Alper Canberk Balcı

Cmpe493 Project3 Report

1. Describe the steps you have performed for data preprocessing.  
   1- The main.py program takes the data and reads it in the main function  
   2- It strips the useless line at the top.  
   3- It finds the id of the document with NEWID.

4- It takes the title and body, and preprocess them with case folding, punctuation removal, and stop word removal. Then, it makes them lists of words  
5- It fills the inverted index for tf.idf scores that are used in KNN, and that’s pretty much it for the training, when we do testing, we will be calculating tf.idf scores with these.

6- Before these it also do the learning of parameters for Naïve Bayes algorithm. It processes topics, finds the topics, max 10 topics, multilabelled documents.

It also finds if it is a train and test data, finds nk , n, and at the end finds the vocabulary. I finds the P(cj) also, via topics\_train and topics\_test. Thus, we would be ready for testing the Naïve Bayes.

**Size of my resulting vocabulary: 47420**

1. Training set: [('earn', 2877), ('acq', 1650), ('money-fx', 539), ('grain', 434), ('crude', 391), ('trade', 369), ('interest', 347), ('wheat', 212), ('ship', 198), ('corn', 183)]

Test set: [('earn', 1088), ('acq', 719), ('crude', 189), ('money-fx', 180), ('grain', 149), ('interest', 133), ('trade', 118), ('ship', 89), ('wheat', 71), ('corn', 56)]

sorted\_id\_topics\_training\_set

sorted\_id\_topics\_test\_set

length of these lists hold: What is the total number of documents in the training set and the total number of documents in the test set.

And their length are: 6494, 2548 respectively for train and test set sizes.

**How many documents are labeled with more than one of the top 10 classes: 622 in training**

So estimation: 10% of the data is multilabelled.

1. Train size is 6494 so, if I choose splitting the train data as development set, I can choose first 1200 of these. 1200/6494 = 18.5% (almost, it is 18.478..). However, I was already late submitting, so I will do this part(developing by coding) on my own later. Instead, I chose parameters for multilabelling in NB and k for KNN, on my own, by trying values by hand.

And here it is how I did it:

For NB I chose a value beta = 1.04. I sorted the probabilities of the classes my classifier finds, then normalized them into an interval of [0,1]. Than I took the min value, which corresponds to the predicted class and I calculated a fraction. The fraction is other values for classes divided by min value. If it is less than beta = 1.04, I take that class as a predicted answer as well, and my prediction became multilabelled. Thinking about how 10% of the documents are multilabelled in the training set. This value seemed alright. However I needed to improve it by hand, and try different values for beta.

For beta=1.04

Text

Description automatically generated

For beta = 1.03

Text

Description automatically generated

For beta = 1.025

Text

Description automatically generated

For beta = 1.021

Text

Description automatically generated

For beta = 1.017

Text

Description automatically generated

For beta = 1.014

Text

Description automatically generated

As we can see when I decrease the beta value precision increases, recall decreases. If we look at F1 scores, for macro it increases, however micro avg F score fluctuates. If I have to choose a beta value among these according to micro and macro avg F scores, I should choose beta = 1.021, as it generally gives good results overall. However if macro avg F score and precision scores are the correct ones to choose, beta = 1.014 is more successful. So, I assume the latter, and choose my beta as 1.014. I didn’t care about recall much.

My strategy is pretty successful for this dataset, but I don’t see how it would perform with other datasets.

I am running KNN with K = 5. I assumed it. My strategy is that:

There will be 5 IDs of training documents selected by the KNN for a specific test document. I put all the labels, with repetition, into a set. The maximum count of label in the list is the first class to be predicted. Also, if other labels’ counts are greater than 50% of the count of the first predicted label, than they are also accepted, and the prediction becomes multilabelled.

1. Evaluation Results:

**For Naïve Bayes:**

For beta = 1.021

Text

Description automatically generated

**For beta = 1.014 (the one I deemed better)**

Text

Description automatically generated

**For KNN:**

My KNN algorithm works correctly as it can be seen via the screenshot below. However, I did run the algorithm for only 14826 new\_id document. When I run 2500 test documents program works for a loooong time. My pc is not that great, so I am leaving running it with the full data set to you, professor. It found trade as the answer from 5 trade train data set, where trade is the correct answer for the  
 id= 14826 test document

Text

Description automatically generated

Therefore, I cannot do evaluations on KNN, but I am almost sure about its accuracy to be as high as Naïve Bayes as we know they are similar strong performance algorithms.

Text

Description automatically generated