



## Lucene Search Engine Development: A Beginner's Experience

**Azilawati Azizan**

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Perak Branch Tapah Campus,  
Perak, Malaysia

[azila899@uitm.edu.my](mailto:azila899@uitm.edu.my)

**Najwa Izzah Najihah Mohd Sanusi**

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Perak Branch Tapah Campus,  
Perak, Malaysia

[2017769845@isiswa.uitm.edu.my](mailto:2017769845@isiswa.uitm.edu.my)

**Nurkhairizan Khairuddin**

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Perak Branch Tapah Campus,  
Perak, Malaysia

[nurkh098@uitm.edu.my](mailto:nurkh098@uitm.edu.my)

**Ana Salwa Shafie**

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Pahang Branch Jengka  
Campus, Pahang, Malaysia

[anas674@uitm.edu.my](mailto:anas674@uitm.edu.my)

---

### Article Info

#### Article history:

Received Aug 31, 2022

Revised Sept 25, 2022

Accepted Nov 5, 2022

---

#### Keywords:

Search Engine

Lucene

Quran Translation

Information Retrieval

Precision Recall

---

### ABSTRACT

Lucene provides a basic library package for building a complete text-based search engine. It can be used in various ways to benefit both researchers and users. However, for a beginner, to create a search engine utilizing Lucene, require a thorough understanding of the procedures and library packages. Therefore, this project seeks to explore and demonstrate the development of a search engine by employing the Malay Quran translation text as the dataset for testing purposes. This project applied the fundamental Information Retrieval (IR) model as the main methodology for developing the search engine. Apache Lucene framework, a full-text search engine library which is written in JAVA was used to construct the whole search engine components namely the indexer, searcher, query processor, and ranker. Then, the developed search engine was evaluated using a standard IR measurement, where it achieved 67% of precision and 32% recall value. This paper provides a basic approach to developing a text-based search engine that can be used for any IR testing purposes. The result of this project may also benefit the IR community in comparing the retrieval performance.

---

### Corresponding Author:

Azilawati Azizan

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Perak branch Tapah Campus

email: [azila899@uitm.edu.my](mailto:azila899@uitm.edu.my)

---

## 1. Introduction

Search engine is an application under the field on Information Retrieval (IR) which is defined as a process of obtaining information resources that are relevant to the information need from a collection [1]. Basically, search engine application can be divided into two, namely a standalone search system and a web search system (or better known as a web search engine). General or commercial web search engines like Google or Bing are the most widely used by the web users today. These search engines are actively used all around the world to retrieve necessary information, documents, journals, images, recipes, reviews and etcetera. Most of the search engines like Google, Bing and other popular search engines provide advance function of searching. They might use different types of techniques or methods to increase their performance and relevancy in their search results.



---

Those search engines are certainly based on the fundamental IR model which commonly involves several major processes such as crawling, indexing, matching, ranking and query processing [2]. All of these major processes are vital for a search engine to function and operate effectively [3]. To construct any of these major processes, a significant amount of knowledge is needed. Hence, it is crucial to comprehend and grasp the knowledge on developing each process very well.

However, a lot of search engine development works have been eased by the Apache Lucene framework, which contains all the necessary development libraries [4]. Yet, beginners may still find it challenging to comprehend, use and apply it. Therefore this paper aims to share the experience of developing a search engine using the Apache Lucene framework from a beginner's perspective by demonstrating each activity step-by-step. In addition to that, a Malay Quran translation dataset that focuses on Juzuk Amma was created for the purpose of testing the search engine.

## **2. Literature Review**

Lucene is a powerful, full-featured text search engine library written entirely in JAVA. It makes it simple to include full-text search into any application [5]. It is free to download and it is an open source JAVA-based search framework that offers Application Programming Interfaces (API) for carrying out typical search and search-related tasks like indexing, querying, highlighting, language analysis, and many others.

The Lucene project was originally a one-person initiative but it has experienced a substantial transformation over the years to become one of the top search solutions nowadays [6]. Literally, it was initially created in 1997 by Doug Cutting as a way to learn JAVA, and in 2001, Doug Cutting contributed it to the Apache Software Foundation (ASF). Until now, the Apache Software Foundation team of contributors and committers has been in charge of maintaining Lucene under the Apache Software License version 2 [6].

In recent years, Lucene has become exceptionally popular and is now the most widely used in IR applications [7]. As of this writing, Lucene's core JAR has more than 100 official releases encompassing major and minor and patch releases, and the file size is mostly less than 50Mb. Lucene has produced many search-based services, including Solr and Elasticsearch [8]. It drives the search capabilities behind many websites and desktop applications, mobile applications, and in many IR research purposes as well. Many of today's most well-known websites, applications and devices, such as Twitter, Netflix, Instagram, LinkedIn and Bloomberg [6] as well as many other search-based applications are powered by Lucene [9].

In addition to Lucene, there are other several open-source search engines with various feature sets, performance traits, and software licensing arrangements. For example the Indri toolkit from the Lemur Project, focuses on language modeling and information retrieval [10]. Following that is Terrier IR, an open-source research and experimentation toolkit that supports a wide range of IR models [11]. Then there's Xpian, a portable IR library created in the C++ programming language that supports probabilistic retrieval models [12].

## **3. Methodology and Implementation**

The architecture of the search engine for this project is illustrated in Figure 1. It shows how the search engine works. It is based on the fundamental IR model. When a user submits its query via the search engine interface, the query will be processed and matched with the related terms in the index files. The index file was created in earlier stage before the user can use the search engine. It is created by performing the indexing process using the data collection from the dataset which contain the Quran translation text. After the matching process complete, the search engine will rank the search results and present it on the search engine result page (SERP) to the user. There are three major stages to completing the project. The stages are as below:

1. Dataset creation - Malay Quran translation of Juzuk Amma
2. Search engine development using Apache Lucene
3. Testing and evaluation using Precision & Recall

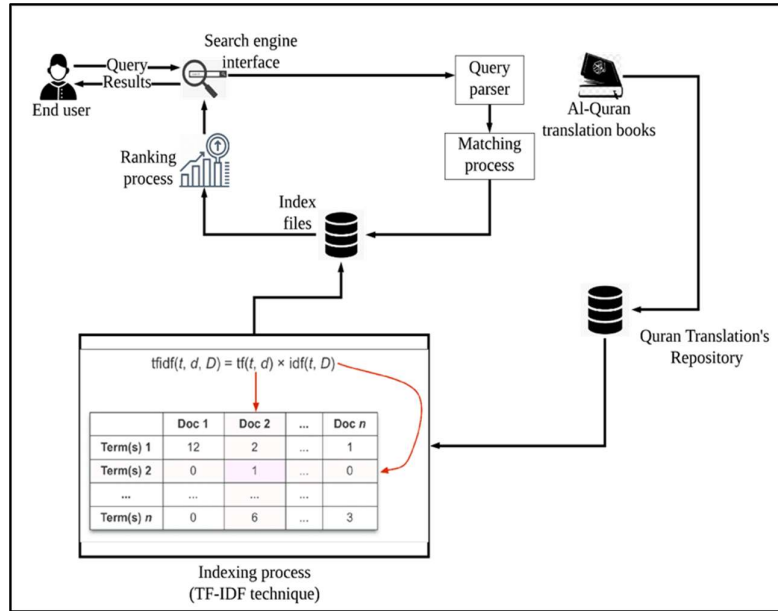


Figure 1. The search engine architecture

A more thorough breakdown of all the activities performed at each stage is described in the next section.

### 3.1 Dataset Creation - Malay Quran Translation of Juzuk Amma

Firstly, a dataset is needed which will be used for developing the index file and testing purposes. Hence, a Malay Quran translation text was chosen to be the dataset in this project. Statistically, Quran has around 78 thousand words and is grouped into verses. Then, a set of verses are grouped into parts, chapters and Hizb quarter. The Quran has 114 chapters (Surahs), 30 Parts (Juzuk), 60 groups (Hizb) and 6236 verses respectively [13]. Meanwhile, Quran translation is the translation of Quran verses in the Arabic language into other languages such as Malay, English, French and others. This translation may assist those who are not familiar with the main language of the Quran which is the Arabic language.

Due to the time limitation of the project, only Juzuk Amma translation text was chosen for the development of the dataset. It is chosen because it is the chapter that is commonly used by many Muslims in their prayer recitations. Juzuk Amma is also widely used for educational purposes such as memorizing or reciting the Surah. Figure 2 shows the Quran translation book that was used as the main source for the creation of the dataset.

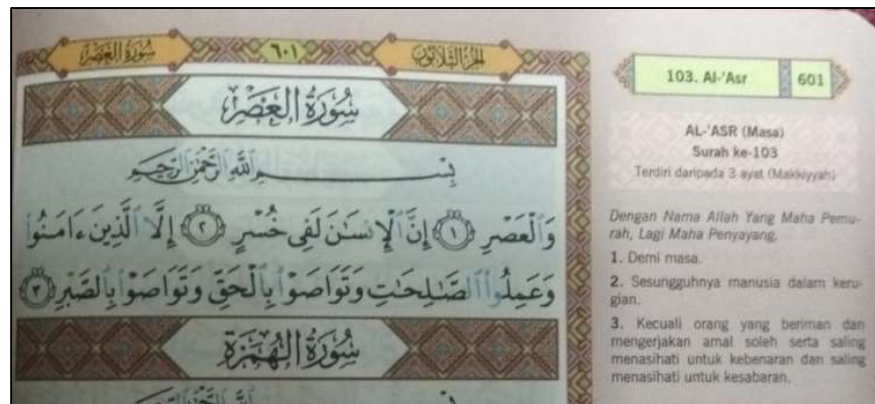


Figure 2. Quran translation in Malay language  
 (Source: Al-Quran Tajwid dan Terjemahan Humaira)

The chosen translations were documented in textual form in a txt file format. The documentation of the translation text has taken almost one and half months to complete. There are 37 surah in Juzuk Amma. All the surah is then separated according to the number of verse in the surah respectively. Figure 3 shows the example of the data that being stored in the dataset.

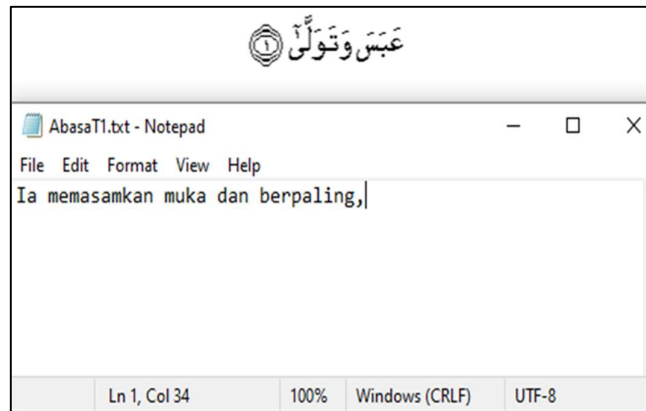


Figure 3. Data collection - surah Abasa, first verse

A total of 561 text files (txt files) were created for the data collection. It is the total number of Quran verses for all 37 surahs in the Juzuk Amma. All the documented translation text files were saved and compiled which will be used in a later stage of the development.

### 3.2 Search Engine Development Using Apache Lucene

This project employed the Apache Lucene framework. It provides a complete library for all basic processes in a search engine to work. The commonly used are the query parser, indexer, searcher and ranker library. In order to start using it, several installations and setups need to be prepared beforehand such as:

- JAVA Development Kit (JDK) – obtained it from the Oracle's JAVA website (<https://www.oracle.com/JAVA/technologies/downloads/>)
- Eclipse IDE – obtained it from the Eclipse Foundation website (<https://www.eclipse.org/downloads/>)
- Apache Lucene Libraries – obtained it from the Apache archive (<https://archive.apache.org/dist/lucene/JAVA/>)

#### Step 1: Setup for JAVA Development Kit (JDK)

The JDK is a JAVA SE Downloads version of the SDK from Oracle's JAVA site. The installation of JDK is completed by following the JDK installation instructions and configuration. Then the PATH and JAVA\_HOME environment variables that refer to the directory which contains JAVA and JAVAc, typically JAVA\_install\_dir/bin and JAVA\_install\_dir were respectively set.

#### Step 2: Setup for Integrated Development Environment (IDE) – Eclipse

The latest version of Eclipse binaries was downloaded from its official website. The binary distribution was unpacked into a convenient location and the PATH variable was set appropriately. Eclipse IDE is used to compile and run the JAVA application.

#### Step 3: Setup for Lucene Framework Libraries

As for the beginner, this project uses the earlier version of Lucene which is 3.6.2. This version is the most stable and versatile. This was suggested by various guides for Lucene's newbie. After unzipping the file, the directory structure is as in Figure 4.

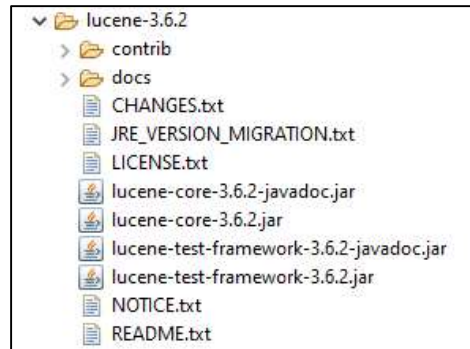


Figure 4. Lucene framework libraries version 3.6.2

After all the setups are completed, the construction of the search engine using the Lucene library starts with the creation of the JAVA project.

#### Step 4: Creating the JAVA Project

A JAVA project was created, and it is named as 'LuceneProject' using the wizard provided in Eclipse IDE. Figure 5 shows the wizard window of 'LuceneProject' in the Eclipse IDE application. Then the following content as in Figure 6 appeared in the Project Explorer tab, which indicates the project is successfully created.

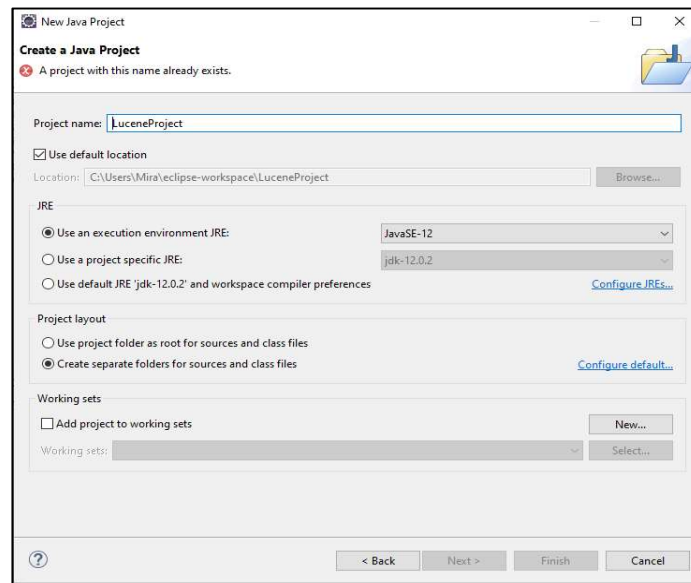


Figure 5. 'LuceneProject' JAVA project

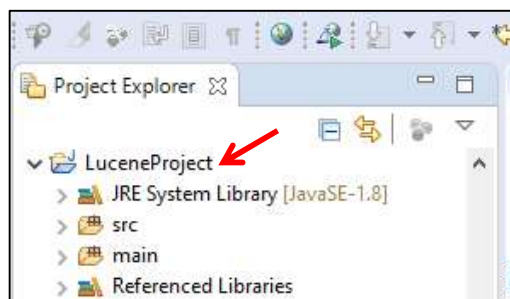


Figure 6. Project Explorer content

### Step 5: Adding the Libraries

Then, the Lucene core framework library was added into the 'LuceneProject'. The insertion of the Lucene core framework library is performed by setting up the build path in JAVA Build Path from the Lucene installation directory within Eclipse IDE. Figure 7 shows JAVA Build Path for adding the library to the project.

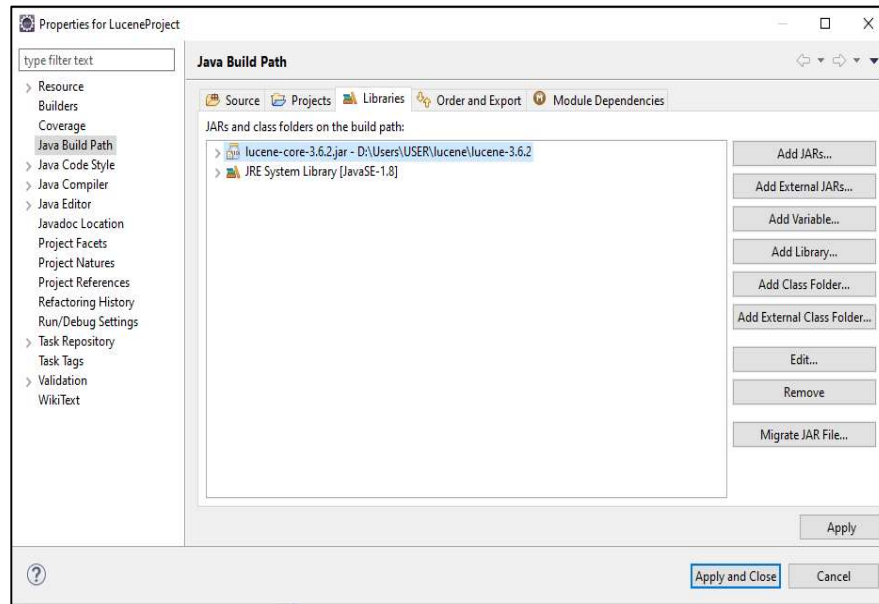


Figure 7. JAVA Build Path window

### Step 6: Creating the Classes

Next is the step for creating the classes under the 'LuceneProject' project. Firstly, a package is created and then several JAVA classes were created under this package that is the LuceneConstant.JAVA, TextFileFilter.JAVA, Indexer.JAVA, Searcher.JAVA and LuceneTester.JAVA.

#### i. LuceneConstant.JAVA

This JAVA class enables the use of various constants across the project. The constants are CONTENTS, FILE\_NAME, FILE\_PATH and MAX\_SEARCH which represent the file of contents, the name of the file together with the path and also the number of retrieved documents whenever searching is done. Figure 8 shows the sample of code in LuceneConstants.JAVA.

```
public class LuceneConstants {  
    public static final String CONTENTS = "contents";  
    public static final String FILE_NAME = "filename";  
    public static final String FILE_PATH = "filepath";  
    public static final int MAX_SEARCH = 10;  
}
```

Figure 8. LuceneConstants.JAVA codes

#### ii. TextFileFilter.JAVA

Next is the text filter class which is used to filter the text file. Figure 9 illustrates the sample codes of TextFileFilter.JAVA.



```

import java.io.File;

public class TextFileFilter implements FileFilter {

    @Override
    public boolean accept(File pathname) {
        return pathname.getName().toLowerCase().endsWith(".txt");
    }
}

```

Figure 9. TextFileFilter.JAVA codes

### iii. Indexer.JAVA

Indexing process is the primary functionalities offered by Lucene [6]. For beginner, it is the most challenging part to understand the flow and the technique used to create and perform the indexing. Basically, this Indexer.JAVA class is used to index the raw data so that the data is searchable using the Lucene library.

Literally, the process for creating index and getting all files in the data directory is performed in this class. It has 5 other classes in it such as the IndexWriter.JAVA, Directory.JAVA, Analyzer.JAVA, Document.JAVA and Field.JAVA. The process begins by collecting the document, analyse it, and process it with IndexWriter.JAVA. The flow and sample of Indexer.JAVA codes are in Figure 10.

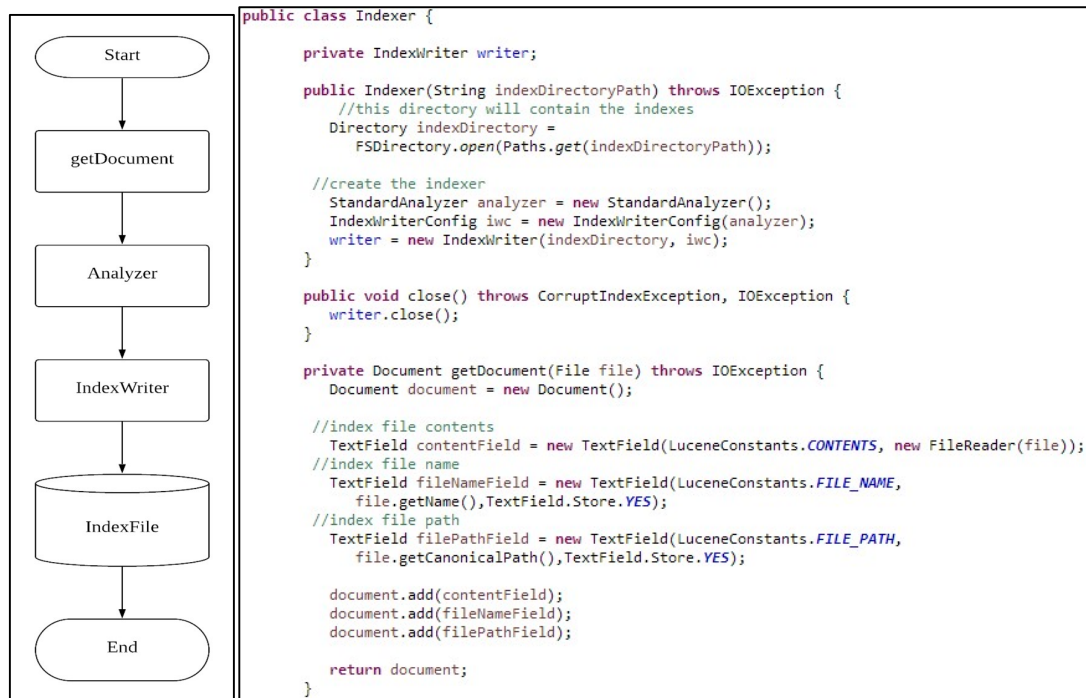


Figure 10. Indexer.JAVA process flow and codes

### iv. Searcher.JAVA

This class is used to locate the indexes created by the Indexer class that corresponds to the user's query. It works whenever it accepts the query string from the user and analyse it using QueryParser class. The Searcher.JAVA configures and instructs the indexSearcher to search the index file for objects linked to the document file. As a result, the user will see the output of the associated document. The Searcher.JAVA procedure is depicted in Figure 11.



Figure 11. Searcher.JAVA process flow and codes

## v. LuceneTester.JAVA

LuceneTester.JAVA is a class that is used to test the indexing and search capability of the Lucene library. Basically, it is a class for displaying the result of the search when a user queries the search engine. It will retrieve the desired information by processing the query string that is submitted by the user. It obtains the data collection path directory and the number of files and displays it to the user if the query string from the user matches the keyword in the index file. Figure 12 illustrates the process flow and the LuceneTester.JAVA codes.

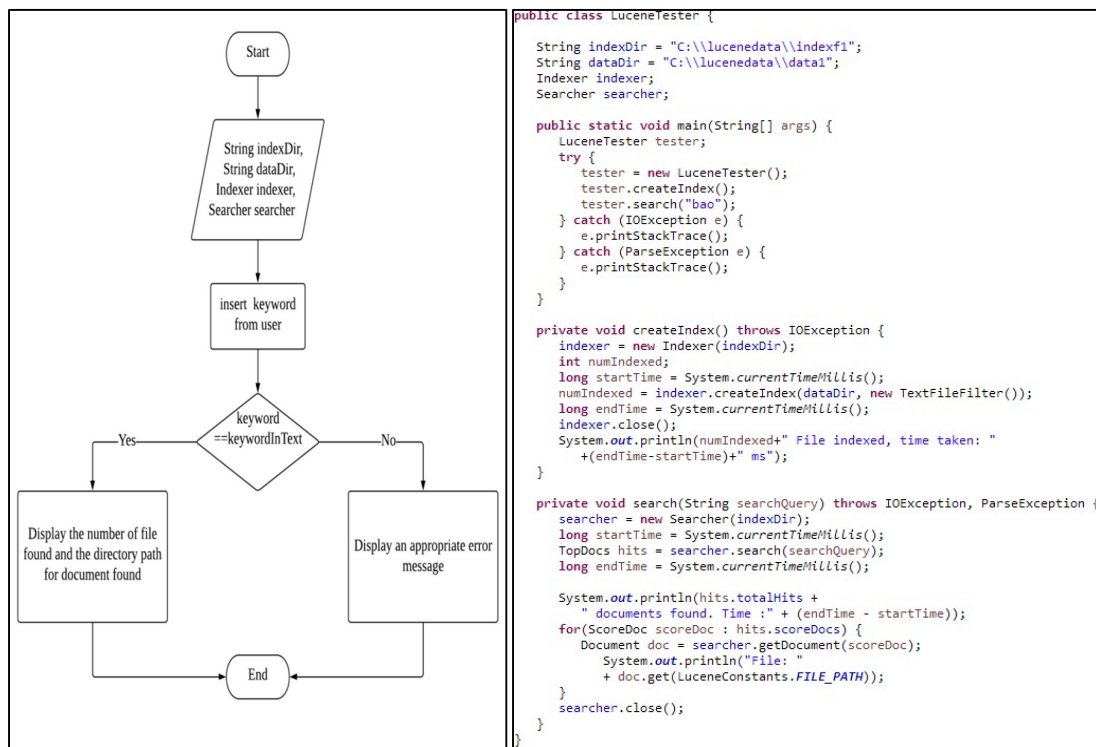
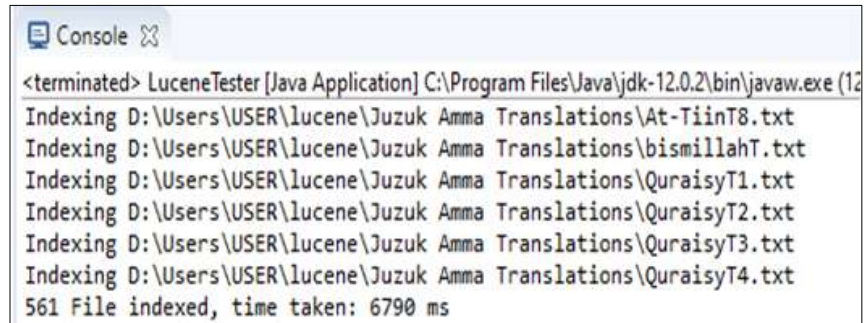


Figure 12. LuceneTester.JAVA process flow and codes



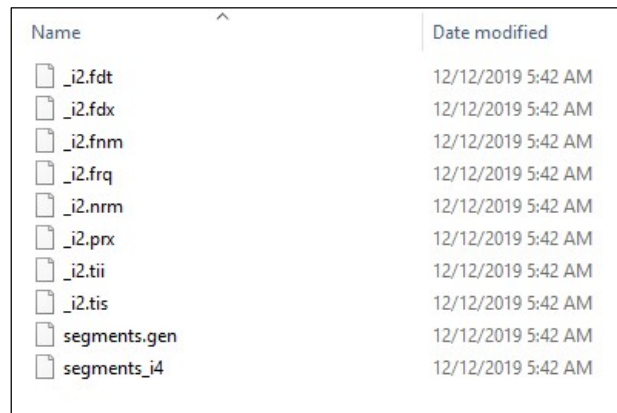
### Step 7: Creating the Index File

The data collection for this project has 561 text files containing the verses of Juzuk Amma translations. It is stored in the local drive of the computer, and the index directory is also created in the same drive. Figure 13 shows the output messages in Eclipse IDE's console when the application runs successfully. When the indexer.JAVA is run, the index file list is created in the same folder drive as shown in Figure 14.



```
<terminated> LuceneTester [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (12
Indexing D:\Users\USER\lucene\Juzuk Amma Translations\At-TiinT8.txt
Indexing D:\Users\USER\lucene\Juzuk Amma Translations\bismillahT.txt
Indexing D:\Users\USER\lucene\Juzuk Amma Translations\QuraaisyT1.txt
Indexing D:\Users\USER\lucene\Juzuk Amma Translations\QuraaisyT2.txt
Indexing D:\Users\USER\lucene\Juzuk Amma Translations\QuraaisyT3.txt
Indexing D:\Users\USER\lucene\Juzuk Amma Translations\QuraaisyT4.txt
561 File indexed, time taken: 6790 ms
```

Figure 13. Output in Eclipse IDE's console when the indexing process is completed



Name	Date modified
_i2.fdt	12/12/2019 5:42 AM
_i2.fdx	12/12/2019 5:42 AM
_i2.fnm	12/12/2019 5:42 AM
_i2.frq	12/12/2019 5:42 AM
_i2.nrm	12/12/2019 5:42 AM
_i2.prx	12/12/2019 5:42 AM
_i2.tii	12/12/2019 5:42 AM
_i2.tis	12/12/2019 5:42 AM
segments.gen	12/12/2019 5:42 AM
segments_i4	12/12/2019 5:42 AM

Figure 14. Index directory content

### Step 8: Run the Application

The last step of the development is compiling and running the application. It is performed once the construction of the classes, the data collection, the data directory and the index directory are completed. It uses the LuceneTester.JAVA class to run the application. The user's query string is submitted to the application and the output is shown in Figure 15.



```
27 documents found. Time :15
File: D:\Users\USER\lucene\Juzuk Amma Translations\Al-BuruujT15.txt
File: D:\Users\USER\lucene\Juzuk Amma Translations\At-TakwiirT20.txt
File: D:\Users\USER\lucene\Juzuk Amma Translations\An-NaaziaatT24.txt
File: D:\Users\USER\lucene\Juzuk Amma Translations\An-NasT3.txt
File: D:\Users\USER\lucene\Juzuk Amma Translations\Al-MutaffifiinT21.txt
<
```

Figure 15. Output in Eclipse IDE's console when the application runs

### 3.3 Testing and evaluation Using Precision & Recall

Once the development is completed, a testing is conducted to evaluate the search engine and its retrieval performance. 15 queries were used to test the search application. The queries are as in Table 1.

Table 1. Test query

Query No	Query String
Query 1	Tuhan
Query 2	Tuhan arasy
Query 3	demi waktu
Query 4	syurga tetap mengalir
Query 5	berpuas
Query 6	bersyukur
Query 7	tuhan mereka maha
Query 8	kitab
Query 9	Demi waktu dhuha sesungguhnya
Query 10	kafir musyrik makhluk jahat
Query 11	TUHAN
Query 12	bpuas
Query 13	bersama-sama
Query 14	bacalah
Query 15	Abasa

All the test queries were submitted to the search engine and the number of relevant and irrelevant results was summed up. It is used to calculate the Precision and Recall value [14]. Precision and recall are the most common measurements used to evaluate retrieval performance for any search application in IR [15]. Value for precision and recall can be calculated as in the equation (1) and (2).

$$Precision = \frac{TP(\text{Number of relevant document retrieved})}{N(\text{Total number of document retrieved})} \quad (1)$$

$$Recall = \frac{TP(\text{Number of relevant document retrieved})}{H(\text{Total number of relevance document})} \quad (2)$$

Precision is the ratio of the number of relevant document to the total number document being retrieved. Meanwhile, recall is the ratio of the number of relevant document retrieved to the total number of relevant document in the data collection. Figure 16 shows the obtained results.

RELEVANCY OF DATA RETRIEVED											
	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	TOTAL RELEVAN
Q1	1	1	1	1	1	1	1	1	1	1	10
Q2	1	0	0	0	0	0	0	0	0	0	1
Q3	1	1	0	0	0	0	0	0	0	0	2
Q4	0	1	0	0	0	0	0	0	0	0	1
Q5	1	1	1	1	0	0	0	0	0	0	4
Q6	1	1	1	1	1	0	0	0	0	0	5
Q7	1	0	0	0	0	0	0	0	0	0	1
Q8	1	1	1	1	1	1	1	1	1	0	9
Q9	0	0	0	0	0	0	0	0	0	0	0
Q10	1	0	0	0	0	0	0	0	0	0	1
Q11	1	1	1	1	1	1	1	1	1	1	10
Q12	0	0	0	0	0	0	0	0	0	0	0
Q13	0	0	0	0	0	0	0	0	0	0	0
Q14	1	1	0	0	0	0	0	0	0	0	2
Q15	0	0	0	0	0	0	0	0	0	0	0

Figure 16. Records for relevancy of results retrieved

Then precision-recall graph is then produced as in Figure 17. The graph indicates that when more results are obtained, the number of relevant result drops. In addition, the graph also explains that the top result obtained are the most relevant, and the lower the order of the results, the less relevant the result is.

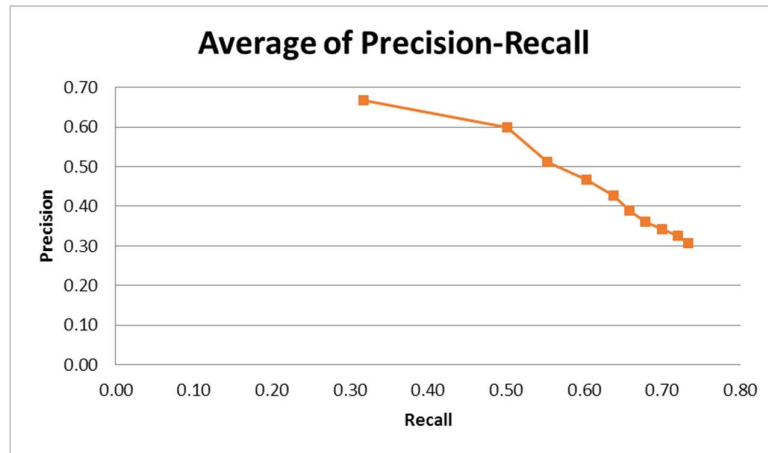


Figure 17. Precision-Recall graph

In average, the search engine scored 67% precision and 32% recall in the evaluation. It means 67% of the search result given by the search engine is relevant, while only 32% of the relevant results from the entire data collection were managed to retrieve successfully. The precision values are generally acceptable, but the recall values are quite low. It is probable that this is due to the small number of data collection. Taken into account the limitation of small data collection and mediocre retrieval performance, this leaves more opportunity for future improvement.

#### 4. Conclusion

This paper seeks to share the experience as a beginner in developing a search engine using the Lucene framework. As a novice in the field of IR, building a search engine is a challenging task. This is due to the fact that the search engine framework is fairly complex. It is made up of various major processes such as indexing, query processing, matching, and ranking, where each process has its own distinct technique. Fortunately, Apache Lucene provides the essential libraries for developing a search engine. Many complicated coding techniques have been simplified. However, in order to utilise the Lucene library correctly, a thorough understanding of the methodology, architecture, and basic operation of a search engine is required.

#### Acknowledgements

The authors gladly acknowledged the student Najwa Izzah Najihah Mohd Sanusi, who developed the application's prototype. Najwa Izzah Najihah Mohd Sanusi graduated in 2021 with a Degree in Computer Science from the Faculty of Computer and Mathematical Sciences. Many thanks are also extended to the Universiti Teknologi MARA (UiTM), Perak branch for providing full support in the completion of this project.

#### Conflict of Interest



The authors declare no conflict of interest in the subject matter or materials discussed in this manuscript.



#### References

- [1] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. ACM Press. Addison Wesley, 1999.
- [2] M. Alecci, T. Baldo, L. Martinelli, and E. Ziroldo, "Development of an IR system for argument search," *CEUR Workshop Proc.*, vol. 2936, pp. 2302–2318, 2021.

- [3] F. Kasmani, R. Maniyar, and M. Narvekar, "Content Based Search Engine for E-Books," *2020 6th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2020*, pp. 528–533, 2020.
- [4] Z. Youzhuo, F. Yu, Z. Ruifeng, H. Shuqing, and W. Yi, "Research on Lucene Based Full-Text Query Search Service for Smart Distribution System," *2020 3rd Int. Conf. Artif. Intell. Big Data, ICAIBD 2020*, pp. 338–341, 2020.
- [5] J. Lin, "A Prototype of Serverless Lucene," 2020.
- [6] A. Bialecki, R. Muri, and G. Ingersoll, "Apache Lucene 4," *Proc. SIGIR 2012 Work. Open Source Inf. Retr.*, pp. 17–24, 2012.
- [7] M. M. Otis Gospodnetic, Erik Hatcher, *Lucene in Action 2nd Edition*, 2nd ed. Simon and Schuster (Manning Publications), 2010.
- [8] A. Grand, R. Muir, J. Ferenczi, and J. Lin, *From MaxSCORE to block-max WAND: The story of how lucene significantly improved query evaluation performance*, vol. 12036 LNCS. Springer International Publishing, 2020.
- [9] P. Yang, H. Fang, and J. Lin, "Anserini: Enabling the Use of Lucene for Information Retrieval Research," *J. Data Inf. Qual.*, vol. 10, no. 4, pp. 1–20, 2017.
- [10] A. Y. Aldailamy, N. A. W. A. Hamid, and M. Abdulkarem, "Distributed indexing: Performance analysis of solr, terrier and katta information retrievals," *Malaysian J. Comput. Sci.*, vol. 31, no. 5, pp. 87–104, 2018.
- [11] J. Lin *et al.*, "Supporting Interoperability between Open-Source Search Engines with the Common Index File Format," *SIGIR 2020 - Proc. 43rd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 2149–2152, 2020.
- [12] W. Iqbal, W. I. Malik, F. Bukhari, K. M. Almustafa, and Z. Nawaz, "Big data full-text search index minimization using text summarization," *Inf. Technol. Control*, vol. 50, no. 2, pp. 375–389, 2021.
- [13] M. Alhawarat, M. Hegazi, and A. Hilal, "Processing the Text of the Holy Quran : a Text Mining Study," vol. 6, no. 2, pp. 2–7, 2015.
- [14] A. Azizan, Z. Abu Bakar, N. A. Rahman, S. Masrom, and N. Khairuddin, "A comparative evaluation of search engines on finding specific domain information on the web," *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 1–4, 2018.
- [15] M. Agosti, G. Maria, D. Nunzio, and S. Marchesin, "An Analysis of Query Reformulation Techniques for Precision Medicine," in *SIGIR*, 2019, pp. 973–976.

#### Biography of all authors

Picture	Biography	Authorship contribution
	Dr. Azilawati Azizan is currently a senior lecturer in the faculty of Computer & Mathematical Sciences, UiTM Perak Campus, Tapah Branch. She obtained her PhD from UiTM Shah Alam Selangor in 2022. Her areas of interest are in text processing, query processing, and search engine. She can be contacted at azila899@uitm.edu.my.	Designing and supervising the project work and drafting article.
	Najwa Izzah Najihah Mohd Sanusi is currently an alumni of Universiti Teknologi MARA Perak Branch, Tapah Campus (UiTM Perak). She was a student of Bachelor Degree of Computer Sciences. Her main research interests are in Information Retireval and web development.	Developing and running the prototype application and conduct the testing for the project work

	<p>Nurkhairizan Khairudin works as Senior Lecturer at Universiti Teknologi MARA, Department of Computer Science. She is currently pursuing her study in PHD in Intelligent System at Universiti Putra Malaysia (UPM). She received Bachelor Degrees in Computer Science and Msc in Information Science from Universiti Kebangsaan Malaysia Her main research interest is in recommendation system and neural network.</p>	<p>Advising the research, editing and revise the article in the scope of IR</p>
	<p>Ana Salwa Shafie is a lecturer at UiTM Pahang, Jengka Campus. She obtained both her Bachelor and Master degree in Computer Science at Universiti Teknologi Malaysia. She has greater interest in sentiment analysis, text processing, artificial intelligence and machine learning.</p>	<p>Framework and final checking.</p>