An Implementation of A Simple Search Engine

Renxia WANG, Yaojia LV, Yujie WU, Zixiang LI
0930300035, 0930300024, 0931100044, 0930300019

December 20, 2012

HKBU-BNU United International College
COMP4210: Information Retrieval and Search Engine
For: Professor Jing ZHAO

Abstract

This report presents an implementation of a search engine that supports ranked retrieval. The search engine accepts key-word based and phrase based queries and return documents ordered by their ranking with respect to the user's information need.

The implementation consist of three modules: tokenzier, indexer and query engine. The tokenizer parses the documents with dealing with stop words, normalization, stemming and lemmatization. The tokens produced by the tokenizer are processed by the indexer. The indexer builds an inverted index with positional information to support regular keyword based queries and phrase queries. Finally, a query engine is implemented to accept a user query and return a list of matching documents in the format of "10 blue links".

1 Introduction

A search engine is designed to search for information on a specific data set. The search results are usually presents in a list of summarized data items in a ranked order. This report describes how to implement a search engine with ranked retrieval feature on a given data set. The implemented search engine supports key-word based and phrase based queries and the search results are returned in the format of "10 blue links" order by their ranking with respect to the user's information need. A title and an summary of each result is presented with highlighted matched words.

The data set used in this experiment is the *Reuters-21578 text categorization collection* which contains 21578 text documents. In this implementation, the data set is adopted as the document corpus and a dictionary and an inverted index are built upon it.

The dictionary is built by the tokenizer. The tokenizer parses the documents and converts them into lists of legitimate terms. Stop words, normalization, stemming and lemmatization routines are invoked in the tokenizer to emit reasonable terms that are to be added into the dictionary. The NLTK (Natural Language Toolkit) package is used in this module and the results are stored in the MongoDB database.

The indexer builds up the inverted index with positional information based on the output of the tokenizer. The inverted index is used to deal with regular keyword based queries. Positional information is used to provide the required search capability of phrase queries. The results are stored in the MongoDB database.

The query engine provides an user interface to accept user's query and return the ranked relevant documents. The query engine accept two kinds of query: 1. regular keyword based queries and 2. phrase queries. A ranking scheme adopted in this implementation is the TF-IDF which is a weighting scheme that provides the ranking capability. The ranked results are presents in the format of "10 blue links". Each result presented has a title and a summary with matched words highlighted. The summary consists of at most three sentences that contain keywords.

1.1 Workload Distribution

In this project, the tokenization module was accomplished by Zixiang LI and the Indexing module was developed by Yaojia LV. Yujie WU and Renxia WANG were in charge of the Query Engine and Ranking modules. After the implementation, this report was finished by Zixiang LI, Yaojia LV, Yujie WU and Renxia WANG.

1.2 Report Structure

The details of the implementation are described in the rest of this report. Section 2 illustrates the procedures for implementing the search engine and the softwares and tools that are invoked in this implementation. The details of those procedures are presented in the following sections. Section 3 describes the format and features of the given data set and the strategies used to parse it are showed in Section 4. Section 5 demonstrates the implementation of the indexer and the details of the query engine are presented in Section 6. After introducing the implementation description, the user interface and usage of the implemented search engine are given in Section 7. At the end of this report, the conclusion is given in Section 8.

2 Methodology

The following procedures are implemented to construct the search engine:

1. Tokenization
    (a) Loads the a article from the data set

   (b) Parses the article which is in the SGML format and extracts the text except the tags
   (c) Normalizes the text of the article by lowercasing all words and replacing the non-alphanumeric characters with spaces
   (d) Deletes the stop words in the text
   (e) Stems the text
   (f) Lemmatizes the text
   (g) Stores the tokenized article into the MongoDB database
   (h) Repeat the above procedures until all articles are processed

2. Indexing
   (a) Loads a article from the MongoDB database
   (b) Maps each token in the article with the article ID and build an inverted index
   (c) Records the positions of each token in this article in its posting list
   (d) Repeat step (a), (b) and (c) until all tokens in articles are processed
   (e) Stores the inverted index and positional index into the MongoDB database

3. Ranking
   (a) Computes the term frequency (TF) of each token in each article
   (b) Computes the document frequency (DF) of each token
   (c) Computes the inverse document frequency (IDF) of each token
   (d) Combines the TF and IDF and constructs a TF-IDF vector for each document
   (e) Computes the TF and IDF of each tokens in a query and constructs a TF-IDF vector for it
   (f) Computes the cosine similarity between the TF-IDF vector of the query and each of the TF-IDF vectors of the results
   (g) Sorts the cosine similarities in a descending order
   (h) Outputs the top ten results

4. Displaying the results
   (a) Summarizes the article by extracting at most the top three sentences that contain the query keywords.
   (b) Highlights the matched words in title and summary of each result

The implementation is developed under the Ubuntu 12.04 OS, with the following softwares and packages:

- Python 2.7
- Natural Language Toolkit 2.0: a Python package used to deal with the stop words, stemming and lemmatization in tokenizer module
- Bob 1.04: a Python package used to compute cosine similarity between high dimensional vectors efficiently
- Dicts 0.3.1: a Python package used to construct sorted dictionaries efficiently
- Pymongo 2.4.1: a Python package used to connect to MongoDB in Python
- MongoDB 2.2.2: A NoSQL database used to store source data, tokens and index

3 Data Set

The Reuters-21578 text categorization collection(distribution 1.0) used in this implementation was released by David D. Lewis on 26 September 1997. This data set is a resource that widely adopted in information retrieval, machine learning and other corpus-based research.

This data set consists of 22 data files which are in Standard Generalized Markup Language (SGML) format which is a ISO standard: ”ISO 8879:1986 Information processing Text and office systems Standard Generalized Markup Language (SGML)”.

Each of the first 21 files contain 1000 documents while the last one contains 578 documents. The following shows one of the documents in this data set:

```
<REUTERS TOPICS="NO" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET"
OLDID="12611" NEWID="428">
<DATE> 2-MAR-1987 09:55:14.04</DATE>
<TOPICS></TOPICS>
<PLACES><D>tanzania</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS><D>imf</D></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>&#5;&#5;&#5;T&#22;&#22;&#1;f0578&#31;reuted f
BC-TANZANIA-SAYS-NO-NEED   03-02 0138</UNKNOWN>
<TEXT>&#2;
<TITLE>TANZANIA SAYS NO NEED FOR NEW ECONOMIC MEASURES</TITLE>
<DATELINE>    DAR ES SALAAM, March 2 - </DATELINE>
<BODY>Tanzania's ruling Chama cha
Mapinduzi (CCM) party has endorsed the government's economic
```

```
reform programme but said it did not think more changes, such
as a further devaluation of the shilling, would be needed.
    Tanzania has devalued the shilling more than 65 pct in less
than a year and has started to overhaul inefficient government
firms in line with a package agreed with the IMF.
    The CCM's national executive committee said it was
satisfied with government efforts to implement IMF conditions.
"Measures taken so far are satisfactory and there is no need to
take other ones -- the devaluation of the shilling included," it
said.
    The committee's statement was in response to a government
report on the IMF package submitted last Thursday.
 Reuter
&#3;</BODY>
</TEXT>
</REUTERS>
```

Each document starts with a tag in the form of

```
<REUTERS TOPICS=?? LEWISSPLIT=?? CGISPLIT=?? OLDID=?? NEWID=??>
```

and ends with a tag of the form:

```
</REUTERS>
```

The attributes, *TOPICS, LEWISSPLIT, CGISPLIT, OLDID, NEWID* have meanings for different purposes. In the implementation, the *NEWID*, which is the identification number the document has in the Reuters-21578 collection(distribution 1.0), is used to identify documents.

The meaning of each internal tags are listed below:

- DATE: the date and time of the document
- TOPICS: the list of TOPICS categories this document belongs to, if any
- PLACES: the list of PLACES categories this document belongs to, if any
- PEOPLE: the list of PEOPLE categories this document belongs to, if any
- ORGS: the list of ORGS categories this document belongs to, if any
- EXCHANGES: the list of EXCHANGES categories this document belongs to, if any

- COMPANIES: the list of EXCHANGES categories this document belongs to, if any
- UNKNOWN: tags bracket control characters and other noisy and/or somewhat mysterious material in the Reuters documents.
- TEXT: the content of the document, including the following elements:

  AUTHOR: author of the document

  DATELINE: originated location and day of the year of the document

  TITLE: title of the document

  BODY: the main text of the document

The content of the *TEXT* is the most important information to be presented to user. In the elements of the *TEXT*, the *BODY* is the most significant element since it covers data that exist in other elements under the *TEXT*. Therefore, in this implementation, the whole document is stored while only the content of the *TEXT* is taken to do tokenization and indexing.

To improve the performance of reading document when processing, all documents are transfer to the MongoDB database. The following text shows a sample data stored in the mongoDB database:

```
{
 "_id": ObjectId("5080e16d848eb017a3cb354c"),
 "body": "\n\nTanzania’s ruling Chama cha\nMapinduzi (CCM)
 party has endorsed the government’s economic\nreform programme
 but said it did not think more changes, such\nas a further
 devaluation of the shilling, would be needed.\n    Tanzania
 has devalued the shilling more than 65 pct in less\nthan a
 year and has started to overhaul inefficient government\nfirms
 in line with a package agreed with the IMF.\n    The CCM’s
 national executive committee said it was\nsatisfied with
 government efforts to implement IMF conditions.\n\"Measures
 taken so far are satisfactory and there is no need to\ntake
 other ones -- the devaluation of the shilling included,\"
 it\nsaid.\n    The committee’s statement was in response to a
 government\nreport on the IMF package submitted last Thursday.
 \n Reuter\n",
 "companies": "",
 "orgs": "tanzania",
 "date": " 2-MAR-1987 09:55:14.04",
 "dateline": "    DAR ES SALAAM, March 2 - ",
 "exchanges": "",
 "meta": {
   "topics": "NO",
    "newid": "428",
    "cgisplit": "TRAINING-SET",
    "lewissplit": "TRAIN",
    "oldid": "12611"
  },
  "orgs": "imf",
  "people": "",
  "places": "tanzania",
  "title": "TANZANIA SAYS NO NEED FOR NEW ECONOMIC MEASURES",
  "topics": "",
  "unknown": " \nT\nf0578reute\nd f BC-TANZANIA-SAYS-NO-
  NEED   03-02 0138",
```

4 Tokenization

While parsing the documents, which words will be the terms in the index should be decided. Not all words in the document are necessary to build the index. Therefore, operations to delete unnecessary words are invoked.

1. Loads body of a document from the database.
2. Lowercases all words.
3. Replaces all non-alphanumeric characters with spaces by regular expression. The number of non-alphanumeric characters, including digits, punctuations, etc, is large while the meaning is low. If these characters are indexed, the size will be large. By this consideration, all the non-alphanumeric are eliminated.
4. Eliminates stop words. Stop words are frequently exist in documents while the importance of them are little. Therefore, they are deleted before constructing the index by using the NLTK's stop words corpus which contains 2400 stop words for 11 languages.
5. Stems the words. By striping off affixes of words, different words with same root can be combine into one word, which reduce the size of the index. The Porter Stemmer defined in the NLTK is used in this procedure.
6. Lemmatizes the tokens. To make sure that the resulting form is a known word in a dictionary, lemmatization process is invoked. The WordNet lemmatizer is used to accomplish this task.

Based on the consideration of the size of index, the implementation do not invoke synonyms during the tokenization process. A word has multiple synonyms. If synonyms of a word are taken to build index, then the size of the index will be increased dramatically.

Soudex is not useful in information retrieval field, thus it is not implemented in the tokenizer module.

Tokens of documents are then stored into the MongoDB database after tokenization. The following is part of a sample result of tokenization:

```
 {
  "_id": "428",
  "tokens": [
    "tanzania",
    "say",
    "need",
    "new",
    "econom",
    "measur",
    "tanzania",
    "rule",
    "chama",
    "cha",
    "mapinduzi",
    ...
  ]
}
```

5 Indexing

5.1 Query Types

This implementation supports two kinds of query:

1. Regular key words query: the result documents contains at least one word in the query.
2. Phrase query: queries are typed inside double quotes and the matching documents contain the words in the query exactly in the specified order.

To support these kinds of query, inverted index and positional index are implemented.

5.2 Inverted Index

The inverted index contains mappings from tokens to the documents that those tokens appear in. Each vocabulary word is a key in the index whose value is its postings list. A token's postings list is the list of documents that the token appears in.

For instance, given three documents:

1. I love you.
2. Hello world.
3. You hate me, don't you?

Then the inverted index for "you" is [1, 3].

5.3 Positional Information

To provide the capability of phrase query which requires the matching documents contain the words in the query exactly in the specified order, an additional information in the posting list is the positions of token occurrences within the document. Without knowing the positions of the tokens in the document, we can only check whether the query tokens simply appear in a document.

In the example listed above, the inverted index with position information for "you" is [[1, [2]], [3, [0, 4]]]

5.4 Implementation

First, the tokens of a document are loaded. Then a inverted index of those tokens are build in the format described above. However, since the index that being built is just for the current document, the postings list of a token would not be a list of lists. It will simply be a list where the first element is the document ID, and the second element is the list of positions the term appears in that document. For example, when the document 3 in the example described above is processed, the posting list would be [3, [0, 4]]. Therefore, the index of the current document needed to be merged with the index for the whole corpus. To do this operation, the postings list of every token in the current document is appended to the postings list of that token in the index for the whole corpus. For instance, the final index for "you" in the example would be [[1, [2]], [3, [0, 4]]].

After processed all corpus, the index are stored into the MongoDB database. There are totally 32125 tokens being indexed. The following is part of a sample result:

```
"emeri": [
  [
    "12195",
   [
     8,
     33,
     38,
     108,
     143,
     155,
     161,
     182,
     193,
     213,
     228,
     313,
     391
   ]
 ],
 [
   "12305",
   [
     0,
     7,
     38,
     61,
     63,
     101,
     104
   ]
 ],
 [
   "17610",
   [
     0,
     6,
     29,
     37,
     46
   ]
 ],
 ...
]
```

6 Query Engine and Ranking

6.1 Procedures

These are the main steps of the query engine. The details of the algorithms are introduced in the following sections.

1. Read the query and tokenize it
2. Compute query's TF-IDF vector
3. Search on inverted index(or positional index, depends on the type of the query), return relevant documents' IDs
4. Use the TF-IDF vectors of documents according to the documents' IDs
5. Compute the similarity of query's TF-IDF vector and the documents' vectors one by one
6. Rank the similarity
7. Get the top 10's corresponding documents' IDs
8. Read the corresponding source documents
9. Summarize each document by extract at most three sentences that contains the query keyword(s)
10. Display the result

6.2 TF-IDF

As one of the most popular weighting schemes in information retrieval field, the TF-IDF scheme assigns each token in a document a weight based on its term frequency (TF) and inverse document frequency (IDF).

Term frequency (TF) indicates the number of occurrences of the token in the document. However, if only pure occurrences counts are applied, the longer documents will be favored more. Thus, a normalization must be invoked to remedy the effect.

$$||D|| = \sqrt{w_1^2 + w_2^2 + w_3^2 + \ldots + w_M^2} \tag{1}$$

$||D||$ is the Euclidean norm which is calculated by taking the square of each unnormalized TF value of tokens in the document, summing them up, and taking the square root of the sum.

$$tf_{t,d} = \frac{N_{t,d}}{||D||} \tag{2}$$

Then divide the number of occurrences of the token in the document $N_{t,d}$ by the $||D||$.

The document frequency (DF) of a token is the number of documents containing the token. The inverse document frequency (IDF) of a token is the number of documents in the corpus divided by the document frequency of that token. That is

$$idf_t = \frac{N}{df_t} = \frac{N}{N} \tag{3}$$

where N is the number of document in the corpus and N_t is the number of documents containing the token t. This value is also needed to be normalized by the following equation:

$$idf_t = \log_{10} \frac{N}{df_t} \tag{4}$$

The TF and IDF of a token now has been calculated, then the TF-IDF value can be computed by

$$tf - idf_{t,d} = tf_{t,d} \cdot idf_t \tag{5}$$

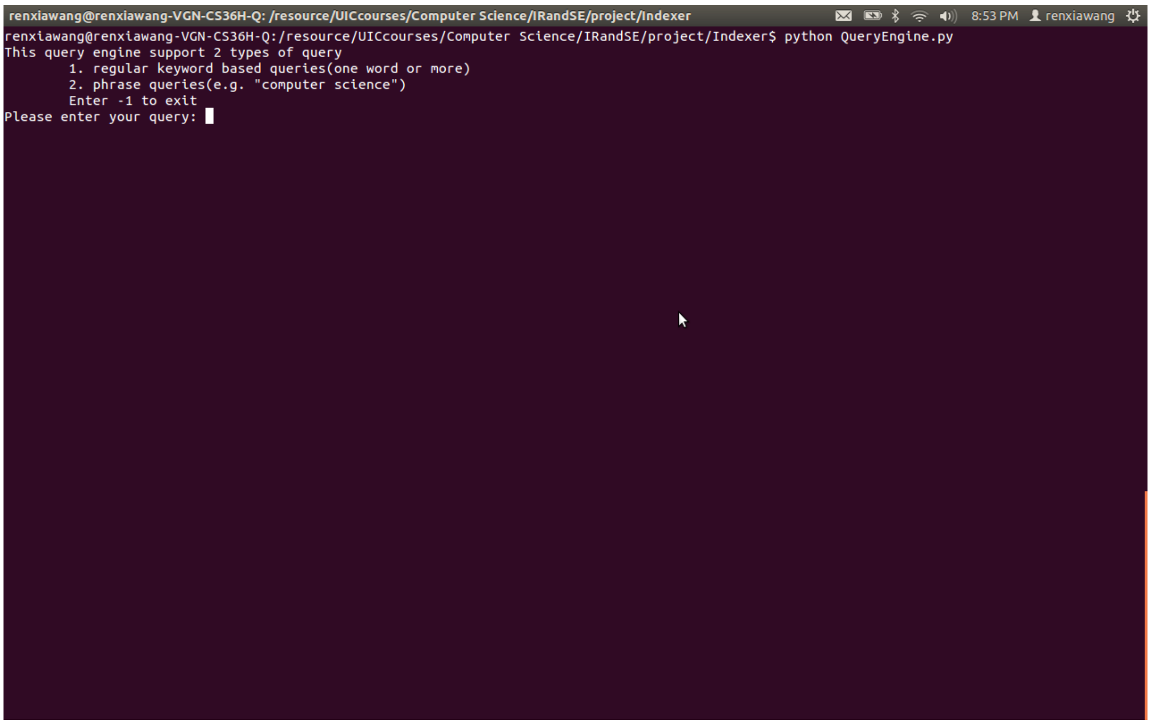
6.3 Vector Space Model

A document consists a set of tokens. Each token in a document has a TF-IDF value. Thus, a document can be represented as a vector which the entries are the values of tokens. A 32125-dimensional vector is constructed to represent each document in the corpus since there are 32125 tokens have been indexed. For tokens that are not existed in a specific document, the TF values of those tokens are 0, therefore the corresponding TF-IDF values are 0. For those exist in the document, their TF-IDF are assigned to the vector. Finally, a sparse vector is constructed to be a representation of a document.

21578 TF-IDF vectors are constructed and stored into the database for further ranking.

6.4 Cosine Similarity

The Rank of a document depends on its similarity to the query. A query is a small document, therefore it can be represented as a vector just like a normal document. The query is received and tokenized and then a corresponding TF-IDF vector is calculated. Then the similarity between the query and the document is computed by applying the cosine similarity algorithm. Finally, by sorting the similarities in a descending order, a ranked result is retrieved.


renxiawang@renxiawang-VGN-CS36H-Q: /resource/UICcourses/Computer Science/IRandSE/project/Indexer
8:53 PM
renxiawang

```
renxiawang@renxiawang-VGN-CS36H-Q:/resource/UICcourses/Computer Science/IRandSE/project/Indexer$ python QueryEngine.py
This query engine support 2 types of query
        1. regular keyword based queries(one word or more)
        2. phrase queries(e.g. "computer science")
        Enter -1 to exit
Please enter your query: 
```

Figure 1: Startup

7 User Interface and Usage

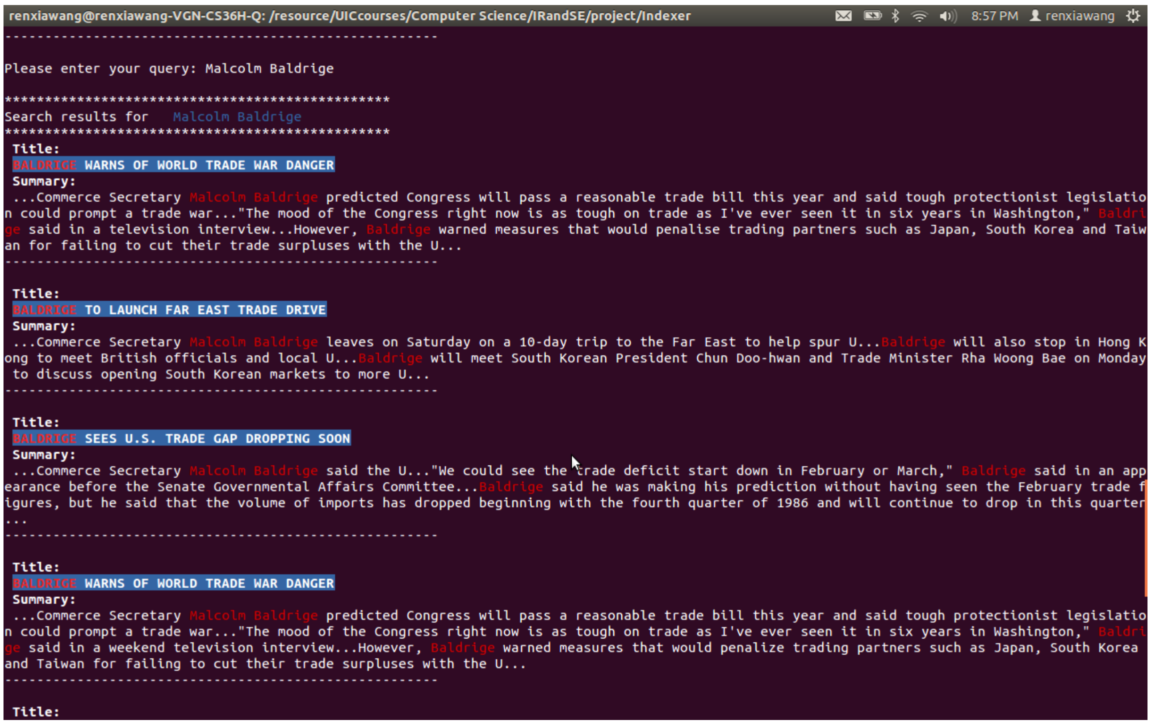
Figure 1 is the startup screen of the search engine. Queries input without quotes are considered as regular keyword queries, otherwise are phrase queries. -1 indicates exit the program.

Figure 2 shows the result screen of a regular keywords query. The results contains at most 10 items. Each resulting items consists of a title and a summary. Matched words in the title and the summary are highlighted. The summary is generated dynamically, which consists of at most three sentences that containing the search keywords.

Figure 3 demonstrates the result screen of a phrase query. The displaying style is the same as the result of regular keywords queries.

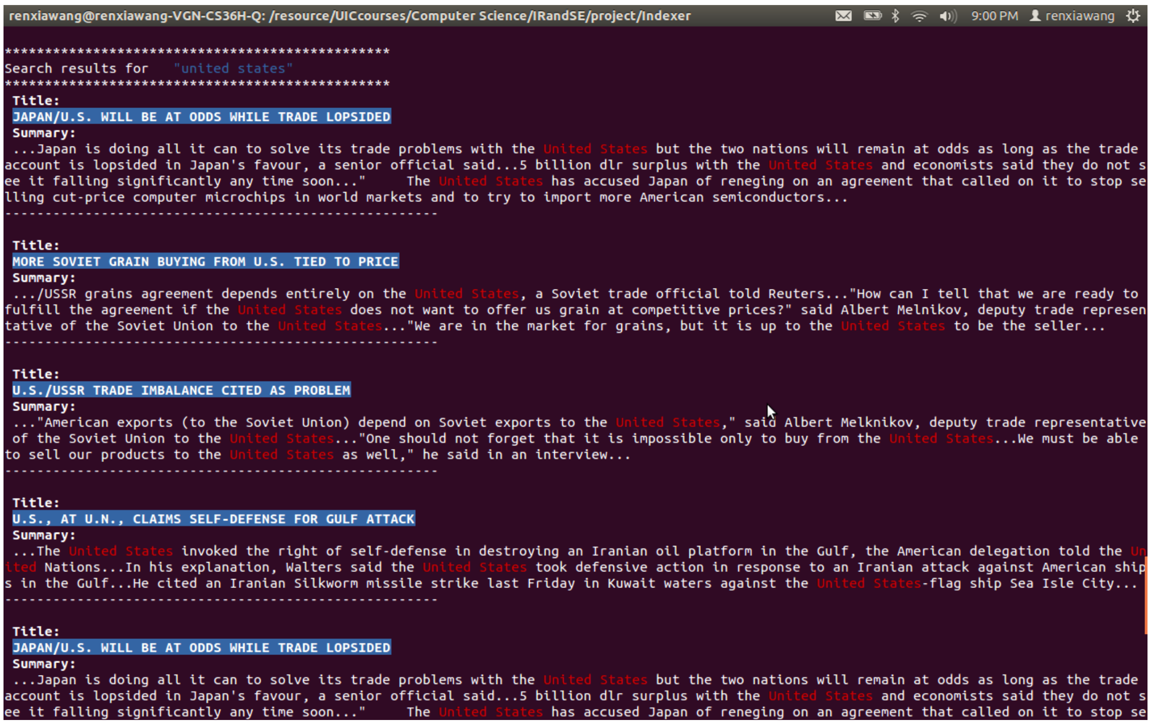
8 Conclusion

The implementation of the search engine provides the basic functions to support keyword-based queries. By tokenization and indexing the text from 21578 documents, 32125 tokens are indexed with positional information and a 32125-dimensional TF-IDF vector is assigned to each document to provide ranking capability. When a query is received, its TF-IDF vector is constructed and it was compared with the relevant documents' vector to gain similarity scores. Based on the similarity scores, a ranked



```
renxiawang@renxiawang-VGN-CS36H-Q: /resource/UICcourses/Computer Science/IRandSE/project/Indexer     8:57 PM  renxiawang
------------------------------------------------------

Please enter your query: Malcolm Baldrige

************************************************
Search results for   Malcolm Baldrige
************************************************
 Title:
 BALDRIGE WARNS OF WORLD TRADE WAR DANGER
 Summary:
 ...Commerce Secretary Malcolm Baldrige predicted Congress will pass a reasonable trade bill this year and said tough protectionist legislatio
n could prompt a trade war..."The mood of the Congress right now is as tough on trade as I've ever seen it in six years in Washington," Baldri
ge said in a television interview...However, Baldrige warned measures that would penalise trading partners such as Japan, South Korea and Taiw
an for failing to cut their trade surpluses with the U...
------------------------------------------------------

 Title:
 BALDRIGE TO LAUNCH FAR EAST TRADE DRIVE
 Summary:
 ...Commerce Secretary Malcolm Baldrige leaves on Saturday on a 10-day trip to the Far East to help spur U...Baldrige will also stop in Hong K
ong to meet British officials and local U...Baldrige will meet South Korean President Chun Doo-hwan and Trade Minister Rha Woong Bae on Monday
 to discuss opening South Korean markets to more U...
------------------------------------------------------

 Title:
 BALDRIGE SEES U.S. TRADE GAP DROPPING SOON
 Summary:
 ...Commerce Secretary Malcolm Baldrige said the U..."We could see the trade deficit start down in February or March," Baldrige said in an app
earance before the Senate Governmental Affairs Committee...Baldrige said he was making his prediction without having seen the February trade f
igures, but he said that the volume of imports has dropped beginning with the fourth quarter of 1986 and will continue to drop in this quarter
...
------------------------------------------------------

 Title:
 BALDRIGE WARNS OF WORLD TRADE WAR DANGER
 Summary:
 ...Commerce Secretary Malcolm Baldrige predicted Congress will pass a reasonable trade bill this year and said tough protectionist legislatio
n could prompt a trade war..."The mood of the Congress right now is as tough on trade as I've ever seen it in six years in Washington," Baldri
ge said in a weekend television interview...However, Baldrige warned measures that would penalize trading partners such as Japan, South Korea
and Taiwan for failing to cut their trade surpluses with the U...
------------------------------------------------------

 Title:
```

Figure 2: Regular Keywords Query and Results



```
renxiawang@renxiawang-VGN-CS36H-Q: /resource/UICcourses/Computer Science/IRandSE/project/Indexer     9:00 PM  renxiawang

************************************************
Search results for   "united states"
************************************************
 Title:
 JAPAN/U.S. WILL BE AT ODDS WHILE TRADE LOPSIDED
 Summary:
 ...Japan is doing all it can to solve its trade problems with the United States but the two nations will remain at odds as long as the trade
account is lopsided in Japan's favour, a senior official said...5 billion dlr surplus with the United States and economists said they do not s
ee it falling significantly any time soon..."      The United States has accused Japan of reneging on an agreement that called on it to stop se
lling cut-price computer microchips in world markets and to try to import more American semiconductors...
------------------------------------------------------

 Title:
 MORE SOVIET GRAIN BUYING FROM U.S. TIED TO PRICE
 Summary:
 .../USSR grains agreement depends entirely on the United States, a Soviet trade official told Reuters..."How can I tell that we are ready to
fulfill the agreement if the United States does not want to offer us grain at competitive prices?" said Albert Melnikov, deputy trade represen
tative of the Soviet Union to the United States..."We are in the market for grains, but it is up to the United States to be the seller...
------------------------------------------------------

 Title:
 U.S./USSR TRADE IMBALANCE CITED AS PROBLEM
 Summary:
 ..."American exports (to the Soviet Union) depend on Soviet exports to the United States," said Albert Melknikov, deputy trade representative
 of the Soviet Union to the United States..."One should not forget that it is impossible only to buy from the United States...We must be able
to sell our products to the United States as well," he said in an interview...
------------------------------------------------------

 Title:
 U.S., AT U.N., CLAIMS SELF-DEFENSE FOR GULF ATTACK
 Summary:
 ...The United States invoked the right of self-defense in destroying an Iranian oil platform in the Gulf, the American delegation told the Un
ited Nations...In his explanation, Walters said the United States took defensive action in response to an Iranian attack against American ship
s in the Gulf...He cited an Iranian Silkworm missile strike last Friday in Kuwait waters against the United States-flag ship Sea Isle City...
------------------------------------------------------

 Title:
 JAPAN/U.S. WILL BE AT ODDS WHILE TRADE LOPSIDED
 Summary:
 ...Japan is doing all it can to solve its trade problems with the United States but the two nations will remain at odds as long as the trade
account is lopsided in Japan's favour, a senior official said...5 billion dlr surplus with the United States and economists said they do not s
ee it falling significantly any time soon..."      The United States has accused Japan of reneging on an agreement that called on it to stop se
```

Figure 3: Phrase query

result is retrieved. The result is in the format of "10 blue links" with matched words highlighted. For each document, a summary is generated by concatenating at most three sentences that contain the keywords.