CS54701 Information Retrieval
Project 1 Latent Semantic Indexing

Penghao Wang

wang1128@purdue.edu

February 22, 2016

1 Programming Report

1.1 Introduction

The aim of this project is to build a retrieval engine by using the concept of tf-idf and Latent Semantic Indexing. Tfidf is the short for term frequency inverse document frequency. It is used to evaluate how important a term is to a document in a collection. Latent Semantic Indexing use SVD to retrieve query to "concept".

1.2 Program and Instructions

1.2.1 TFIDFMatrix.cpp

There are two main program that I write. First one is the TFIDFMatrix.cpp. The aim of this program is to build tf-idf by using the Indri package that is given. In this program, there are two functions.

1. The first is the getTFIDF() function. In this function, by using the api provided by Indri package, it is easy to calculate the TF and IDF. The function return the value TF*IDF.
2. The second function is main() function. This function is basically to calculate the TF*IDF and output the result and the stem vocabulary into a list. The aim is to provide value to be used by following program.

1.3 LSI.py

The second part of the project is programmed by python. The import package is numpy and scipy.

1. The first step is to read the matrix that is stored previously. The file is the output.txt. In the file, the matrix of tf-idf is stored. The row number is the term number. The column number is the document number.

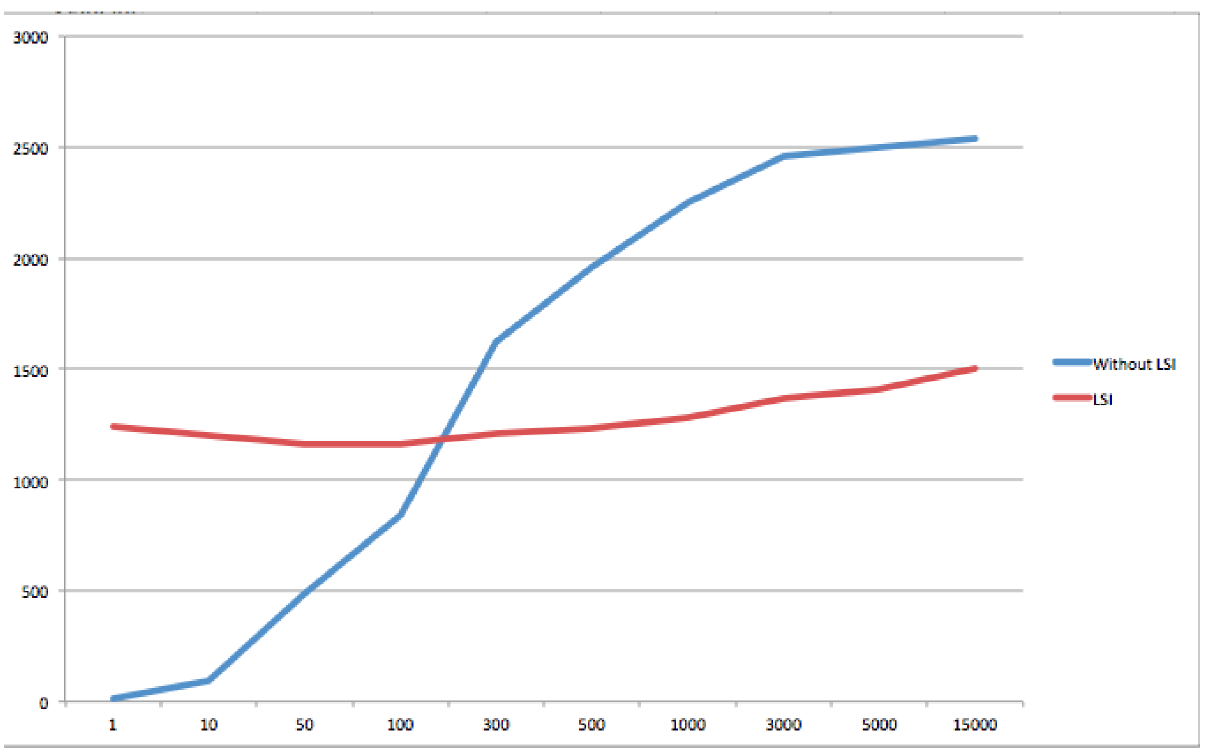
2. The second the step is the calculate the svd by using np.linalg.svd(). Three different matrix U, S, V are get by this step.
3. The next step is the calculate the new Matrix by the function setK(). There are four parameters in this function. They are U, S, V, k. The return result is newU, newS, newV. The program could get the new matrix by setK(). And they can be used for the following calculating.
4. Then we need to calculate the different cosine similarity and store them into a list. Then sort the list by descending. That means the top 5 document number is the most relevant one. In this program, the getCosine(ainS, U, V,count,q) function is to solve this problem. The parameter is ainS, U, V,count and q. The q is the query vector.
5. For the query vector,the length of the vector is the term. The original query vector is [0,0,0,...0]. When the word in query occurs in the vector, set that position to 1. getQuery() is design for this. The parameter is the list of query. It could calculate the right query vector.
6. Function tfcosineList(L,randomQuery) is similar to getCosine() function. This function is used to evaluate the cosine similarity between the query and the original tf-idf score. This function is to be design for evaluating the model compare with the tf-idf without LSI.
7. The getRandomQuery() function is used for generate the random queries. It is used for evaluating the engine we build.
8. The evaluateAveRecall(randomQuery) function is used to evaluate the average recall number by using random queries. It helps us to compare the LSI model with svd and the tf-idf that without the svd.
9. bestK() is to calculate the best K that works the best.

1.4 Evaluation

1.4.1 TF-IDF vs LSI

To evaluate resulting retrieval engine, there are several ways to do it. One baseline is the original TF-IDF without using LSI. First, I need use getRandomQuery(querynum,queryLen) function to generate 500 random queries in every different length of the term. Because, how many words in the query is one variable that affects the recall result. In this program, 1,10,50,100,300,500,1000,3000,5000,15000 and the length of the words in query. For each length, there are 500 random queries. In that way, there are 5000 queries are used to evaluate the engine.

The following picture represent the relationship:



The x-axis of the figure means the how many words in the query. The y-axis shows how many result is retrieved, which means the returned result. The red line represents the the retrieve model based on TF-IDF with LSI. The blue line represents the the model with out LSI. In another word, it did not do the svd. It only considers about terms but not the concepts.

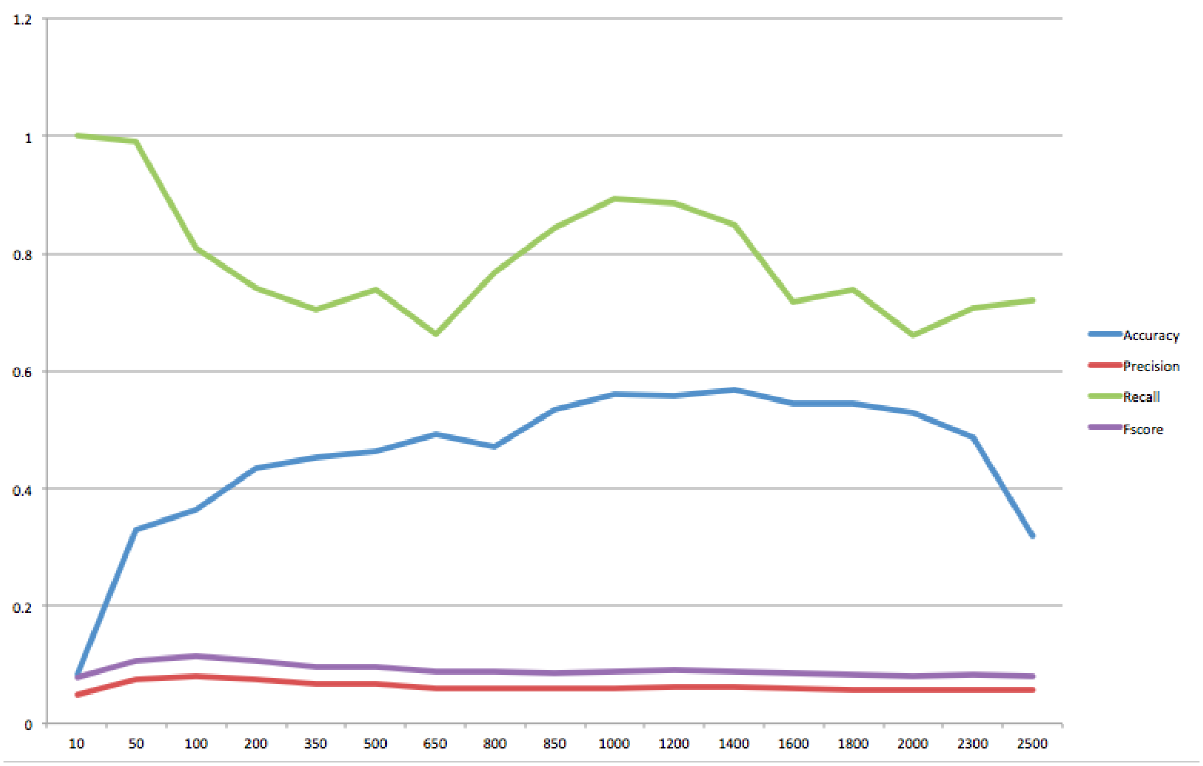
For the retrieve engine that doesn't use LSI. The performance is not that good. If the words in the query is not that much. It will return less number of the file. It basically return the documents that contain the terms. In that case, it increased a lot when the numbers words in query increase. In that case, the engine that only use TF-IDF is not sound.

For the retrieve engine we build that by using LSI. The result is reasonable. If there is only one word in the query, for instance, 'comput', it retrieves relative high number of the document. The reason is LSI focus on the concept but not the terms. In that case, the returned result won't change much as the previous model does.

As a conclusion, the LSI retrieve model will increase the performance of the research engine.

1.4.2 LSI vs Indri

In this section, I compare the engine we build to the Indri engine provided by the Indri. The assumption we make is that the results that retrieve by Indri are right. In theory, we should compare as many queries as we can. However, it is impossible that use Indri to do that because we need to use parameter.xml to do it. In that case, I compare 30 queries. In the future, this part should be improved. In this case, the term that will retrieve a lot of results and the term not are both selected. In this way, the result should be more reasonable. In this section, we calculate the precision, fscore and recall and calculate the average. The k value keep changing. Also, the accuracy is calculated. The accuracy means how many top result that retrieve by LSI engine are same with the Indri one. For instance, if there are 30 results that return by the Indri. The accuracy is the top results in the LSI engine that are same in top 30. Then divide the number by 30. The following picture represents the relationship.



The x- axis represents the k value. From the picture, we could know that the recall decrease when k increase and then increase when k is around 1000. The recall is about 0.9. The accuracy that I defined increase when k increase. When k is around 1000. The precision and F-score is low because I include the some words that is not easy to retrieve. For example, the Indri retrieve only one result, but LSI model retrieve 50. However, because we assume the Indri model is the benchmark, the precision is very low in this case. After evaluation, we found the best k is around 1000. Suppose we have $d1, d2, ..., d_N inS, andwewantsum(di^2)/sum(d^2) > 0.95$ the cutoff point is k =1117. From the result I got, when k is around 1000, the retrieve engine works the best.