Instituto Politécnico Nacional

Escuela Superior de Cómputo

Regresión Logística Ordinal

Natural Language Processing

Estudiante:
Nicolás Sayago Abigail

Profesora:
Olga Kolesnicova

Mayo 19, 2020

1 Resultados

La primera parte muestra solo algunos resultados comparando la predicción y el valor real. La segunda parte tiene la matriz de confusión y la tercera las métricas obtenidas con el paquete de métricas implementado en python.

```
-------------- COMPARATION --------------
Prediction: 3  Real: 3
Prediction: 3  Real: 3
Prediction: 3  Real: 3

------------ CONFUSION MATRIX -----------
[[ 1 20  2  0  0]
 [ 0 11 36  0  0]
 [ 0  4 84  5  0]
 [ 0  1 50 19  0]
 [ 0  0 24 20  1]]


---------------- METRICS ----------------
              precision    recall  f1-score   support

           1       1.00      0.04      0.08        23
           2       0.31      0.23      0.27        47
           3       0.43      0.90      0.58        93
           4       0.43      0.27      0.33        70
           5       1.00      0.02      0.04        45

    accuracy                           0.42       278
   macro avg       0.63      0.29      0.26       278
weighted avg       0.55      0.42      0.34       278
```

✓Código fuente

```
import nltk
import re
import math
import numpy as np
import mord
from bs4 import BeautifulSoup
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn import metrics

##############################################
#                          GETTING CORPUS TEXT
##############################################
def getText(corpusRoot, code, n):
        reviews = list()
        for i in range(2, n):
                try:
                        # print("--->", str(i))
                        f = open(corpusRoot + str(i) + ".review.pos",
                        ↪  encoding = code) #Cod: utf-8, latin-1
                        text = f.readlines()
                        tokens = [nltk.word_tokenize(line) for line in text]
                        review = list()
                        for line in tokens:
                                if len(line) > 0:
                                        review.append(line[1])
                        # print("Asi recibimos review:", review)
                        reviews.append(review)
                        f.close()
                except:
                        continue
        return reviews

def getRank(corpusRoot, code, n):
        ranks = list()
        for i in range(2, n):
                try:
                        print("--->", str(i))
                        f = open(corpusRoot + str(i) + ".xml")
                        lines = f.readlines()
                        j = lines[0].index( ' rank=' )
```

```
                        ranks.append(int(lines[0][j+7]))
                        f.close()
                except:
                        continue
        return ranks

def removeStopwords(tokens, language):
        sw = stopwords.words(language)
        clean = []
        for review in tokens:
                text = ''
                # print("------original:", review)
                for word in review:
                        if word not in sw:
                                text = text + word + " "
                # print("Así queda sin stopwords:", text)
                clean.append(text)
        return clean

##############################################
##############################################
##############################################

# Get Tokens by corpus
fpath =
↪  '/Users/abiga/Desktop/AbiiSnn/GitHub/Natural-Language-Processing/Practice/24/corpus
code = 'ISO-8859-1'
n = 4382
reviews = getText(fpath, code, n)
cleanReviews = removeStopwords(reviews, 'english') # List of string

# print(reviews[:3])
# print(cleanReviews[:3])

ranks = getRank(fpath, code, n)
# print(ranks[:5])

# TF-IDF
vectorizer = TfidfVectorizer(norm='l2', smooth_idf=True, use_idf=True)
vec = vectorizer.fit_transform(cleanReviews)

X = np.round(vec.todense(), 2)
Y = np.array(ranks)
print(len(X))
```

```
print(len(Y))

n = 3600
trainingX = np.array(X[:n])
trainingY = np.array(Y[:n])
testX = np.array(X[n:])
testY = np.array(Y[n:])

c = mord.LogisticIT()
c.fit(trainingX, trainingY)

Ypredict = c.predict(testX)
print(len(Ypredict))
print(len(testY))

print("")
print("")
print("-------------- COMPARATION --------------")
for i in range(0, len(Ypredict)):
        if i % 100 == 0:
                print("Prediction:", Ypredict[i], " Real:", testY[i])
print("")

print("------------ CONFUSION MATRIX -----------")
matrix = metrics.confusion_matrix(testY, Ypredict)
print(matrix)
print("")
print("")

print("---------------- METRICS ----------------")
target_names = ['1', '2', '3', '4', '5']
print(metrics.classification_report(testY, Ypredict, target_names =
↪  target_names))
```