T.C.

GEBZE TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

DATA MINING
CSE-454

FINAL PROJECT

Öğrencinin Adı ve Soyadı: Samet GÜLMEZ

Öğrencinin Numarası : 161044110

My project TEXT CATEGORIZATION. I separate the text files I have according to the classes I have specified. I have 5 classes. MAGAZINE, ECONOMY, SPORTS, POLITICS, HEALTH. By comparing the input text, it is found out which class it belongs to. I used the K-Nearest Neighbor (KNN) algorithm in my project.

DATASET

I have a total of 1100 texts for the dataset of this project. There are 230 files in each category. I use 219 of 230 files in each category to fill my vector files. I give the rest of my files as input to the program and find out which category it is in.

PREPROCESSİNG

```
STOPWORDS = "da,de, ve , vs. , ile , mi , mu "
def remove_stopwords(text):
    """custom function to remove the stopwords"""
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
```

Here I clean the data in my text files. The cleaner and simpler my data is, the clearer the results are. I categorize text in my project. Words like "de, da, and, etc." in texts do not mean anything to me. So I remove them from the text and make them get better results.

TFIDF

I used TFIDF method to convert my words to a numerical value. So my numbers became a numerical value and I managed to aggregate them. All the text files in the 5 categories I have were browsed and the vectors of all the words in them were extracted with TFIDF.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \qquad \mathrm{tf}(t, d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

...

TF: Term Frequency
IDF: Inverse Document Frequency

Term frequency; is the division of our selected term by the total number of terms found in the text.

Reverse Document Frequency; It shows how many terms we have in our texts. It is the logarithm of our total text quantity to the text quantity that contains the term.

KNN

With KNN, basically the closest points to the new point are searched. K represents the amount of the closest neighbors of the unknown point. We choose the quantity k (usually an odd number) of the algorithm to predict the results.

In K-NN classification, the output is class membership. An object is classified by the majority vote of its neighbors; the object is given to the class most common among its closest neighbors (k is a small positive integer). If k = 1, the object is simply assigned to that nearest neighbor's class.

I found the similarities of these vectors I created by putting them into the cosSim operation. Then I put the output values into the sort operation. Then, according to the sorted values, I returned the classification value that has the most in the first k value.

POST PROCESSİNG

F1 score method was used for postprocessing. I ran my 10 test files. these 10 files have real results in my hand. Then I list the results of this KNN algorithm. I create a Confusion Matrix using Actual and predict results that I have.

Then I read the matrix and find the True Positive, True Negative, False Positive and False negative values.

Then I find the scores of Accuracy , Precision and Recall with these values.

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{(True Positive + True Negative + False Positive + False Negative)}} = \frac{5+8}{5+8+4+3} = \frac{13}{20} = 0.65$$

Accuracy Score

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}} = \frac{8}{8+3} = \frac{8}{11} = 0.72$$

Precision

$$\text{Recall} = \frac{\text{True Positive}}{(\text{True Positive + False Negative})} = \frac{8}{8+4} = \frac{8}{12} = 0.67$$

Finally, using all these scores, I get my f1 score.

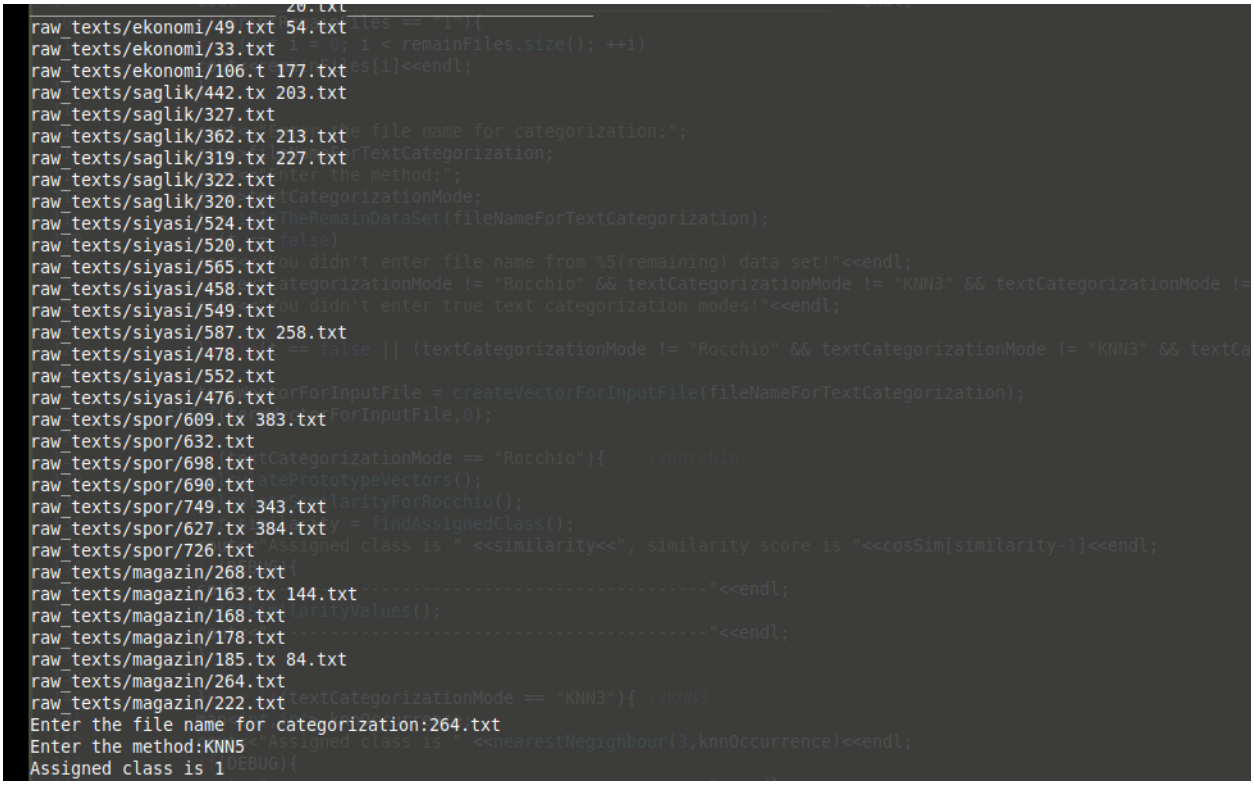
$$\text{F1 - score} = 2 * \frac{\text{Precision * Recall}}{\text{Precision + Recall}} = 2 * \frac{0.72 * 0.67}{0.72 + 0.67} = 2 * \frac{0.48}{1.39} = 0.68$$

f1 - score

Thus, I found the f1 scores of all my classes.

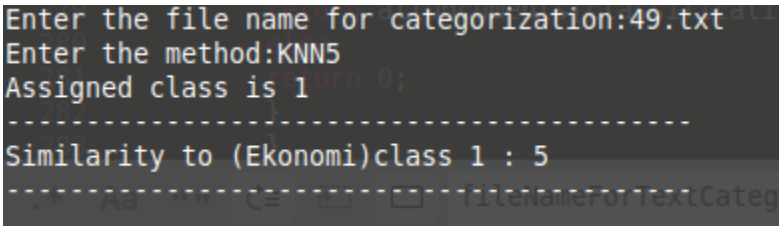
TEST

The following files are my remaining files for testing



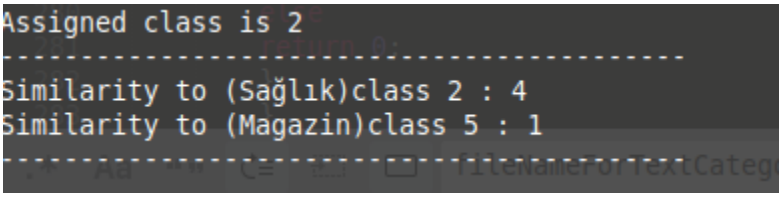
İnput : 49.txt

49.txt is normally in economy class. The result after running with the KNN algorithm is as follows.



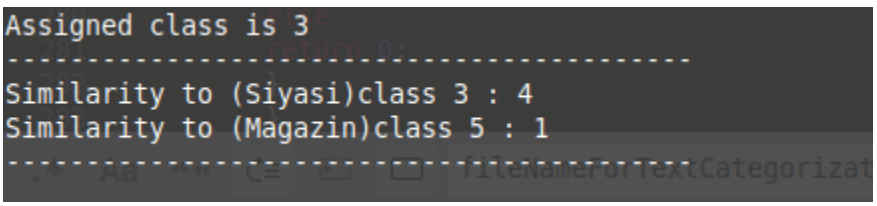
İnput : 327.txt

49.txt is normally in Healthy class. The result after running with the KNN algorithm is as follows.



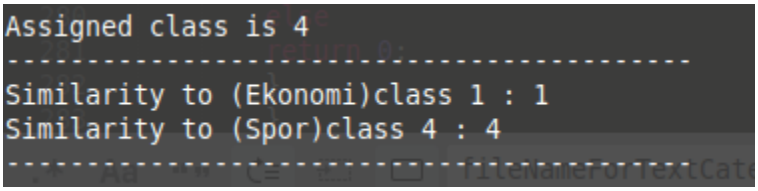
İnput : 476.txt

476.txt is normally in Politic class. The result after running with the KNN algorithm is as follows.



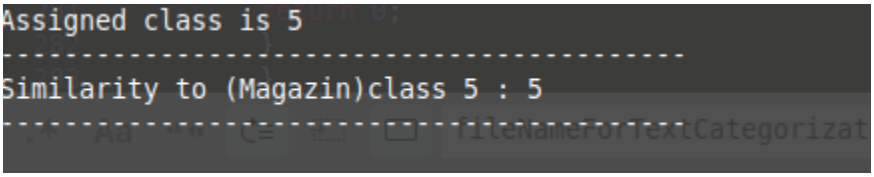
İnput : 698.txt

698.txt is normally in Sport class. The result after running with the KNN algorithm is as follows.



İnput 178.txt.

178 .txt is normally in Magazine class. The result after running with the KNN algorithm is as follows.



POST PROCESSİNG TEST

```
################################ WELCOME ################################
Processing files...raw_texts/ekonomi/49.txt
raw_texts/saglik/320.txt
raw_texts/siyasi/565.txt
raw_texts/spor/632.txt
raw_texts/magazin/178.txt
raw_texts/ekonomi/33.txt
raw_texts/saglik/322.txt
578.txt
raw_texts/spor/749.txt
raw_texts/magazin/185.txt
Real result : 1   knnRsult : 1
Real result : 2   knnRsult : 2
Real result : 3   knnRsult : 2
Real result : 4   knnRsult : 4
Real result : 5   knnRsult : 5
Real result : 1   knnRsult : 1
Real result : 2   knnRsult : 1
Real result : 3   knnRsult : 1
Real result : 4   knnRsult : 4
Real result : 5   knnRsult : 5
--------------------Conflusion Matrix-----------------
2 0 0 0 0
1 1 0 0 0
1 1 0 0 0
0 0 0 2 0
0 0 0 0 2
Ekonomi ---->  Recall : 1  Precision : 0.5 f1Score : 0.666667
Saglık ----->  Recall : 0.5  Precision : 0.5 f1Score : 0.5
Siyasi -----> Recall : 0  Precision : 1 f1Score : 0
Spor ------> Recall : 1  Precision : 1 f1Score : 1
Magazain --->  Recall : 1  Precision : 1 f1Score : 1
Program closing...
################################ THE END ################################
```