

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.

## PREPROCESSING

```
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.

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

...

**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 PROCESSING

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.

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

Accuracy Score

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

Precision

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

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

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

f1 - score

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

## TEST

The following files are my remaining files for testing

```
raw_texts/ekonomi/49.txt 54.txt files == 1){
raw_texts/ekonomi/33.txt 1 = 0; 1 < remainFiles.size(); ++i)
raw_texts/ekonomi/106.txt 177.txt es[i]<<endl;
raw_texts/saglik/442.txt 203.txt
raw_texts/saglik/327.txt
raw_texts/saglik/362.txt 213.txt file name for categorization:";
raw_texts/saglik/319.txt 227.txt TextCategorization;
raw_texts/saglik/322.txt Enter the method:";
raw_texts/saglik/320.txt CategorizationMode;
raw_texts/siyasi/524.txt TheRemainDataSet(fileNameForTextCategorization);
raw_texts/siyasi/520.txt false)
raw_texts/siyasi/565.txt you didn't enter file name from '%5(remaining) data set!'"<<endl;
raw_texts/siyasi/458.txt categorizationMode != "Rocchio" && textCategorizationMode != "KNN3" && textCategorizationMode !=
raw_texts/siyasi/549.txt you didn't enter true text categorization modes!"<<endl;
raw_texts/siyasi/587.txt 258.txt
raw_texts/siyasi/478.txt == false || (textCategorizationMode != "Rocchio" && textCategorizationMode != "KNN3" && textCa
raw_texts/siyasi/552.txt
raw_texts/siyasi/476.txt orForInputFile = createVectorForInputFile(fileNameForTextCategorization);
raw_texts/spor/609.txt 383.txt ForInputFile,0);
raw_texts/spor/632.txt
raw_texts/spor/698.txt CategorizationMode == "Rocchio"){ // Rocchio
raw_texts/spor/690.txt createPrototypeVectors();
raw_texts/spor/749.txt 343.txt similarityForRocchio();
raw_texts/spor/627.txt 384.txt = findAssignedClass();
raw_texts/spor/726.txt Assigned class is " <<similarity<<, similarity score is "<<cosSim[similarity-1]<<endl;
raw_texts/magazin/268.txt {
raw_texts/magazin/163.txt 144.txt .....<<endl;
raw_texts/magazin/168.txt similarityValues();
raw_texts/magazin/178.txt .....<<endl;
raw_texts/magazin/185.txt 84.txt
raw_texts/magazin/264.txt
raw_texts/magazin/222.txt textCategorizationMode == "KNN3"){ // KNN3
Enter the file name for categorization:264.txt
Enter the method:KNN5 Assigned class is " <<nearestNeghbour(3,knnOccurrence)<<endl;
Assigned class is 1 DEBUG}
```

Input : 49.txt

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

```
Enter the file name for categorization:49.txt
Enter the method:KNN5
Assigned class is 1
-----
Similarity to (Ekonomi)class 1 : 5
-----
```

Input : 327.txt

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

```
Assigned class is 2
-----
Similarity to (Sağlık)class 2 : 4
Similarity to (Magazin)class 5 : 1
-----
```

Input : 476.txt

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

```
Assigned class is 3
-----
Similarity to (Siyasi)class 3 : 4
Similarity to (Magazin)class 5 : 1
-----
```

Input : 698.txt

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

```
Assigned class is 4
-----
Similarity to (Ekonomi)class 1 : 1
Similarity to (Spor)class 4 : 4
-----
```

Input 178.txt.

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

```
Assigned class is 5
-----
Similarity to (Magazin)class 5 : 5
-----
```



## POST PROCESSING 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 knnResult : 1
Real result : 2 knnResult : 2
Real result : 3 knnResult : 2
Real result : 4 knnResult : 4
Real result : 5 knnResult : 5
Real result : 1 knnResult : 1
Real result : 2 knnResult : 1
Real result : 3 knnResult : 1
Real result : 4 knnResult : 4
Real result : 5 knnResult : 5
-----Confusion 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
Saglik -----> 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 #####
```

