Report - Surname Assignment

Aaron Alarcon & Oscar Galindo

Dr. Nigel Ward

Topics in Language Processing

September 10, 2020

Surname Assignment

Introduction:

For this assignment we were provided with two different datasets. The first which is called ***surnames-dev.csv***, this file data which was used as our *training* data. The file ***test_surnames.txt*** was used as the *testing/evaluation* data. In this case, as instructed in the assignment's description we were supposed to make a classifier based on the classification for Russian surnames.

Design and Implementation:

For this assignment we implemented a vectorization of the surnames. This was achieved by having an array that represented the count of letters found in every surname. For example, 'Molina' had the following array: [1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], with this normalized mass distribution among the variables:[166666667e-01, 0, 0, 0, 0, 0, 0, 0,4.62962963e-03, 0, 0, 7.71604938e-04, 2.14334705e-05, 2.77777778e-02, 128600823e-04, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]. You can notice that for the normalization we implement 27 variables for every representation and that is because we treat upper case letters and lower case letters as equal as long as they represent the same variable. Furthermore, any special special character outside of the classic 'a'-'z' range is added as part of the last variable in the vector.

Our classifier is also opened for the classification in all languages one can provide from the training one has to change none but one variable, which is the language variable. In all cases we implemented a 'look-up' of the optimal threshold for classification by calculating the best f score of the model using any threshold from the range [0.01,99] (both extreme ends are inclusive). After finding such threshold we use it to produce theoretically the best results in the testing data.

Experimental Results:

- Weights learned after training on the ***surnames-dev.csv*** data:

```
################### Weights ######################
Intercept: -0.06540065754994062
Weights: [-3.12022044e-04  1.45808340e-01 -7.95598998e-02  5.84858350e-02
  1.51511123e-02  1.58987131e-01  6.40364557e-02  1.54890442e-01
  9.21359735e-02  1.87682929e-01  1.83675364e-01  1.76534181e-02
  6.67972388e-02  1.13476507e-01  5.18350151e-03  9.94998592e-02
 -1.76335977e-01 -2.35561565e-02 -7.34123810e-03  6.63085491e-02
  6.39046204e-02  4.59443448e-01 -4.99345334e-02 -2.21432218e-01
  1.61608368e-01  1.66457128e-01 -5.72100627e-02]
##################################################
```

- "Optimal" Threshold calculated: 0.41
- Training set results:

```
################### Train Set ###################
              precision    recall  f1-score   support

         0.0       0.86      0.82      0.84      1593
         1.0       0.81      0.84      0.82      1408

    accuracy                           0.83      3001
   macro avg       0.83      0.83      0.83      3001
weighted avg       0.83      0.83      0.83      3001

[[1307  286]
 [ 221 1187]]
################################################
```

- Testing set results:

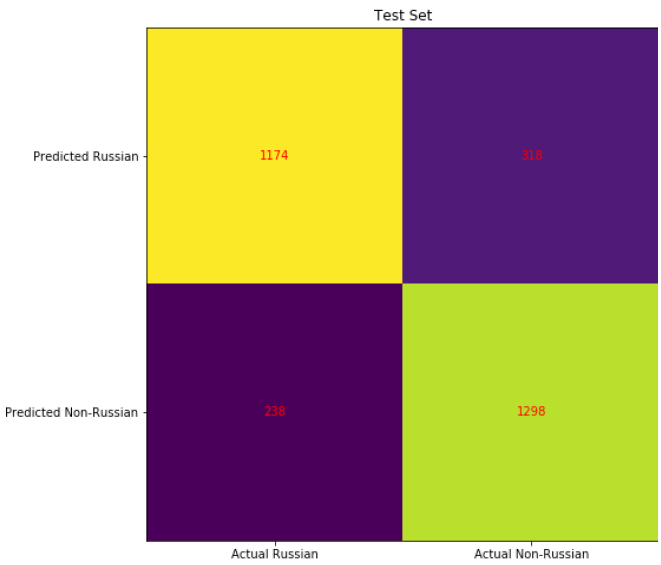
```
################### Test Set ####################
              precision    recall  f1-score   support

         0.0       0.85      0.80      0.82      1616
         1.0       0.79      0.83      0.81      1412

    accuracy                           0.82      3028
   macro avg       0.82      0.82      0.82      3028
weighted avg       0.82      0.82      0.82      3028

[[1298  318]
 [ 238 1174]]
################################################
```

- Confusion Matrix with Notes:



Conclusions:

- Based on the results we observed during this experimentation we can conclude this method seems much more accurate, more precise, and more capable of recalling on the data sets that were provided to us (i.e. ***surnames-dev.csv*** and ***test_surnames.txt***)

- **Rationale for False Positives**: In this case we selected ***Lawniczak*** which is a Polish surname. In this case we assume that the structure and syntax of both Russian and Polish are similar, at least on the data we were provided for training, and so this could be a reasonable explanation for the misclassification. In other words, this example could be an outlier.

- **Rationale for False Negatives**: In this case we selected **Balanowski** which is indeed a Russian last name. For this one we also think it might be an outlier as it might be encoded in a vector that resembles the vectors for other languages. In other words, this surname might be confusing the network because it is too similar to other non-russian examples.

Appendix (Python 3.8):

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Sep  4 11:56:43 2020

@authors: oscargalindo and aalarcon
"""
import numpy as np, matplotlib.pyplot as plt
import copy
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

def read_file(src):
    f = open(src,encoding='utf-8').readlines()
    l = []
    for e in f:
        l.append(e.strip('\n').split(','))
    return l

def vectorize(list_of_pairs,country):
    y = np.zeros((len(list_of_pairs),))
    x = np.zeros((len(list_of_pairs),27))
    xlabels = []
    i = 0
    for pair in list_of_pairs:
        vector = encode(pair[0])
        y_label = 1 if pair[1] == country else 0

        #setting variables
```

```
        y[i] = y_label
        x[i] += vector
        xlabels.append(pair[0])
        i+=1
    return xlabels,x,y

def train(model,x_train,y_train):
    model.fit(x_train,y_train)

def predict(model,x):
    return model.predict(x)

def classify(predicted,threshold):
    predicted[predicted < threshold] = 0
    predicted[predicted >= threshold] = 1

def print_weights(model):
    print("#################### Weights ######################")
    print("Intercept:",model.intercept_)
    print("Weights:",model.coef_)
    print("###################################################")

def get_stats(actual,prediction,language,title):
    print("####################",title,"####################")
    conf_m = confusion_matrix(actual, prediction)
    print(classification_report(actual,prediction))
    print(conf_m)
    print("###################################################")
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.set_title(title)
    ax.imshow(conf_m)
    ax.grid(False)
    ax.xaxis.set(ticks=(0, 1), ticklabels=('Actual '+language,'Actual Non-'+language))
    ax.yaxis.set(ticks=(0, 1), ticklabels=('Predicted '+language, 'Predicted Non-'+language))
    ax.set_ylim(1.5, -0.5)
    for i in range(2):
        for j in range(2):
            ax.text(i, j, conf_m[i-1, j-1], ha='center', va='center', color='red')
    plt.show()
    plt.close()

def print_to_file(y_pred, y_act, x_labels, title,language):
    results = open("{}DataResults.txt".format(title), "w", encoding = "utf-8")

    #setting up string holders for each category
    template = "{}\nName\tPrediction\n"
    true_pos = template.format("****************** True Positives ******************")
    true_neg = template.format("****************** True Negatives ******************")
    false_pos = template.format("****************** False Positives ****************")
    false_neg = template.format("****************** False Negatives ****************")

    lineformat = "{}, {}\n"
    for i in range(len(y_act)):
        line = lineformat.format(x_labels[i], y_pred[i])
```

```
        #determining which file to write to
        if y_act[i] != y_pred[i]:
            if y_pred[i] == 1:
                false_pos = false_pos + line
            else:
                false_neg = false_neg + line
        else:
            if y_pred[i] == 1:
                true_pos = true_pos + line
            else:
                true_neg = true_neg + line

    results.write(true_pos)
    results.write(true_neg)
    results.write(false_pos)
    results.write(false_neg)
    results.close()

#Takes in a name and inserts each character into its slot
#and normalizes the vector
def encode(name):
    vector = np.zeros(27, dtype = float)
    for character in name:
        index = ord(character.lower()) - ord('a')
        #For characters falling outside of normal range
        if index < 0 or index > 25:
            vector[-1] +=1
        else:
            vector[index]+=1
        #Normalize
        # vector = vector / len(name)
    return vector

def get_f(y_act, y_pred):
    pos_act = np.argwhere(y_act)
    pos_pred = np.argwhere(y_pred)

    true_pos = np.intersect1d(pos_act, pos_pred)
    try:
        precision = float(true_pos.shape[0]/pos_pred.shape[0])
    except:
        precision = 0
    recall = float(true_pos.shape[0]/pos_act.shape[0])

    #print("Precision: %.2f\t Recall: %.2f" % (precision, recall))
    try:
        return 2 * ((precision * recall)/(precision + recall))
    except:
        return float("-inf")

def test_thresholds(model, y_act, y_pred):
    best_f = 0
    best_t = 0
    for ind  in range(1, 100, 1):
```

```
        threshold = .01  * ind
        y_pred_copy = copy.deepcopy(y_pred)
        y_pred_copy[y_pred_copy < threshold] = 0
        y_pred_copy[y_pred_copy >= threshold] = 1
        f_meas = get_f(y_act, y_pred_copy)
        if f_meas > best_f:
            best_f = f_meas
            best_t = threshold
    return best_t

if __name__=="__main__":
    language = 'Russian'
    model = LinearRegression()
    ################Train###################
    src = './surnames-dev.csv'
    l = read_file(src)
    x_labels_tr,x,y = vectorize(l,language)
    train(model,x,y)
    print_weights(model)
    y_p = predict(model, x)
    threshold = test_thresholds(model, y, y_p)
    print("Best Threshold:",threshold)
    classify(y_p, threshold)
    get_stats(y, y_p, language,"Train Set")
    print_to_file(y_p, y, x_labels_tr, "Training",language)
    #################Test###################
    src = './test_surnames.txt'
    l_t = read_file(src)
    x_labels_te,x_t,y_t = vectorize(l_t,language)
    y_p = predict(model,x_t)
    classify(y_p, threshold)
    get_stats(y_t,y_p,language,"Test Set")
    print_to_file(y_p, y_t, x_labels_te, "Testing",language)
    ########################################
```