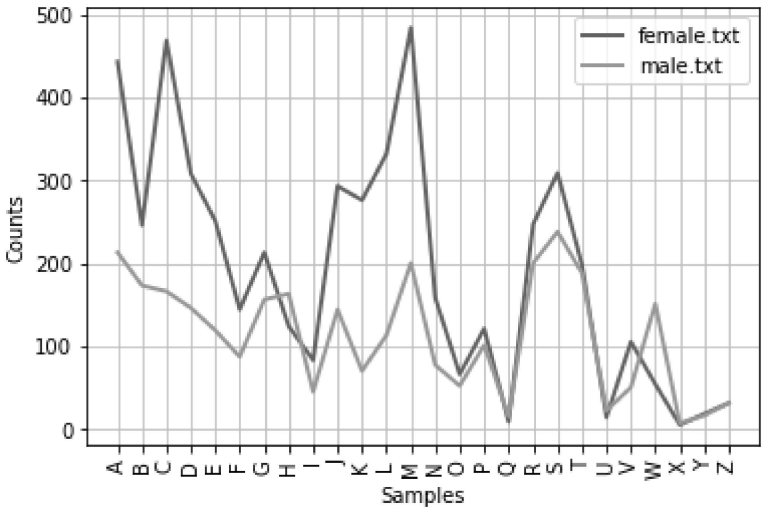
```
[nltk_data] Downloading package names to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package names is already up-to-date!
```

In [2]:
```
len(names.words('female.txt')), len(names.words('male.txt'))
```

Out[2]:
```
(5001, 2943)
```

We can see from above numbers that there are more female names (almost twise as male names) compared to the male names in the dataset.

In [3]:
```
# Check distribution of the first letter of the name.
gender_freq = nltk.ConditionalFreqDist((fileid, name[0])
  for fileid in names.fileids()
  for name in names.words(fileid))
gender_freq.plot()
```



Out[3]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>

Above graph shows the comparison of distribution of beginning letter of the names by gender. We can see that more female names starts with letter A, C, M and S.

In [4]:
```
# combine male and female names and shuffle them.
names = ([(name, 'male') for name in names.words('male.txt')] + \
        [(name, 'female') for name in names.words('female.txt')])
random.shuffle(names)
```

In [5]:
```
# Take a look at the combined data
names[1:5]
```

Out[5]:
```
[('Nanette', 'female'),
 ('Neddie', 'male'),
 ('Magdaia', 'female'),
 ('Tillie', 'female')]
```

In [6]:
```
# unique names
len(set(item[0] for item in names))
```

Out[6]: 7579

We removed the names that was listed as both males and females and was listed twice.

In [7]:
```
# names that are not unique
names_only = [item[0] for item in names]
names_dist = nltk.FreqDist(names_only)
names_duplicates = [(k,v) for k,v in names_dist.items() if v>1 ]
names_duplicates [0:10]
```

Out[7]:
```
[('Sydney', 2),
 ('Clem', 2),
 ('Chris', 2),
 ('Henrie', 2),
 ('Kim', 2),
 ('Dominique', 2),
 ('Dion', 2),
 ('Britt', 2),
 ('Augustine', 2),
 ('Rey', 2)]
```

In [8]:
```
# Remove the duplicates and show total number of unique names
names2remove = [item[0] for item in names_duplicates]
final_names = [item for item in names if not item[0] in names2remove]
len(final_names)
```

Out[8]: 7214

In [9]:
```
names = final_names
len(names)
```

Out[9]: 7214

Split into Train, DevTest and Test Data

In [10]:
```
train_set = names[1000:]        # Training set
test_set = names[:500]   # Test
devtest_set = names[500:1000]   #development test
```

In [11]:
```
# Show size of the three subsets
print("Training Set = {}".format(len(train_set)))
print("Dev-Test Set = {}".format(len(devtest_set)))
print("Test Set = {}".format(len(test_set)))
```

```
Training Set = 6214
Dev-Test Set = 500
Test Set = 500
```

In [12]:
```
# Write a function for manually calculate the recall error and precision formance, for
#(to compare the automatically generated performance indicator)

def performance_metrics(model, train, digits=4):
    """Prints the precision and recall of an NLTK Naive Bayes model."""

    reference = collections.defaultdict(set)
    test = collections.defaultdict(set)

    for i, (features, label) in enumerate(train):
        reference[label].add(i)
        pred = model.classify(features)
        test[pred].add(i)

    m_precision = round(precision(reference['male'], test['male']), digits)
    f_precision = round(precision(reference['female'], test['female']), digits)

    m_recall = round(recall(reference['male'], test['male']), digits)
    f_recall = round( recall(reference['female'], test['female']), digits)

    print('Male precision: ', m_precision)
    print('Female precision: ', f_precision)
    print('Male recall: ', m_recall)
    print('Female recall: ', f_recall)
```

Features - First Feature, Letter Suffix Prefix, Vowels

The first feature contains the first letter, last letter, prefix and suffix, as well as the vowels (aeiou, we did not include the y in here which is optional)

In [13]:
```
def gender_features1(name):
    features = {}
    features["firstletter"] = name[0].lower()
    features["lastletter"] = name[-1].lower()
    features["suffix2"]= name[-2:].lower()
    features["preffix2"]= name[:2].lower()

    for letter in 'aeiou':   ## vowels
        features["count(%s)" % letter] = name.lower().count(letter)
        features["has(%s)" % letter] = (letter in name.lower())
    return features
```

In [14]:
```
featuresets = [(gender_features1(n), g) for (n,g) in names]
featuresets[0]
```

Out[14]:
```
({'firstletter': 'l',
  'lastletter': 'y',
  'suffix2': 'ny',
  'preffix2': 'le',
  'count(a)': 0,
  'has(a)': False,
  'count(e)': 1,
  'has(e)': True,
  'count(i)': 0,
  'has(i)': False,
  'count(o)': 0,
  'has(o)': False,
  'count(u)': 0,
  'has(u)': False},
 'male')
```

Split Data Based on 1st Feature

In [15]:
```
### Split data
train_set_fe = featuresets[1000:]
test_set_fe =featuresets[:500]
devtest_set_fe =featuresets[500:1000]
```

Classifier - NaiveBayes -on 1st Feature

In [16]:
```
classifier = nltk.NaiveBayesClassifier.train(train_set_fe)
print(classifier.classify(gender_features1('Stefani')))
print(classifier.classify(gender_features1('Steffan')))
```

```
female
male
```

In [17]:
```
# Show Accuracy
print("train_set: ", nltk.classify.accuracy(classifier, train_set_fe))
print("test_set: ", nltk.classify.accuracy(classifier, test_set_fe))
print("devtest_set: ", nltk.classify.accuracy(classifier, devtest_set_fe))
```

```
train_set:  0.8439008690054716
test_set:  0.812
devtest_set:  0.836
```

In [18]:
```
# Show important features
classifier.show_most_informative_features(5)
```

```
Most Informative Features
                 suffix2 = 'na'           female : male   =    143.7 : 1.0
              lastletter = 'k'              male : female =     65.8 : 1.0
                 suffix2 = 'us'             male : female =     64.3 : 1.0
                 suffix2 = 'la'           female : male   =     63.5 : 1.0
              lastletter = 'a'            female : male   =     54.4 : 1.0
```

In [19]:
```
# Check errors
errors = []
for (name, tag) in devtest_set:
    guess = classifier.classify(gender_features1(name))
    if guess != tag:
        errors.append( (tag, guess, name) )
```

In [20]:
```
for (tag, guess, name) in sorted(errors):
   print('correct=%-8s guess=%-8s name=%-30s' % (tag, guess, name))
```

```
correct=female   guess=male     name=Cherish
correct=female   guess=male     name=Corliss
correct=female   guess=male     name=Daloris
correct=female   guess=male     name=Danell
correct=female   guess=male     name=Debby
correct=female   guess=male     name=Delores
correct=female   guess=male     name=Devan
correct=female   guess=male     name=Fortune
correct=female   guess=male     name=Gabbey
correct=female   guess=male     name=Gayleen
correct=female   guess=male     name=Gilligan
correct=female   guess=male     name=Glad
correct=female   guess=male     name=Honey
correct=female   guess=male     name=Ivory
correct=female   guess=male     name=Ivy
correct=female   guess=male     name=Jacquelin
correct=female   guess=male     name=Jocelyn
correct=female   guess=male     name=Jojo
correct=female   guess=male     name=Jonis
correct=female   guess=male     name=Kerrin
correct=female   guess=male     name=Kerstin
correct=female   guess=male     name=Lucy
correct=female   guess=male     name=Marj
correct=female   guess=male     name=Mildred
correct=female   guess=male     name=Millisent
correct=female   guess=male     name=Miriam
correct=female   guess=male     name=Modesty
correct=female   guess=male     name=Mommy
correct=female   guess=male     name=Nanon
correct=female   guess=male     name=Nonah
correct=female   guess=male     name=Norean
correct=female   guess=male     name=Pearl
correct=female   guess=male     name=Piper
correct=female   guess=male     name=Prudence
correct=female   guess=male     name=Renel
correct=female   guess=male     name=Romy
correct=female   guess=male     name=Rosaleen
correct=female   guess=male     name=Ruthe
correct=female   guess=male     name=Sheelagh
correct=female   guess=male     name=Shirl
correct=female   guess=male     name=Starr
correct=female   guess=male     name=Susann
correct=female   guess=male     name=Tamar
correct=female   guess=male     name=Tory
```

```
correct=female   guess=male     name=Wandie
correct=male     guess=female   name=Allah
correct=male     guess=female   name=Allin
correct=male     guess=female   name=Alwin
correct=male     guess=female   name=Anurag
correct=male     guess=female   name=Avi
correct=male     guess=female   name=Bartie
correct=male     guess=female   name=Beale
correct=male     guess=female   name=Cole
correct=male     guess=female   name=Cy
correct=male     guess=female   name=Dabney
correct=male     guess=female   name=Davy
correct=male     guess=female   name=Demetre
correct=male     guess=female   name=Dickie
correct=male     guess=female   name=Esau
correct=male     guess=female   name=Felipe
correct=male     guess=female   name=Herbie
correct=male     guess=female   name=Irvine
correct=male     guess=female   name=Ishmael
correct=male     guess=female   name=Jeremy
correct=male     guess=female   name=Kalle
correct=male     guess=female   name=Keil
correct=male     guess=female   name=Lawrence
correct=male     guess=female   name=Lennie
correct=male     guess=female   name=Manny
correct=male     guess=female   name=Maximilian
correct=male     guess=female   name=Maxwell
correct=male     guess=female   name=Merrel
correct=male     guess=female   name=Mikael
correct=male     guess=female   name=Mike
correct=male     guess=female   name=Mordecai
correct=male     guess=female   name=Natale
correct=male     guess=female   name=Raphael
correct=male     guess=female   name=Salomone
correct=male     guess=female   name=Sebastian
correct=male     guess=female   name=Siffre
correct=male     guess=female   name=Tarrance
correct=male     guess=female   name=Wittie
```

In [21]:
```
print("Error count: ", len(errors))
```

```
Error count:  82
```

In [22]:
```
train_set1 = [(gender_features1(n), g) for (n,g) in train_set]
devtest_set1 = [(gender_features1(n), g) for (n,g) in devtest_set]
test_set1 = [(gender_features1(n), g) for (n,g) in test_set]

nb1 = nltk.NaiveBayesClassifier.train(train_set1)

print('Devset accuracy is')
print(nltk.classify.accuracy(nb1, devtest_set1))
print('Test accuracy is')
print(nltk.classify.accuracy(nb1, test_set1))
```

```
Devset accuracy is
0.836
Test accuracy is
0.812
```

In [23]:
```
performance_metrics(nb1, train_set1)
```

```
Male precision:  0.7651
Female precision:  0.8913
```

```
Male recall:  0.8088
Female recall:  0.8632
```

In [24]:
```
performance_metrics(nb1, devtest_set1)
```

```
Male precision:  0.775
Female precision:  0.8767
Male recall:  0.8073
Female recall:  0.8539
```

Classifier - DecisionTree-on 1st Feature

In [25]:
```
classifier_tree = nltk.DecisionTreeClassifier.train(train_set_fe)

print("train_set: ", nltk.classify.accuracy(classifier_tree, train_set_fe))
print("test_set: ", nltk.classify.accuracy(classifier_tree, test_set_fe))
print("devtest_set: ", nltk.classify.accuracy(classifier_tree, devtest_set_fe))
```

```
train_set:  0.9763437399420662
test_set:  0.81
devtest_set:  0.82
```

Feature2 - Last Letter Only

In [26]:
```
def first_letter(name):
    name = name.lower()
    return {
        'first_1_letter': name[0]
        }
first_letter("Mary")
```

Out[26]:
```
{'first_1_letter': 'm'}
```

In [27]:
```
train_set2 = [(first_letter(n), g) for (n,g) in train_set] ###
devtest_set2 = [(first_letter(n), g) for (n,g) in devtest_set] ###
test_set2 = [(first_letter(n), g) for (n,g) in test_set] ###

nb1b = nltk.NaiveBayesClassifier.train(train_set2)  ###

print('Devset accuracy is')
print(nltk.classify.accuracy(nb1b, devtest_set2)) ###
print('Test accuracy is')
print(nltk.classify.accuracy(nb1b, test_set2)) ###
```

```
Devset accuracy is
0.646
Test accuracy is
0.668
```

In [28]:
```
performance_metrics(nb1b, devtest_set2) ###
```

```
Male precision:  0.6923
Female precision:  0.6421
Male recall:  0.1406
Female recall:  0.961
```

In [29]:
```
performance_metrics(nb1b, train_set2) ###
```

```
Male precision:  0.6271
Female precision:  0.6676
```

```
Male recall:  0.1364
Female recall:  0.9553
```

In [30]:
```
# Looking at the most informative features
nb1b.show_most_informative_features(10)  ###
```

```
Most Informative Features
          first_1_letter = 'w'              male : female =      5.7 : 1.0
          first_1_letter = 'q'              male : female =      3.0 : 1.0
          first_1_letter = 'k'            female : male   =      2.5 : 1.0
          first_1_letter = 'h'              male : female =      2.3 : 1.0
          first_1_letter = 'u'              male : female =      2.2 : 1.0
          first_1_letter = 'x'              male : female =      2.1 : 1.0
          first_1_letter = 'y'              male : female =      1.9 : 1.0
          first_1_letter = 'l'            female : male   =      1.9 : 1.0
          first_1_letter = 'c'            female : male   =      1.8 : 1.0
          first_1_letter = 'z'              male : female =      1.7 : 1.0
```

In [31]:
```
classifier_tree = nltk.DecisionTreeClassifier.train(train_set2)

print("train_set: ", nltk.classify.accuracy(classifier_tree, train_set2))
print("test_set: ", nltk.classify.accuracy(classifier_tree, test_set2))
print("devtest_set: ", nltk.classify.accuracy(classifier_tree, devtest_set2))
```

```
train_set:  0.6644673318313485
test_set:  0.668
devtest_set:  0.646
```

Feature - Last 2 Letters

In [32]:
```
## FEATURE - LAST TWO LETTERS
def last_2_letters(name): ###
    name = name.lower()
    return {
        'last_2_letters': name[-2:]
        }
last_2_letters("Andy")
```

Out[32]:
```
{'last_2_letters': 'dy'}
```

In [33]:
```
train_set3 = [(last_2_letters(n), g) for (n,g) in train_set] ###
devtest_set3 = [(last_2_letters(n), g) for (n,g) in devtest_set] ###
test_set3 = [(last_2_letters(n), g) for (n,g) in test_set] ###

nb3 = nltk.NaiveBayesClassifier.train(train_set3)  ###

print('Devset accuracy is')
print(nltk.classify.accuracy(nb3, devtest_set3)) ###
print('Test accuracy is')
print(nltk.classify.accuracy(nb3, test_set3)) ###
```

```
Devset accuracy is
0.822
Test accuracy is
0.804
```

In [34]:
```
performance_metrics(nb3, train_set3) ###
```

```
Male precision:  0.8264
Female precision:  0.8365
Male recall:  0.6729
Female recall:  0.9221
```

In [35]:
```
performance_metrics(nb3, devtest_set3) ###
```

```
Male precision:  0.8323
Female precision:  0.8174
Male recall:  0.6719
Female recall:  0.9156
```

In [36]:
```
# Looking at the most informative features
nb3.show_most_informative_features(10)
```

```
Most Informative Features
          last_2_letters = 'na'           female : male   =    143.7 : 1.0
          last_2_letters = 'us'             male : female =     64.3 : 1.0
          last_2_letters = 'la'           female : male   =     63.5 : 1.0
          last_2_letters = 'ra'           female : male   =     53.7 : 1.0
          last_2_letters = 'ia'           female : male   =     48.3 : 1.0
          last_2_letters = 'rt'             male : female =     47.8 : 1.0
          last_2_letters = 'ta'           female : male   =     39.4 : 1.0
          last_2_letters = 'do'             male : female =     25.4 : 1.0
          last_2_letters = 'rd'             male : female =     24.6 : 1.0
          last_2_letters = 'ld'             male : female =     20.9 : 1.0
```

Feature 4- Combo of Letters Vowels

In [37]:
```
def gender_features2nd(name):
    name = name.lower()
    return{
        'first_letter': name[0].lower(),
        'first2_letter': name[0:2].lower(),
        'first3_letter': name[0:3].lower(),
        'last_letter': name[-1].lower(),
        'last2_letter': name[-2:].lower(),
        'last3_letter': name[-3:].lower(),
        'last_vowel': (name[-1] in 'aeiou')
    }
```

In [38]:
```
gender_features2nd("Mary")
```

Out[38]:
```
{'first_letter': 'm',
 'first2_letter': 'ma',
 'first3_letter': 'mar',
 'last_letter': 'y',
 'last2_letter': 'ry',
 'last3_letter': 'ary',
 'last_vowel': False}
```

Vectorize the Features (Feature)

In [39]:
```
import numpy as np

# Vectorize the features function
features = np.vectorize(gender_features2nd)
#print(features(['Rose', 'Mike']))
print(features(['Mary', 'Joe']))
```

```
[{'first_letter': 'm', 'first2_letter': 'ma', 'first3_letter': 'mar', 'last_letter':
'y', 'last2_letter': 'ry', 'last3_letter': 'ary', 'last_vowel': False}
 {'first_letter': 'j', 'first2_letter': 'jo', 'first3_letter': 'joe', 'last_letter':
'e', 'last2_letter': 'oe', 'last3_letter': 'joe', 'last_vowel': True}]
```

In [40]:
```
# Extract the features for entire dataset
X = np.array(features(names))[:, 0] # X contains the features

# Get the gender column
y = np.array(names)[:, 1]           # y contains the targets

print("Name: %s, features=%s, gender=%s" % (names[0][0], X[0], y[0]))
```

```
Name: Lenny, features={'first_letter': 'l', 'first2_letter': 'le', 'first3_letter': 'le
n', 'last_letter': 'y', 'last2_letter': 'ny', 'last3_letter': 'nny', 'last_vowel': Fals
e}, gender=male
```

In [41]:
```
# Shuffle and split:  train, dev-test, test
from sklearn.utils import shuffle
X,y = shuffle(X,y)

X_test, X_dev_test, X_train = X[:500], X[500:1000], X[1000:]
y_test, y_dev_test, y_train = y[:500], y[500:1000], y[1000:]

print("test: " , len(X_test))
print("devtest: ", len(X_dev_test))
print("train: ", len(X_train))
```

```
test:  500
devtest:  500
train:  6214
```

In [42]:
```
# Use vectorizer to transform the features into feature-vectors.
from sklearn.feature_extraction import DictVectorizer

#print(features(['Rose', 'Mike']))
print(features(['Mary', 'Joe']))

# train the vectorizer to know the possible features and values.
vectorizer = DictVectorizer()
vectorizer.fit(X_train)

transform = vectorizer.transform(features(['Mary', 'Joe']))
print(transform)
print(type(transform))
print(transform.toarray()[0][12])
print(vectorizer.feature_names_[12])
```

```
[{'first_letter': 'm', 'first2_letter': 'ma', 'first3_letter': 'mar', 'last_letter':
'y', 'last2_letter': 'ry', 'last3_letter': 'ary', 'last_vowel': False}
 {'first_letter': 'j', 'first2_letter': 'jo', 'first3_letter': 'joe', 'last_letter':
'e', 'last2_letter': 'oe', 'last3_letter': 'joe', 'last_vowel': True}]
  (0, 128)      1.0
  (0, 984)      1.0
  (0, 1518)     1.0
  (0, 1732)     1.0
  (0, 1871)     1.0
  (0, 3111)     1.0
  (0, 3113)     0.0
  (1, 109)      1.0
  (1, 840)      1.0
  (1, 1515)     1.0
  (1, 1691)     1.0
  (1, 2341)     1.0
  (1, 3092)     1.0
  (1, 3113)     1.0
<class 'scipy.sparse.csr.csr_matrix'>
```

```
0.0
first2_letter=ap
```

Classifier - DecisionTree¶ On Vectorized Transformed Data

In [43]:
```
from sklearn.tree import DecisionTreeClassifier
# DT classifier to extract discriminating rules from the features.
DT_classifier = DecisionTreeClassifier()

DT_classifier.fit(vectorizer.transform(X_train), y_train)
```

Out[43]:
```
DecisionTreeClassifier()
```

In [44]:
```
print(DT_classifier.predict(vectorizer.transform(features(["Sebastian", "Amy"]))))
```

```
['male' 'female']
```

In [45]:
```
# Accuracy
print("Accuracy on training set: ", DT_classifier.score(vectorizer.transform(X_train),
print("Accuracy on dev-test set: ",DT_classifier.score(vectorizer.transform(X_dev_test)
print("Accuracy on test set: ",DT_classifier.score(vectorizer.transform(X_test), y_test
```

```
Accuracy on training set:  0.9982298036691342
Accuracy on dev-test set:  0.856
Accuracy on test set:  0.842
```

Cross Validation

In [46]:
```
# cross validation
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score

pred_train = cross_val_predict(DT_classifier, vectorizer.transform(X_train), y_train, c
pred_dev_test = cross_val_predict(DT_classifier, vectorizer.transform(X_dev_test), y_de
pred_test = cross_val_predict(DT_classifier, vectorizer.transform(X_test), y_test, cv =

score_train = accuracy_score(y_train, pred_train)
score_dev_test = accuracy_score(y_dev_test, pred_dev_test)
score_test = accuracy_score(y_test, pred_test)

print("Cross Validation")
print("Train Score = {0:5f}".format(score_train))
print("Dev Test Score = {0:5f}".format(score_dev_test))
print("Test Score = {0:5f}".format(score_test))
```

```
Cross Validation
Train Score = 0.847119
Dev Test Score = 0.774000
Test Score = 0.784000
```

In [47]:
```
## Summary
#https://www.nltk.org/book/ch06.html
```