

ADVANCED INFORMATION STORAGE AND RETRIEVAL (IFN647)

ASSESSMENT 2 REPORT

A ASSESSMENT SUBMITTED TO  
THE SCIENCE AND ENGINEERING FACULTY  
OF QUEENSLAND UNIVERSITY OF TECHNOLOGY



GUO, Jinning (n9858598) (Submitter)

ZHAO, Jin (n9800174)

JIANG, Yuchen (n9573950)

LIU, Hanchuan (n8554544)

School of Electrical Engineering and Computer Science

Science and Engineering Faculty

Queensland University of Technology

Due Date: Tuesday, 29<sup>th</sup> of October 2019

## Table of Contents

1 Statement of Completeness and Work Distribution .....	2
2 Design .....	3
2.1 Overall Design.....	3
2.2 Indexing Strategy .....	5
2.2.1 Classes and methods.....	5
2.2.2 Handling the structure of the source document .....	5
2.3 Search Strategy.....	5
2.3.1 Classes and methods.....	5
2.3.2 User Interface.....	6
3. Changes to Baseline.....	12
4. System Evaluation .....	14
4.1 Efficiency Metrics.....	14
4.1.1 Indexing Size and Time.....	14
4.1.2 Searching Time.....	15
4.2 Effectiveness Metrics .....	17
4.2.1 Query 001 (Retrieved all relevant passages) .....	17
4.2.2 Query 002 (Retrieved all relevant passages) .....	18
4.2.3 Query 001 (Retrieved Top 20 Relevant Passages) .....	19
4.2.4 Query 002 (Retrieved Top 20 Relevant Passages) .....	20
5. Comparison with Baseline.....	22
5.1 Efficiency Metrics.....	22
5.2 Effectiveness Metrics .....	23
5.2.1 Retrieved all relevant passages .....	23
5.3 Overall Interpolated Recall-Precision Curve.....	24
6. User Guide.....	26
6.1 Instruction .....	27
6.1.1 Main interface.....	27
6.1.2 Detail information .....	30
6.1.3 Log box .....	31
7. Advanced Features .....	32
7.1 POS tagging.....	32
7.2 WordNet.....	32

## 1 Statement of Completeness and Work Distribution

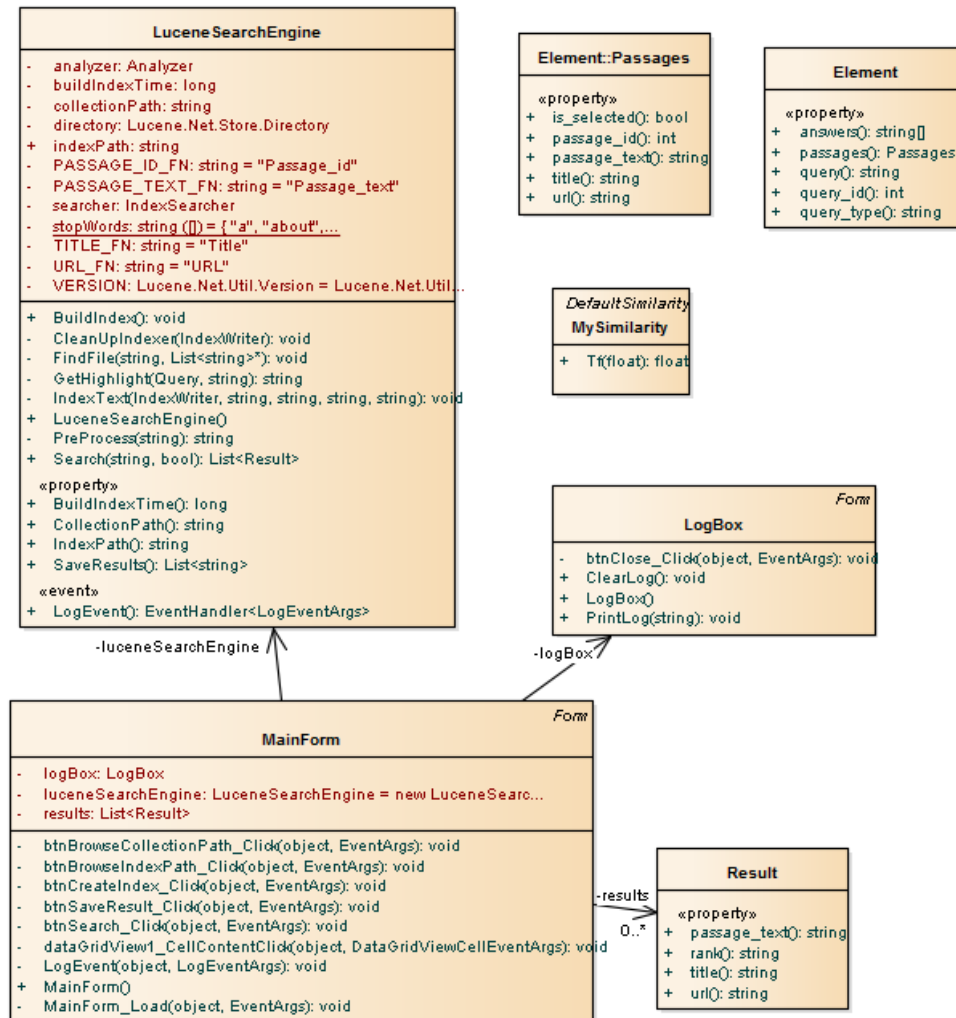
In this project, it divided into two parts, which including program and report. In the first part, the function of the program including the baseline system, development of evaluation framework, two more changes to baseline system, changes to Lucene Scoring and improve the functionality of query processing. In this report, we illustrate the overall design of the program, changes to baseline and query processing. We also evaluate the program performance in efficiency and effectiveness, and compare the performance of IRSearcher with the baseline system. In the last section, we also explain how to use the program and how to improve the program with the advanced features.

The following table shows the work distribution and how marks should be split between team members.

	Jin Zhao	Yuchen Jiang	Jinning Guo	Hanchuan Liu
Baseline System	25%	25%	25%	25%
Index Strategy	25%	25%	25%	25%
Search Strategy	25%	25%	25%	25%
Query processing	25%	25%	25%	25%
Improve Lucene Scoring	25%	25%	25%	25%
Changes to baseline system		50%	50%	
GUI Design	50%			50%
Program Coding	25%	25%	25%	25%
Bug Test	50%		50%	
Debug	20%	30%	30%	20%
Evaluation	10%	70%	10%	10%
Comparison to Baseline	40%	10%	20%	30%
Advanced Features	15%	15%	30%	40%

## 2 Design

### 2.1 Overall Design



The created Index is stored in a folder in a user-defined path. The system can show the time cost when index was created. The search engine searches and displays results from the source file by using index. The system provides two different search results which is with and without pre-processing, and the user can choose to use or not use pre-processing. After the search, the user can see a list of related documents containing rank, title and URL in the GUI interface. The user can click on each relevant document to see the content of the document. Keywords in the document are highlighted in red. In

addition, the system can show how long the search took. Furthermore, the user can choose to save the search results and customize the id to save the document.

Inputs:

- Source document path
- Index path
- Query
- Whether to pre-processing
- The ID of the saved result

Outputs:

- Time cost of Index building
- Time taken by Search
- Query
- Results of search
- Warning information
- Specific content of relevant documents
- The saved results after search

This program will run correctly at the establishment of an index and search success, and in the pop-up window display and search result list. But, when they encounter the error popup LogBox window, highlighting code to the wrong place at the same time, to prompt the user the location of the error, and prompt the user to operate correctly.

- Cannot find .json source file
- Search without built index
- Search without .json source file

## 2.2 Indexing Strategy

### 2.2.1 Classes and methods

There are methods which are used to create indexing:

- btnCreateIndex\_Click
- BuildIndex: Build index files
- FindFile: Find .json file
- IndexText: Create a Lucene and add to Index
- CleanUpIndexer: Close index
- IndexWriter

The user first browse the source file and index, and then click the create index button. The methods listed above are then called in turn. Extracted from URL, the title in Index will be extracted from URL.

### 2.2.2 Handling the structure of the source document

First traversal the file with the .json extension in the specified directory and read the .json string in the file. Then, by JsonConvert static methods DeserializeObject converts .json string to Element object list. Then traverse the list of objects, extracting passage\_text, passage\_id. Through processing the URL to namely to extract the URL in the title of the last used for indexing. UrlFiled is not used for search, so it is stored but now token. TitleField stores and token. PassageField storage and token. PassageidField don't used to search, so storage but not token.

## 2.3 Search Strategy

### 2.3.1 Classes and methods

There are methods which are using in search:

- btnSearch\_Click
- Search: Create searcher
- Pre-process: Remove Stop words, Space and Punctuation

- Parser: MultifieldParser (Title and Passage Fields)
- Boost: Title and Passage Fields
- results

After the user enters the keyword they want to search for and clicks the search button, the program first checks to prevent no words from being entered. In addition, if the user wants to do pre-processing, they need to check the pre-processing option. Otherwise, the system will default to no pre-processing and use simple search. In addition, the result is divided into two parts: one is the result displayed in the interface, and the other is the result saved in the file.

When the user wants to do pre-processing, they need to check the pre-processing option before clicking the search button. When pre-processing, the first step is word token. Then remove the stop word.

### 2.3.2 User Interface

- Build Index: first, the user needs to determine the Index path by selecting the storage path of the source files and indexing files. As shown below, select the source file in the Upload box above, select Index in the Upload box below, and then click Build Index to create Index.

The screenshot shows a web browser window with the address bar displaying `http://EduQuizQuestionAnsweringSystem.qut.net.au`. The page title is "EduQuiz Question Answering System". The interface contains two identical rows of input fields. Each row has a "File upload" text box and a "Choose Source..." button. Below these rows is a "Build Index" button.

EduQuiz Question Answering System

File upload Choose Source...

File upload Choose Index...

Build Index

- Index building time: after building the index, user can see that how long it take to building index.





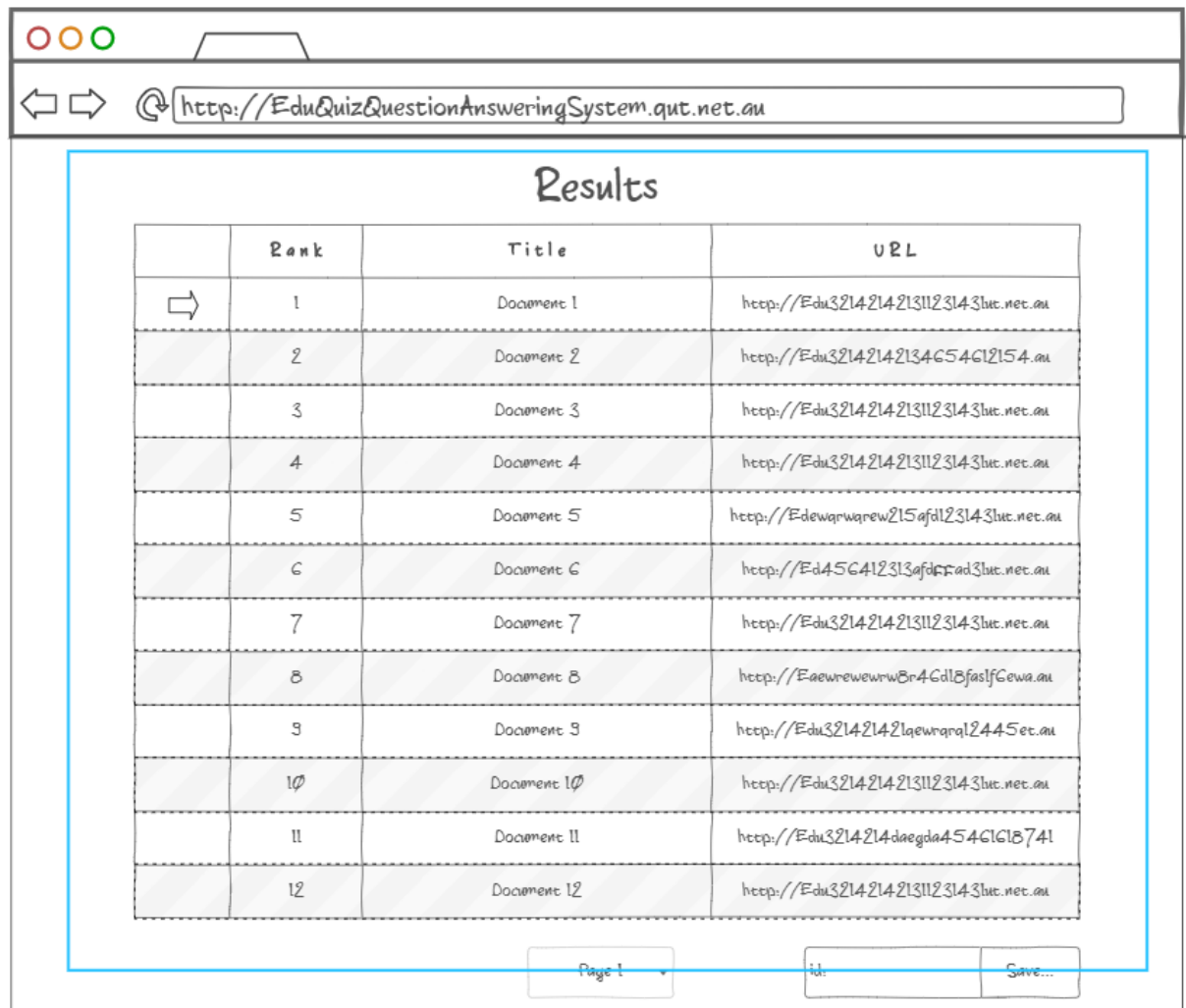
- The user enters the natural language information what they want to search in the text box, and then clicks the search button to start the search. In addition, users can choose pre-processing or not. If they want to do pre-processing, they can do this by checking the pre-processing in the lower right corner.



- Searching time: User can see that how long it search taken.



- After submitting the query, a list of related documents is displayed in the result page. The mock-up picture shows the details of results below, which includes Rank, Title, and URL. At the same time, these documents are ranked according to the score. Users can view more relevant documents through the drop-down list. In addition, the user can click the link in the URL to see the specific content of the relevant document. In addition, the user can choose to download the results to a custom path and can customize the saved ID.



- Display Document: After clicking the URL, the user can see the specific content of the relevant document. At the same time, keywords are highlighted in red so that the user can find them.



### 3. Changes to Baseline

#### List of changes

- Change Single-threaded index to Multi-threaded index

For the baseline system, we use the single-threaded to build index, but we found it takes long time and large storage. Therefore, we tried to use the approach of multi-threaded index to reduce the time and storage. The comparison will be presented in the following sections (System Evaluation & Comparison to Baseline).

- Improve the list of stop word

For this change, we try to improve the list of stop words, which means remove the unnecessary words in the stop word list, so that we can make the index and query more meaningful after pre-processing.

- Change Simple Analyzer to Standard Analyzer

The simple analyser is used to splits text at non-letters and converts text to lower case. However, in this project, we need to pre-process the passage the query to make the index and search query more meaningful, so that we choose to use the standard analyser which contain the function of simple analyser and removing the stop words.

- Only analyse the title and passage field

For this collection file, it includes passage\_id, URL, and passage\_text. Because the passage\_id and URL are not necessary for the search, so that we only analyse the potential fields which including the title (extract from URL) and passage. It can reduce the time of indexing.

- Implementation of the multi fields parser and boost the scoring

The multi fields parser is implemented to allow the search in both title and passage, which means the system can be allowed to match the result in title and passage that is relevant to the search query. The multi field search aims to find the most relevant answer for users. In addition, we also boost the

scoring for title and passage, which means if the keywords from query appear in the both passage and title, that document is more relevance with the information need.

- Change the default similarity returned value

In baseline system, we choose Tf as the default similarity. The default similarity can score the search result with the term frequency, the document has the higher term frequency will get the higher score and higher relevance. However, the higher term frequency does not represent the document is the correct answer that user need. So that we change the returned value from `“(float)Math.Sqrt(freq)”` to 1, which means the term frequency in the specific document will not influence the relevance.

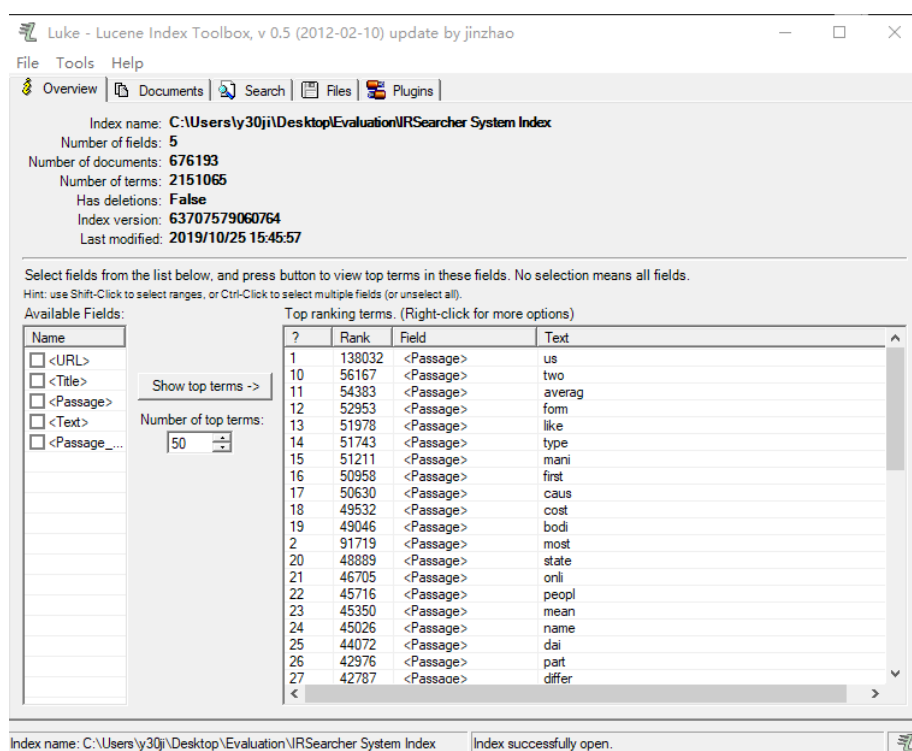
## 4. System Evaluation

In this section, we will use the set of efficiency and effectiveness metrics to evaluate the system. In addition, we will also compare with the baseline system through the efficiency and effectiveness metrics.

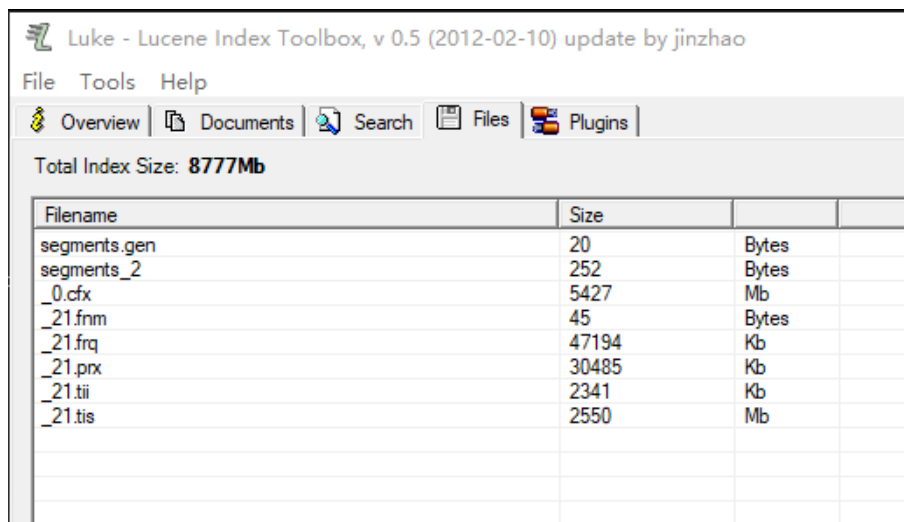
### 4.1 Efficiency Metrics

#### 4.1.1 Indexing Size and Time

In the efficiency evaluation, there are 3 metrics need to be evaluated, such as index size, index time and search time. We will use the Luke.Net to confirm the index building and index size. The following figure represent the index is built successfully.



The index size (8777Mb) for the system is shown in figure ...



Luke - Lucene Index Toolbox, v 0.5 (2012-02-10) update by jinzhao

File Tools Help

Overview Documents Search Files Plugins

Total Index Size: **8777Mb**

Filename	Size		
segments.gen	20	Bytes	
segments_2	252	Bytes	
_0.cfx	5427	Mb	
_21.fnm	45	Bytes	
_21.frq	47194	Kb	
_21.prx	30485	Kb	
_21.tii	2341	Kb	
_21.tis	2550	Mb	

In order to get the most representative indexing time, we build index 6 times. As shown in the following table, the index time is not significant increase or decrease, so that we choose to use the average indexing time to compare with the baseline system.

Times of Indexing	Index time	Index size
1st	82.516s	8777Mb
2nd	83.193s	8777Mb
3rd	86.238s	8777Mb
4th	84.861s	
5th	82.892s	
6th	82.706s	
Average	83.734s	8777Mb

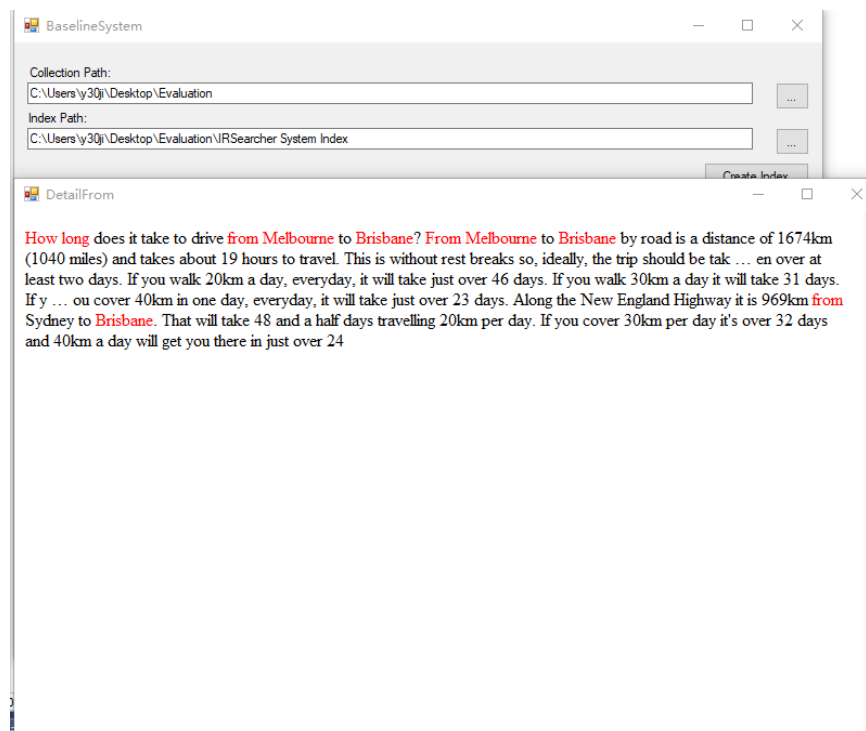
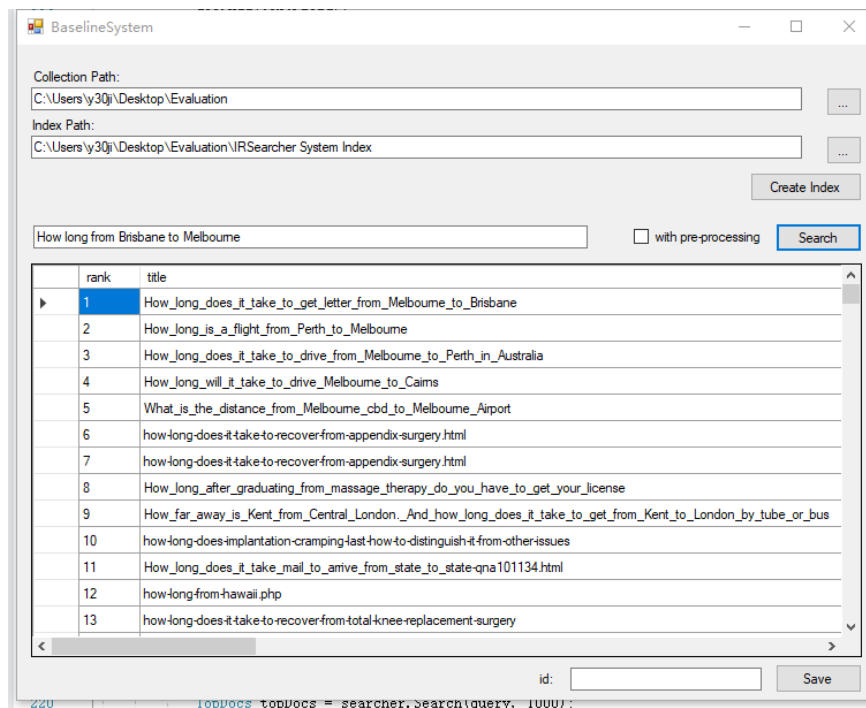
#### 4.1.2 Searching Time

In order to get the relatively stable search time, we test 5 queries 3 times with the same setting. The following table shows the search time of the testing.

Query	1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>
001: How long from Brisbane to Melbourne	0.007s	0.009s	0.008s
002: What is RBA	0.009s	0.008s	0.003s
003: How to use python in Artificial Intelligence	0.004s	0.003s	0.003s
004: Australia government	0.006s	0.003s	0.002s
005: The location of Perth	0.023s	0.018s	0.009s
All	0.049s	0.041s	0.025s



In the case of query 001, the following figures represent that the program works and provides the user with the retrieved results.



Through the result of the testing, we can conclude that the search time of each query is decline slightly after 3 times of search. Therefore, we decide to choose the average search time of the 5 queries as the variable to compare with the search time in baseline system. The average search time in IRSearcher is 0.038s. In baseline system, we will search the same query that we test in the IRSearcher. In order to control the consistency of the variables, both query setting and times of testing are the same.

## 4.2 Effectiveness Metrics

We use four metrics to test the effectiveness of the IRSearcher, which including Precision @10, Mean Reciprocal Rank, MAP and Interpolated Recall. In this section, we use two different strategies include retrieved all relevant passages and retrieved Top20 passages to evaluate the effectiveness.

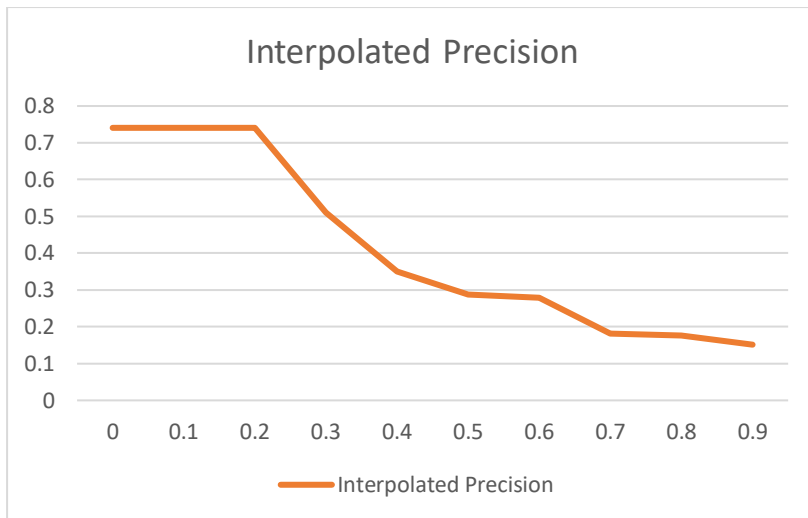
In the “Retrieved all relevant passages” the qrel file contains 58 relevant passages in query 001 and 43 relevant passages in query 002. In the “Retrieved Top 20 selected passages”, the qrel file contains 4 relevant passages in query 001 and 5 relevant passages in query 002.

### 4.2.1 Query 001 (Retrieved all relevant passages)

Topic	MAP	Mean Reciprocal Rank	P@10
001	0.4161	1	0.6

- Interpolated Recall/Precision Curve

Recall	Interpolated Precision
0	1
0.1	0.7407
0.2	0.7407
0.3	0.7407
0.4	0.5106
0.5	0.3505
0.6	0.2869
0.7	0.2789
0.8	0.1815
0.9	0.1755
1	0.1514

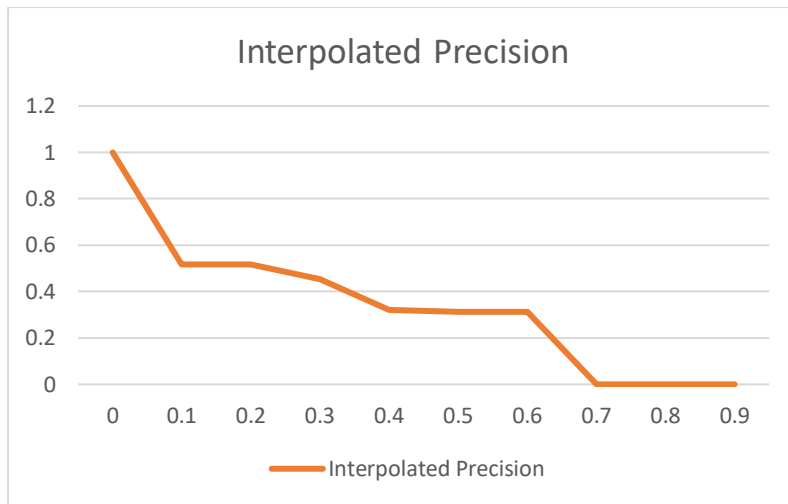


#### 4.2.2 Query 002 (Retrieved all relevant passages)

Topic	MAP	Mean Reciprocal Rank	P@10
002	0.374	1	0.7

- Interpolated Recall/Precision Curve

Recall	Interpolated Precision
0	1
0.1	1
0.2	0.5172
0.3	0.5172
0.4	0.4545
0.5	0.3205
0.6	0.3131
0.7	0.3131
0.8	0
0.9	0
1	0

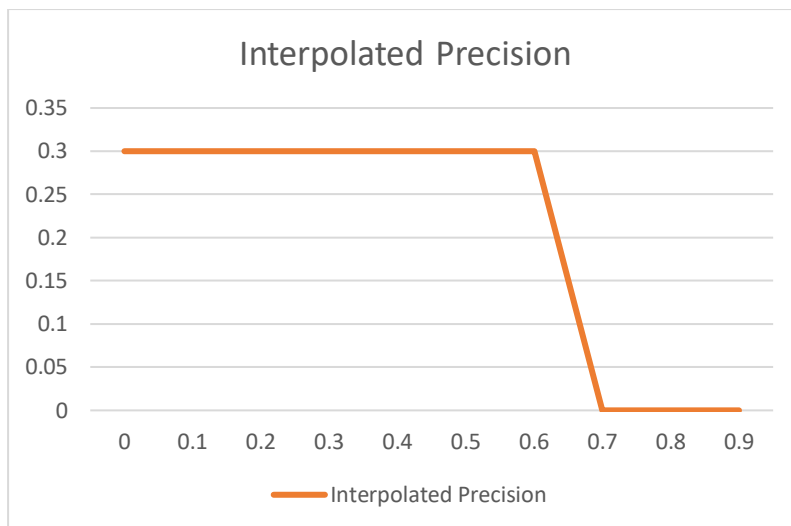


#### 4.2.3 Query 001 (Retrieved Top 20 Relevant Passages)

Topic	MAP	Mean Reciprocal Rank	P@10
001	0.2	1	0.3

- Interpolated Recall/Precision Curve

Recall	Interpolated Precision
0	0.3
0.1	0.3
0.2	0.3
0.3	0.3
0.4	0.3
0.5	0.3
0.6	0.3
0.7	0.3
0.8	0
0.9	0
1	0

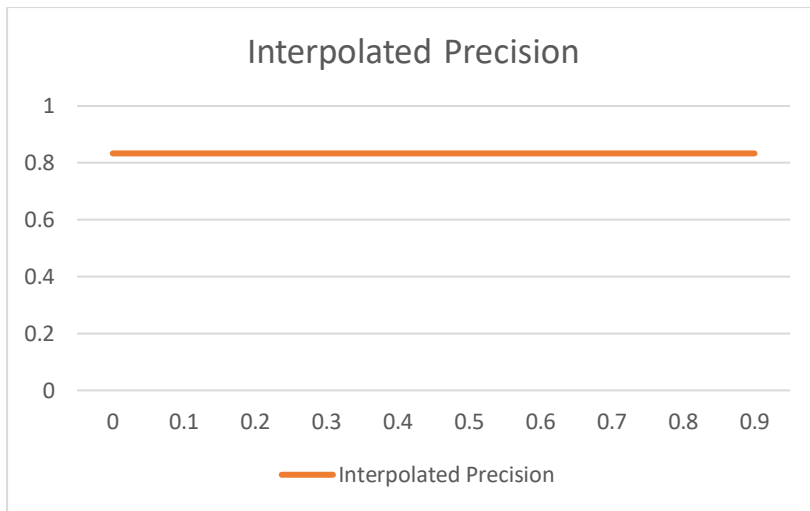


#### 4.2.4 Query 002 (Retrieved Top 20 Relevant Passages)

Topic	MAP	Mean Reciprocal Rank	P@10
002	0.71	0.5	0.5

- Interpolated Recall/Precision Curve

Recall	Interpolated Precision
0	0.8333
0.1	0.8333
0.2	0.8333
0.3	0.8333
0.4	0.8333
0.5	0.8333
0.6	0.8333
0.7	0.8333
0.8	0.8333
0.9	0.8333
1	0.8333



## 5. Comparison with Baseline

In the previous section, we have seen the performance of the IRSearcher through the evaluation of efficiency and effectiveness. In this section, we will use the same strategy to evaluate the Baseline system and compare the performance against IRSearcher.

### 5.1 Efficiency Metrics

The following table will show you the difference of the index size, index time and search time between the baseline system and IRSearcher. For the index time, we also build index 6 times and use the average index time as the variable. For search time, we still search 3 times of each query (from query 001 to 005) and calculate the average time.

Metric	Baseline System	IRSearcher
Index Size	13171Mb	8777Mb
Index Time	139.696s	83.734s
Search Time	0.063s	0.038s

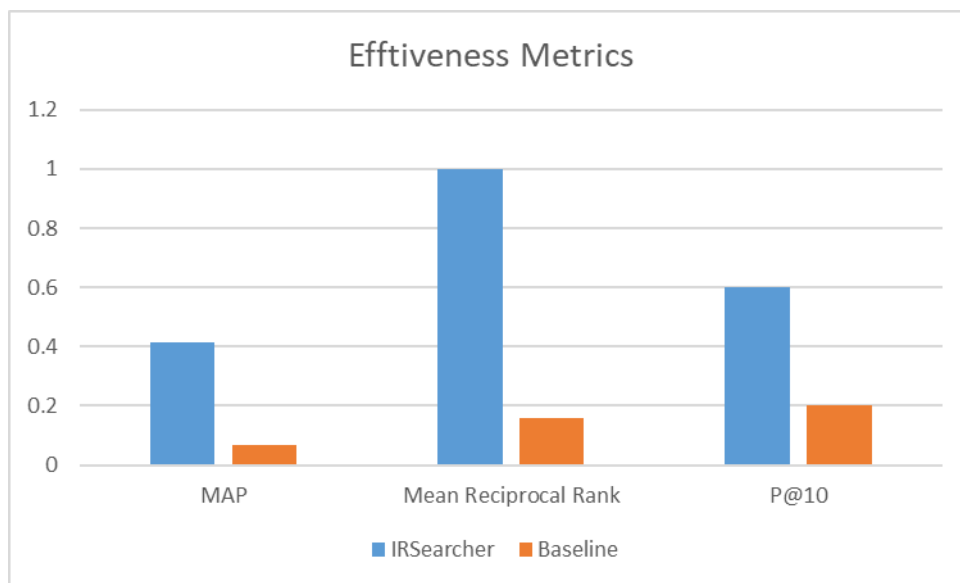
As shown in the table, the index size and index time is decline dramatically, which means the implementation of multi-threaded indexing is one of the changes that can improve the efficiency of baseline system. The search time is decline slightly, which means the changes of search strategy is also improve the efficiency of baseline system.

## 5.2 Effectiveness Metrics

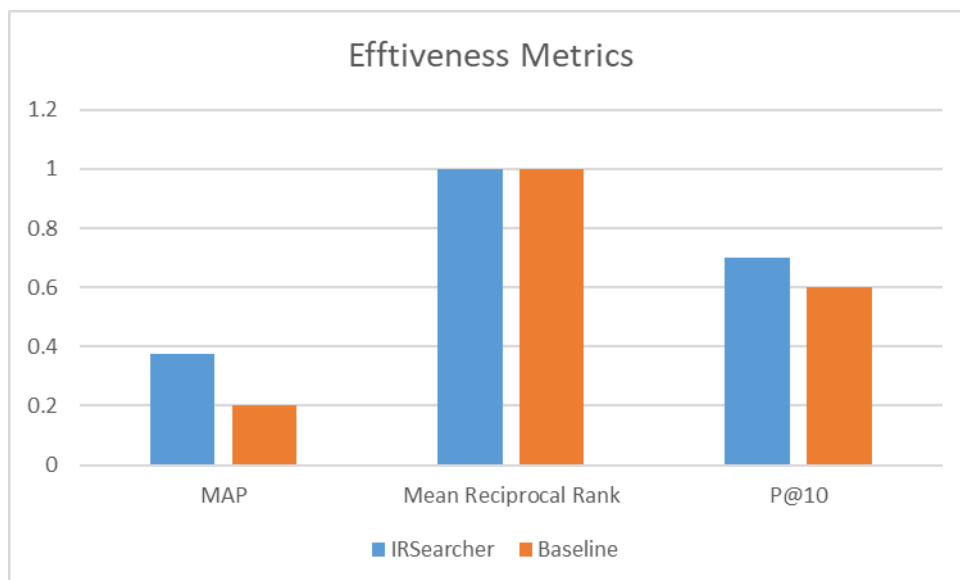
### 5.2.1 Retrieved all relevant passages

Topic	MAP		Mean Reciprocal Rank		P@10	
	IRSearcher	Baseline	IRSearcher	Baseline	IRSearcher	Baseline
001	0.4161	0.069	1	0.16	0.6	0.2
002	0.374	0.2	1	1	0.7	0.6

- Query 001



- Query 002



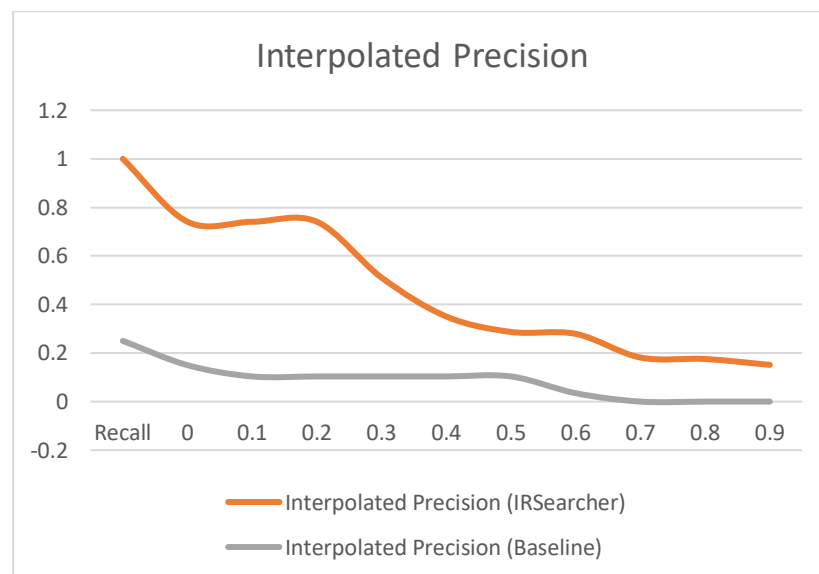


Through the analysis of 3 metrics of effectiveness, the performance of the IRSearcher in query 001 and query 002, which show in the table and figure is better than baseline system, which means the search strategy and scoring boost improve the effectiveness of the baseline system.

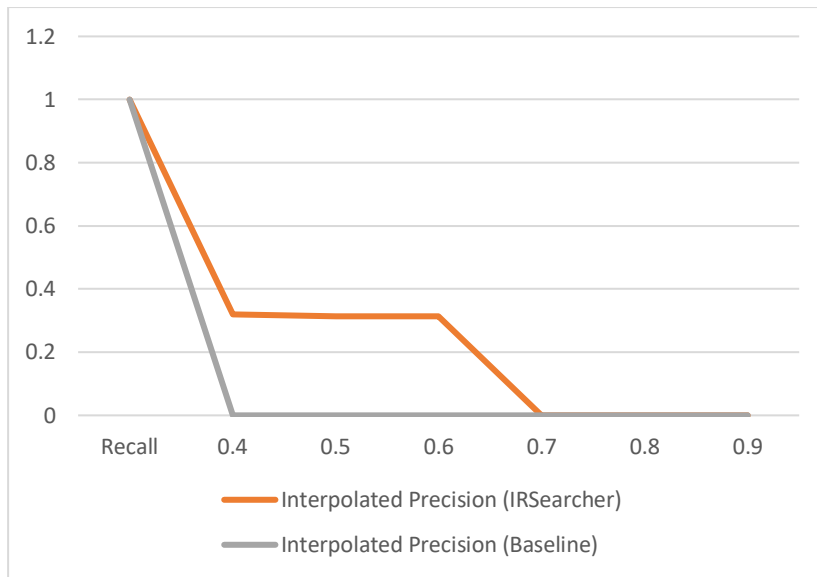
### 5.3 Overall Interpolated Recall-Precision Curve

The following two figures show us the precision-recall curves. It shows how the recall and precision value changes in the IRSearcher and Baseline system. For the same query, we can see the recall-precision curve in IRSearcher is stay higher than baseline system, which means the IRSearcher system performed better than the baseline system.

- Precision Curve (Query 001)



- Precision Curve (Query 002)



## 6. User Guide

In this chapter, a detailed instruction will be explained with the graph in a sequence of routines. The chapter is divided into 3 sections regarding the interface windows, including the main console, detail form, and the log box.

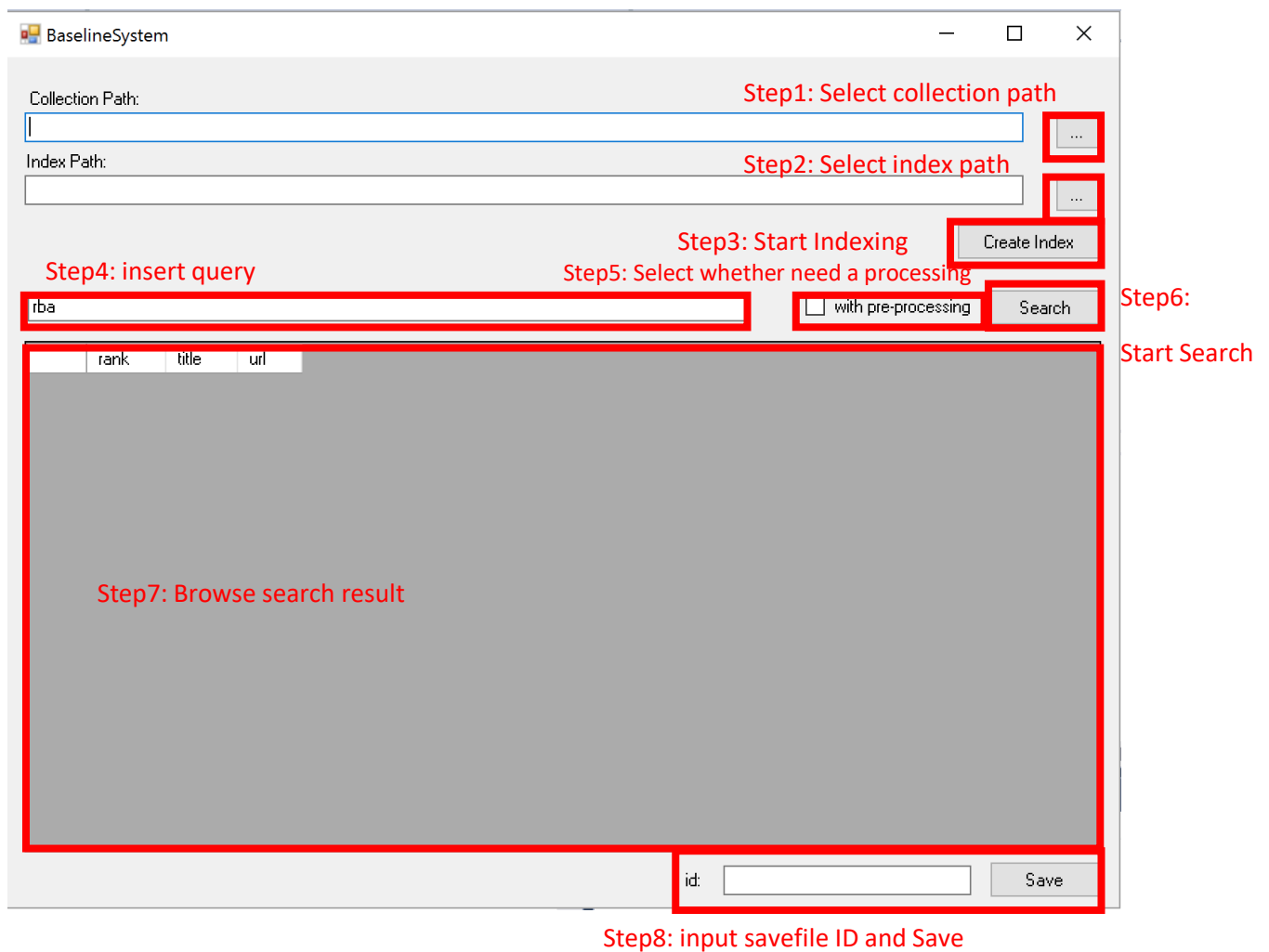
The utility will be instructed including each of below:

- Writing index
- Sending query
- Processing query (optional)
- Matching query and passages
- Present the valid result by rank
- Safe result

## 6.1 Instruction

### 6.1.1 Main interface

There 3 parts of the main interface, the pre-set, search and save the result.



#### Pre-set

This is for the indexing. On the top half of the main form, there are 2 path fields and selection buttons for a user to input the path where collection.js is stored and assign the index file location.

The index command will start by click on the create index and a log box will come up.

## Search

The text input field under the path select section is for a user to the input query. There is an optional choice box for whether it requires query processing.

After a query is typed into the text box, you can simply start searching by clicking the search button.

A detailed window will show up with the selected object.

The dark grey field is for presenting the search result list. It includes 3 fields: the rank of relevance, passage title, and URL.

rank	title
1	what-is-rba
2	what-is-rba
3	what-is-results-based-accountability-rba
4	rebuildable-atomizer-rba
5	rebuildable-atomizer-rba
6	risk-based-authentication-RBA
7	art-micro-coil-beginners-guide-coiling-rba
8	rebuildable-atomizer-rba
9	RBA
10	wholesale_kangertech_subtank_mini_clearomizer_and_rba
11	rba-how-to
12	rebuildable-atomizer-rba
13	rba-how-to

url
<a href="http://rba-africa.com/about/what-is-rba/">http://rba-africa.com/about/what-is-rba/</a>
<a href="http://rba-africa.com/about/what-is-rba/">http://rba-africa.com/about/what-is-rba/</a>
<a href="http://resultsleadership.org/what-is-results-based-accountability-rba/">http://resultsleadership.org/what-is-results-based-accountability-rba/</a>
<a href="https://www.slimvapepen.com/rebuildable-atomizer-rba/">https://www.slimvapepen.com/rebuildable-atomizer-rba/</a>
<a href="https://www.slimvapepen.com/rebuildable-atomizer-rba/">https://www.slimvapepen.com/rebuildable-atomizer-rba/</a>
<a href="http://searchsecurity.techtarget.com/definition/risk-based-authentication-RBA">http://searchsecurity.techtarget.com/definition/risk-based-authentication-RBA</a>
<a href="http://www.vapecore.com/2014/01/art-micro-coil-beginners-guide-coiling-rba/">http://www.vapecore.com/2014/01/art-micro-coil-beginners-guide-coiling-rba/</a>
<a href="https://www.slimvapepen.com/rebuildable-atomizer-rba/">https://www.slimvapepen.com/rebuildable-atomizer-rba/</a>
<a href="http://acronyms.thefreedictionary.com/RBA">http://acronyms.thefreedictionary.com/RBA</a>
<a href="http://www.hawkvape.com/wholesale_kangertech_subtank_mini_clearomizer_and_rba">http://www.hawkvape.com/wholesale_kangertech_subtank_mini_clearomizer_and_rba</a>
<a href="http://vapercoils.com/tag/rba-how-to">http://vapercoils.com/tag/rba-how-to</a>
<a href="https://www.slimvapepen.com/rebuildable-atomizer-rba/">https://www.slimvapepen.com/rebuildable-atomizer-rba/</a>
<a href="http://vapercoils.com/tag/rba-how-to">http://vapercoils.com/tag/rba-how-to</a>

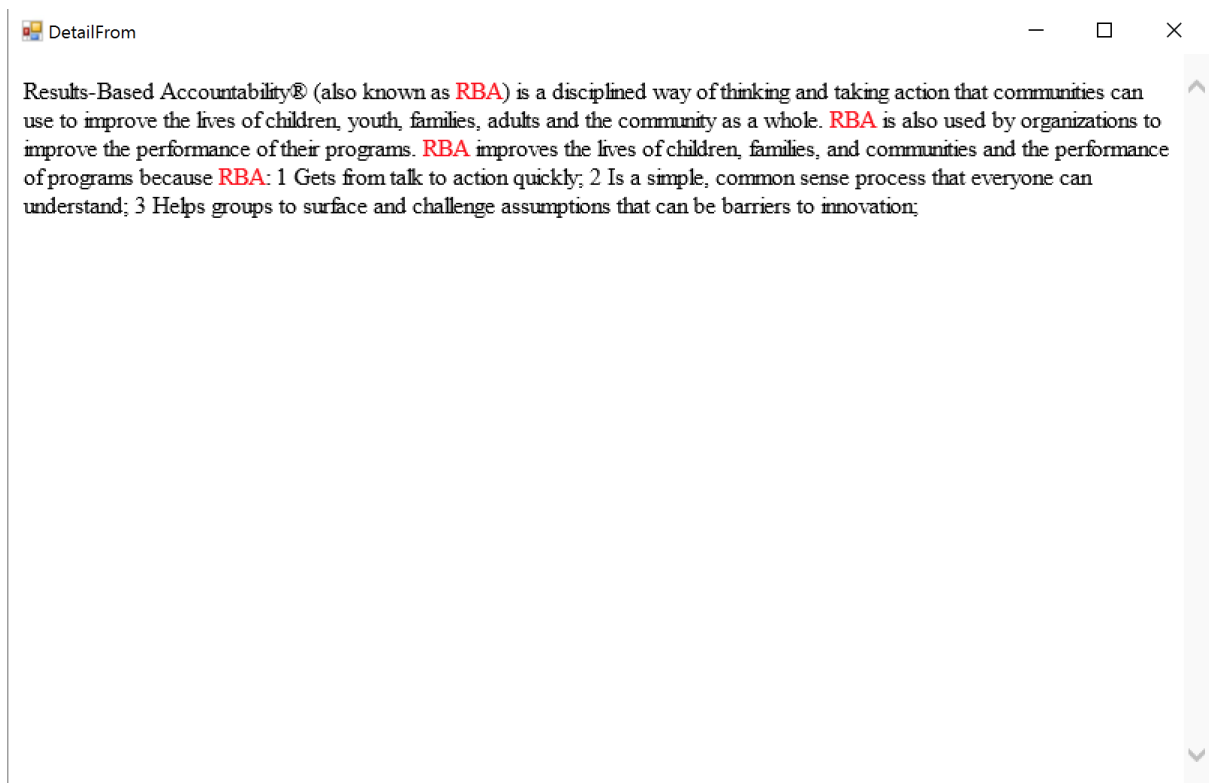
### Saving result

At the right lower corner, there is a text field for input the id and a button for saving the result, it needs to select a location for saving.

The search result list will be writing into a text file with passage ID, rank and score of the relevance for each object in the list.

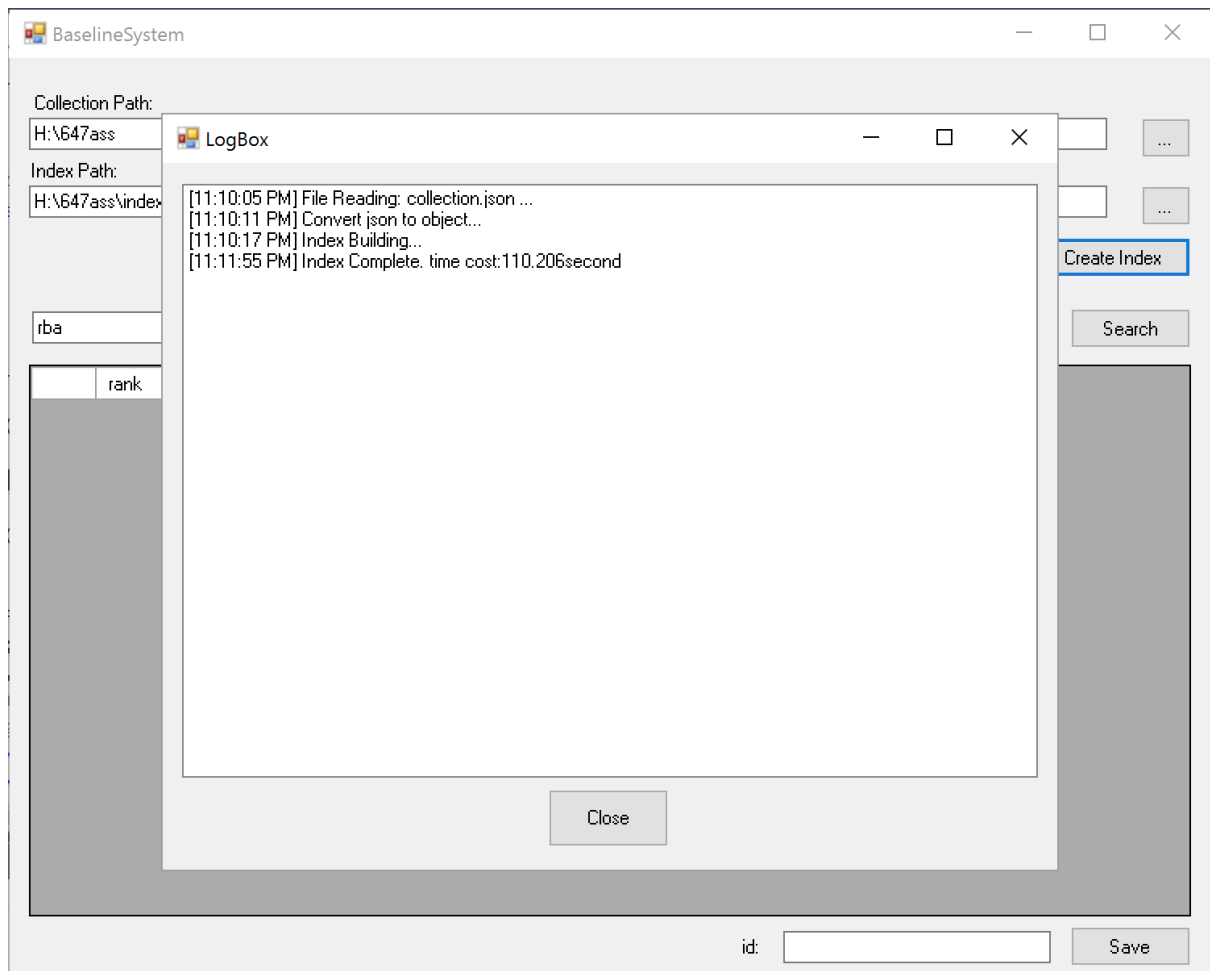
### 6.1.2 Detail information

By simply selecting an option in the result list, a detailed form will pump out, shows the full passage with red highlighted key terms.



### 6.1.3 Log box

A log box is for the present the command progress, with the time shows how long it will take for creating the index and search.





## 7. Advanced Features

For achieving the goal of making a system be able to answer the question rather than return a source, it is crucial to consider the concept of Natural Language Processing. Natural Language Processing is a range of intelligent techniques that providing human-like processing for a text that will be able to build in the applications. (Liddy, 2001) we have considered two approaches implement to our system.

### 7.1 POS tagging

POS tagging represents the Part-Of-Speech tagging, which is a commonly used tool for natural language processing. The function is to tagger the words with its part-of-speech, including a verb, noun or adjective, etc., used for corpus search. (Toutanova, 2003)

POS tagger will make a text meaningful and to be understandable instead of matching the terms, as pre-processing for a Question Answering.

For implementing POS tagging in our system, A Stanford log-linear POS tagger for the .NET opensource library can be utilised in our system by first downloading package and import library, making a method in the LuceneSearchEngine.cs class to load it, and to be added as apart of processing.

### 7.2 WordNet

Another advanced feature that was considered is WordNet. WordNet is a large lexical database of English has been used in areas for decades. It is made use of word similarity determine by clustering the words in term of the synonyms, the synonyms group will be called a synset. It can be used for various purposes including text classification or automatic text summarization, for information retrieval, it can play the role of a text parser.

WordNet can be used as Semantic approaches for our system, by disambiguating the various senses. "it can be used to get the conceptual information of each word in the given query context." (Gharat & Gadge, 2012) In Tari's study, they utilize the hypernyms of a noun to disambiguating. In some cases of a natural language, A verb or noun can indicate different meaning regards to the context, this is making an unexpected result when a user send query because of the variance of meaning. That is why it is important to disambiguate the implications of nouns and verbs for generating correct reasoning. (2006) wordnet is providing such a solution to assign possible senses.

By a search of a method using WordNet as a query expansion approach in C#, we found several third-party API such as Aspose or ebswift. A suggestion is to use Nuget to download the package, then we can add the package in LuceneSearchEngine public class. By processing the terms with WordNet we can generate a Concept-based-term-weighing for the query to achieve ontology retrieval for Question Answering.

## Reference list

Liddy, E. D. (2001). Natural language processing.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003, May). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1* (pp. 173-180). Association for computational Linguistics.

Baral, C., & Tari, L. (2006, December). Using AnsProlog with Link Grammar and WordNet for QA with deep reasoning. In *9th International Conference on Information Technology (ICIT'06)* (pp. 125-128). IEEE.

Gharat, J., & Gadge, J. (2012). web information retrieval using WordNet. *International Journal of Computer Applications*, 56(13).