Project 03: Training a Classifier

Tolga Özdemir 35320066566

Department of Computer Engineering, TED University

CMPE414: Information Retrieval

Asst. Prof. Tayfun Küçükyılmaz

January 17, 2021

K-NN Classifier

For the implementation of K-NN, first, I needed to get the frequencies of whole words in each document. To do that, I edited the code of project 1 to get the document, “radikal.doc_sorted.terms” like in figure 1. In figure 1, The reason for the addition of the string form of terms such as “AA” is easily getting information to the dictionary data structure on Python. After the term name, the number, for example, 398 is the document id and the number of 1 is the frequencies of the term “AA” in the document with the id, 398. After that, I stored this into a dictionary data structure in the form of “{category: [[document_id, text], ...], ...}”. Then, I calculated the number of minimum documents that one category has since the instructor said that you can select N documents for each category and split the test and train data roughly equal. So, I found that the “forum” category has 10 documents. That’s why I get 5 documents from every category for the train set, in other words, the train set contains 70 documents.

Figure 1

Terms and their frequencies for each document.

```
AA  398 1   404 1   405 1   407 1   410 1   413 1   415 1   418 1   419 1   422 1   424 1   434
AAA 2250    1   6688    1
AACHEN  836 1
AADNE   4628    1
AAON    2584    1
AARON   3554    1
AB  135 1   149 3   297 2   848 1   943 1   992 1   1018    2   1034    14  1069    7   1117
ABA 1754    1   2335    1
ABABA   1822    1
ABAC    1173    5
ABACA   1477    3   1548    10  1667    2   1863    2
ABACI   2694    1   3015    3   3205    1   5479    1   6368    3
ABACUS  6658    1
ABALI   4790    1
ABANA   1138    1   4601    1   5053    1
ABANOZ  4335    1
```

After these processes, I created the vector form of the train set. I used sparse representation for it. So, instead of 0s and 1s, it wrote the frequencies of terms in that

document. For the whole test data, I first, vectorized the single test data, then, calculated the seventy distances that are the distances between the train set and this single test data. After that, I sorted this distance list in ascending order, chose the first k distance, and labeled the most frequent category in that k distance to them.

After the K-NN algorithm, I run my K-NN code and sklearn K-NN with different k values 10 times for 250 test data and I get the average accuracy of each of them. I get these accuracies: for k=3, my K-NN gets 0.09 accuracy, whereas, sklearn K-NN gets 0.08; for k=5, my K-NN gets 0.08 accuracy, whereas, sklearn K-NN gets 0.06; for k=7, my K-NN gets 0.0828 accuracies, whereas, sklearn K-NN gets 0.0768. So, this proves that my K-NN algorithm works well and no huge differences with the sklearn K-NN algorithm. The reason for low accuracy may be caused by the number of training set since it contains 70 documents.

Rocchio Classifier

To implement this, I used the sklearn library on Python since I have no time to write this code because of the projects and homework from other lectures. I run the sklearn Rocchio classifier 10 times for 250 test data and I get the average accuracy of each of them. So, it gets the accuracy of the number, 0.17 when the size of the training set is 70. To try that it depends on whether the train size or not, I split the training set in the form of that training set contains fifty percent of the documents from each category. So, it gets the accuracy of the number, 0.37. Hence, we can say that if the training size increases, we get better accuracy.

Latent Dirichlet Allocation

To implement this, I used the gensim library on Python. First, it traversed the whole document and stores them into the “texts” list by tokenizing them. It constructed a dictionary using the corpora module of the gensim library. Then, it converts this dictionary into a bag of words by using the “doc2bow” function. The “corpus” variable holds this bag of words. I got

14 topics with 20 passes by using the LDA model. The purpose of this model is to find the given number of topics of given documents by giving weight to the words.

I checked the words in the topics found by the LDA model whether they are relevant to our category names or not. I think that the topics found by the model are similar to our categories. For example, one topic contains the words, "BESİKTAS, ATLETİZM, ALTIN, BRONZ, TRABZONSPOR, CİN, LİG, KILODA, TİYATROSU, OKAN". By looking at these words, we can predict that this topic is the category, "spor". Hence, I believe that it is similar in nature. Furthermore, these words in topics contain a lot of stopwords such as "COK, ICIN, ILE. VE" etc. So, if we remove stopwords, we can get better results.

Results

The major contributor to the errors of the K-NN algorithm is that it usually predicts the categories as "anasayfa" (main-page). The reason for this is that the main page in nature includes all the news from the other categories. So, eliminating this problem with a two-level classifier as the main-page and not the main-page is an excellent solution. If we did this, I believed that we can get better accuracy. The other errors and suggestions of KNN are that:

1. The training set is so small if we want to get N documents from each category since the "forum" category has 10 documents. This causes accuracy to decrease.
2. To eliminate the first error that I mentioned, we can store the X percentage of documents to the training set in each category. As I did in the Rocchio Classifier, we get better accuracy. However, this leads data to be unbalanced since, in the training set, the number of documents in each category is different now. I think that due to this, I can not get a higher accuracy of more than 0.37 in the Rocchio Classifier part.
3. The documents in the training set can include fewer words such as the document with three words. This leads to these documents not to represent their category efficiently.

So, to prevent this, we can eliminate the data which have less than X words in the preprocessing part of the corpus.

4. Eliminating stopwords and, maybe, the normalization would be a better idea. The reason for this is that in the LDA part, I observed that the topics found by the model include stopwords that have high weight for that topic. So, this showed that if we remove stopwords, we can get better accuracy.