Report on Search Engine(Information Retrieval)

Raj Bhatta
University of Windsor
401 Sunset Avenue
Windsor,Ontario
bhatt11y@uwindsor.ca

ABSTRACT

This report describes the details about Search Engine that we have developed. During the development of our project, we have started from basic search engine function such as building index and query processing. To add some feature to the search engine, we have added keyword highlighting feature and metadata search. Finally, the search engine contains some advanced algorithms such as page rank, Naive Bayes classification and K-Mean clustering algorithm along with vector data generated from Python

Keywords

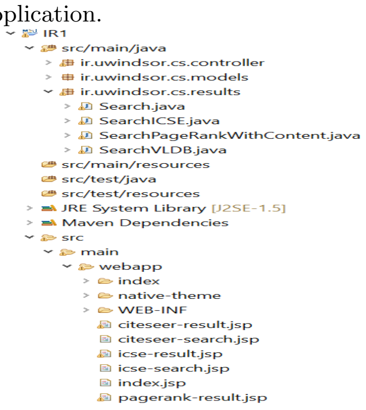
Sigmod,ICSE,VLDB,JDK,Doc2Vec,K-mean,pagerank,Naive Bayes classification

1. INTRODUCTION

Data repositories are growing day by day. So, the need for retrieving information from those data is important.In search engine project,we have implemented full text searching feature using various algorithms. We have used Apache Lucene as API. The *Apache Lucene* is a full-text search library in java,which makes it easy to add search functionality to an application or website. In this project, we have implemented different algorithms of Information Retrieval. The search engine is created using java servlet and JSP page.

2. CONFIGURATION AND SERVER SETUP

We have used Apache tomcat to run our application.There are lot of versions of Apache tomcat but we have used Apache tomcat 1.7[1]. To control Tomcat Server,we have setup tomcat user and we have uploaded our project on server in the form of WAR(Web Achieve file). Architecture of our Java application file is pretty simple. It contains JSP[2],CSS and Jascript file along with index file. The index file is created with the help of console Java application and they are implemented in the dynamic web project. All our Java code is located inside SRC directory and other file such as JSP,CSS and index file are located inside WebContent folder of our application.



```
IR1
  src/main/java
    ir.uwindsor.cs.controller
    ir.uwindsor.cs.models
    ir.uwindsor.cs.results
      Search.java
      SearchICSE.java
      SearchPageRankWithContent.java
      SearchVLDB.java
  src/main/resources
  src/test/java
  src/test/resources
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
    main
      webapp
        index
        native-theme
        WEB-INF
        citeseer-result.jsp
        citeseer-search.jsp
        icse-result.jsp
        icse-search.jsp
        index.jsp
        pagerank-result.jsp
```

3. DATA SOURCES

3.1 Citeseer Data Source

We have used different data sets for our application. The first data sets consists of 10,000 papers and that is in .txt file and .xml file.

3.2 Microsoft Academic Search Data

We have used ICSE,VLDB and SIGMOD Data sets released by Microsoft Academic Search.These data sets contains link in the form of citation graph which is very very useful for our search engine.

4. INDEXING AND SEARCHING

We have used Apache Lucene 5.2.1 for indexing and searching documents.

4.1 Indexing

Index organizes all the data so that it is quickly accessible. So,index consists of documents containing one or more fields.When Lucene indexes a document,it breaks it down into a number of terms. It then stores the terms in an index file where each term is associated with the documents that contain it. In our case, we have created index using java console application. First, we need to read the documents from file directory and pass that into `IndexWriter`. Finally, we have to add fields like `"title","author","contents" and "path"` to create index.

4.2 Searching

Searching is one of the important feature of the search engine and searching allow the end user to get desirable result. Apahce Lucene provides search on index so that searching is very fast and efficient. To make searching fast and efficient, various kinds of improvements is implemented in our search engine.

4.2.1 *Highlighting keyword*

Apahce Lucene provides an API to highlight the keyword in our search result. To highlight keyword, we have to store the term vector and their offsets in the index. The highlighter API will insert HTML Tag with attribute according to our desire, so that we can get result in our desiring format.

4.2.2 *Pagination*

The pagination helps to split the results into smaller chunk so that is it easier for user to get the result they are looking for. Here we have implemented pagination feature in our website. In our case every user can see only top 10 paper at once and this will reduce the server load thus accelerating the most relevant result to the user questions.Furthermore, we can control the limit of record using control given at the top of the page.

4.2.3 *fast Search*

It is slow process to search each and every time on index for each user query. Thus,we have implemented pre-data load technique in our server. The basic idea is to load data and store them in the memory with the help for servlet context initializer.

5. PAGE RANK

Page rank calculation process is the eminent process in search engine optimization. In our project, we have used `sigmod_subgraph.txt` and `pagerank_sigmod_id.tx` file and it contains the information required for link between the papers. The page ranking part consist of different stages like 1) calculating node 2) building adjacent matrix 3)building transition matrix 4)calculating page rank based on Markov Chain.

5.1 Detail Description

5.1.1 *Calculating Node*

we are provided with `sigmod_subgraph.txt` documents and that contains information required to link between papers.So, first we have to read this file, compute total number of node and save all data into `arrayList` to create adjacent matrix.

5.1.2 *Building adjacent matrix*

With nodes number and adjacent table we can create adjacent matrix among the papers.The adjacent matrix `AdjacentMatrix[i][j]` means, paper i have citation with paper j and the value adjacent matrix value is 1 if there is link otherwise it is set to 0. Then finally, with the help of transition matrix we can compute page rank value.

5.1.3 *Sorting page rank value*

We can sort the page rank value using different method. In the front end, we can extract all the numeric value and sort particular column of table using function given below: `dataTable( "order": [[ 0, "desc" ]])` or we can put our data in the map and sort the result using Java API.

5.2 Page Ranking of VLDB and ICSE

The page ranking algorithm is extended to calculate page rank of ICSE and VLDB documents. So, link between paper is provided from `sigmod_vldb_icse_subgraph.txt` documents. If we want to rank the ICSE documents then in that case, we have to input `icse_id.txt` and if we want to rank the VLDB documents then in that case, we have to input `vldb_id.txt`.

6. NAIVE BAYES CLASSIFICATION

We have used two sets of data for classification. First data set is from ICSE documents and second data set is from VLDB documents.Here we are classifying text using Multinominal Navies Bayes technique.So, Mutinominal Navies Bayes is machine Learning algorithm of supervised type that assume each and every feature is independent. Here we are classifying document,so, feature in our case is word.Thus by calculating probability of each word in the title belong to each class, and combine them together ,we can get the probability that title belong to a particular class. For this part we have used reference from group 3.

In the classification process, we need to provide train data and finally we need to test our data for classification. In our case, we have provided test data in the form of CSV file. The CSV file is generated from Java application and that is feed into application to classify it.The necessary code is attached in our web page with web link.

6.1 Training Data and Testing

During the training phase, we count the word in title of each document(VLDB and ICSE) and in the next step, we calculate the conditional probability.Finally, we converted our test document into CSV and we fed that into application for testing purpose.

The necessary formula for computing conditional probability is given below:

$$ConProb(word(i)) = word(i).getCountClassICSE() + 1/$$

$$\text{numberOfWordICSE} + \text{noUniqueWord} \quad (1)$$

7. CLUSTERING

For clustering, we have tested different kinds of algorithm available around the internet. The idea behind clustering data is that we need to convert our document to vector representation. The vector representation shows how a word or text usually is similar with other words or text.

Here in our case, we are creating cluster of ICSE and VLDB documents.We are going to use python to convert our document to vector(for this part,we have used code reference of Shane Peelar). We have tested many API[3] available around the Internet but we have successfully able to run the code in python.So, we will follow vector document generated by python to cluster the documents(VLDB,ICSE).

7.1 Converting Doc2Vec using Python

To convert our document to vector using Python, we have used PIP package manager.To convert document to vector using Python, we must have different kinds of library and finally, we have setup Python programming environment and we are successfully able to run the code using Python interpreter using library listed below.

7.1.1 *library required*

1.Numpy
2.Gensim

7.1.2 *Reading Doc File*

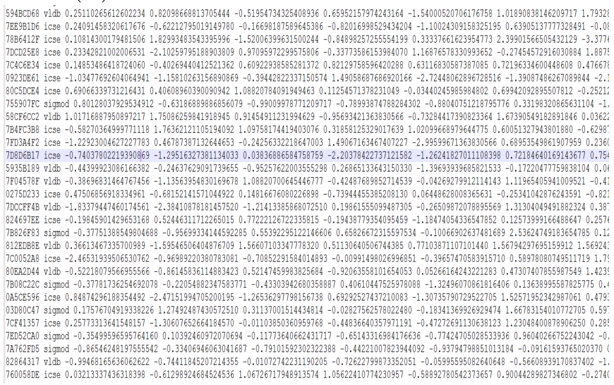
if args.model is not None:
print("Loading " + args.model)
model$_dm = Doc2Vec.load(args.model)$
elif args.train is not None:
files = args.train
tags = [] for f in files:
print("Reading " + f)
for line in open(f):
components = line.split("$\widehat{"}$) tags.append(TaggedDocument(copone
[components[0]]))

7.1.3 *building model*

print("Building Model for document")
model$_dm = Doc2Vec(documents = tags, alpha = 0.025, size =$
$90, workers = 4, dbow_words = 1, iter = 90, min_count = 2)$
In this case, we have make vector size as 90 and minimum count number is 2.

7.1.4 *Exporting to Vector space documents*

if args.dumpdocvectors
is not None and model_dm != "":
print("Writing document vectors to
file: " + args.dumpdocvectors)
f = open(args.dumpdocvectors, "w")
for i in
model$_dm.docvecs.offset2doctag$:
f.write(i + "$\widehat{"}$)
for j in model$_dm.docvecs[i]$:
f.write(":.17f$\widehat{"}$.format(j))
f.write("")

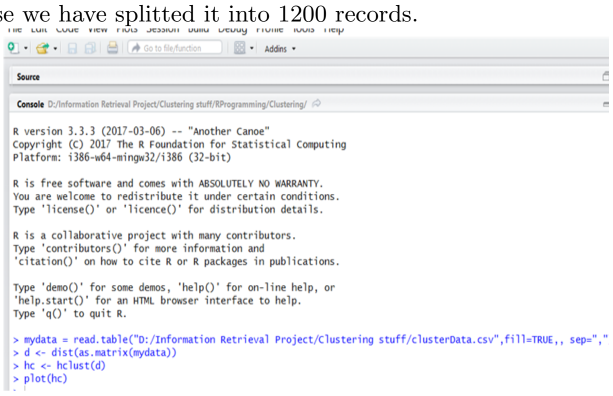


```
594BCD68 vldb 0.25110265612602234 0.82098668813705444 -0.51954734325408936 0.65952157974243164 -1.54000520706176758 1.01890838146209717 1.7932
7EE9B1D6 icse 0.24091458320617676 -0.62212795019149780 -0.16698187589645386 -0.82016998529434204 -1.10024309158325195 0.63905137777328491 -0.0
78B6412F icse 0.10814300179481506 1.82993483543395996 -1.52006399631500244 -0.84898257255554199 0.33337661623954773 2.39901566505432129 -3.377
7DCD25E8 icse 0.2334282100206531 -2.10259795188903809 0.97095972299575806 -0.33773586153984070 1.16876578330993652 -0.27454572916030884 1.887
7C4C6E34 icse 0.14853486418724060 -0.40269440412521362 0.60922938585281372 0.82129758596420288 0.63116830587387085 0.72196334600448608 0.47667
0923DE61 icse -1.03477692604064941 -1.15810263156890869 -0.39442822337150574 1.49058687686920166 -2.72448062896728516 -1.39087486267089844 -2.1
80C5DCE4 icse 0.69066339731216431 0.40608960390090942 1.08820784091949463 0.11254571378231049 -0.03440245985984802 0.69942092895507812 -0.2521
755907FC sigmod 0.80128037929534912 -0.63186889886856079 -0.99009978771209717 -0.78993874788284302 -0.88040751218795776 0.33198320865631104 -1.
58CF6CC2 vldb 1.01716887950897217 1.75086259841918945 0.91454911231994629 -0.95693421363830566 -0.73284417390823364 1.67390549182891846 0.0362
7B4FC3B8 icse -0.58270364999771118 1.76362121105194092 1.09758174419403076 0.31858125329017639 1.02099668979644775 0.60051327943801880 -0.6298
7FD3A4F2 icse -1.22923004627227783 0.46787387132644653 -0.24256332218647003 1.49067163467407227 -2.99599671363830566 0.68953549861907959 0.236
7D8D6B17 icse -0.74037802219390869 -1.29516327381134033 0.03836886584758759 -2.20378422737121582 -1.26241827011108398 0.72184640169143677 0.75
5935B189 vldb -0.44399923086166382 -0.24637629091739655 -0.95257622003555298 0.26865133643150330 -1.39693939685821533 -0.17220477759838104 0.0
7F04578F vldb -0.38696831464767456 -1.13563954830169678 1.08820700645446777 -0.42487689852714539 -0.04269279912114143 1.11965405941009521 -0.4
0275D233 icse 0.47506856918334961 -0.68152141571044922 0.14816676080226898 -0.73944455385208130 0.06448628008365631 -0.25341042876243591 -0.82
7DCCFF4B vldb -1.83379447460174561 -2.38410878181457520 -1.21413385868072510 0.19861555099487305 -0.26509872078895569 1.31304049491882324 0.38
824697EE icse -0.19845901429653168 0.52446311712265015 0.77222126722335815 -0.19438779354095459 -1.18474054336547852 0.12573999166488647 0.257
7B826F83 sigmod -0.37751388549804688 -0.95699334144592285 0.55392295122146606 0.65826672315597534 -0.10066902637481689 2.53624749183654785 0.1
812EDB8E vldb 0.36613467335700989 -1.59546506404876709 1.56607103347778320 0.51130640506744385 0.77103871107101440 1.56794297695159912 1.56924
7C0052A8 icse -2.46531939506530762 -0.96989220380783081 -0.70852291584014893 -0.00991498026996851 -0.39657470583915710 0.58978080749511719 1.7
80EA2D44 vldb -0.52218079566955566 -0.86145836114883423 0.52147459983825684 -0.92063558101654053 0.05266164243221283 0.47307407855987549 1.423
7B08C22C sigmod -0.37781736254692078 -0.22054882347583771 -0.43303942680358887 0.40610447525978088 -1.32496070861816406 0.13638995587825775 0.4
0A5CE596 icse 0.84874296188354492 -2.47151994705200195 -1.26536297798156738 0.69292527437210083 -1.30735790729522705 1.52571952342987061 0.479
03D80C47 sigmod 0.17576704919338226 1.27492487430572510 0.31137001514434814 -0.02827562578022480 -0.18341369926929474 1.66783154010772705 0.59
7CF41357 icse 0.25773313641548157 -1.30607652664184570 -0.01103850360959768 -0.44836640357971191 -0.47272691130638123 1.23048400878906250 0.28
7ED52CA0 sigmod -0.35499596595764160 0.10392460972070694 -0.11773640662431717 -0.65143316984176636 -0.77424705028533936 0.96040266752243042 -0.
7A762FD5 sigmod -0.86546248197555542 -0.33406946063041687 -0.79101592302322388 -0.44221007823944092 -0.93794798851013184 -0.09161593765020370 (
82864317 vldb -0.99468165636062622 -0.74411845207214355 -0.01072742231190205 -0.72622799873352051 -0.05995595082640648 -0.56608939170837402 -1.
760058DE icse 0.03213337436318398 -0.61298924684524536 1.06726717948913574 1.05622410774230957 -0.58892780542373657 0.90044289827346802 -0.274
```

In this way, we will get document in the form of `documentID, documentType and vectorValue` format.Finally, we have converted this text document into csv format with the help of Java application and CSV file is cluster with the help of R programming.

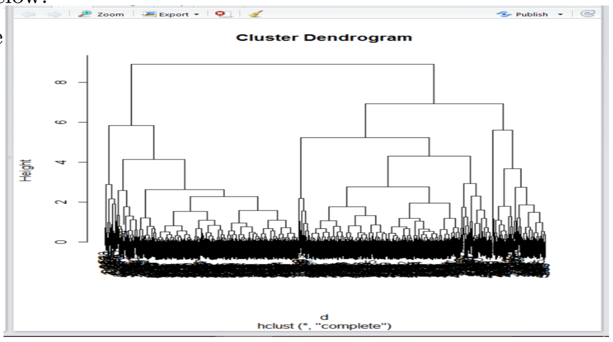
7.2 Clustering Vector value using Chart in R

In the R programming, we have read the CSV[] data from clusterData.csv and we have stored that in myData. Doc2Vec document is very large about 5GB so for ease of use we have splitted it into 1200 records.


Go to file/function
Addins
Source
Console D:/Information Retrieval Project/Clustering stuff/RProgramming/Clustering/

```
R version 3.3.3 (2017-03-06) -- "Another Canoe"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> mydata = read.table("D:/Information Retrieval Project/Clustering stuff/clusterData.csv",fill=TRUE,, sep=","
> d <- dist(as.matrix(mydata))
> hc <- hclust(d)
> plot(hc)
```

Finally we have plotted it in the Dendrogram as show below:



8. WEBSITE LINK

http://137.207.234.72:8080/

9. REFERENCES

1.https://tomcat.apache.org/tomcat-7.0-doc/building.html
2.http://www.oracle.com/technetwork/java/javaee/jsp/index.html
3.https://github.com/deeplearning4j/dl4j-examples/tree/master/dl
examples/src/main/java/org/deeplearning4j/examples/nlp
4.https://deeplearning4j.org/doc2vec
5.https://rare-technologies.com/doc2vec-tutorial/
6.https://radimrehurek.com/gensim/models/doc2vec.html