Intelligent Resume Retrieval Based on Lucence

Jianping Du, Dongping Ma*
Beijing Union University, Beijing, 100101, China.

* Corresponding author. Email: hgtdongping@buu.edu.cn
Manuscript submitted April 15, 2021; accepted June 15, 2021.
doi: 10.17706/jsw.17.1.29-35

Abstract: With the development of Internet, the electronic resume has gradually replaced the paper one. It is the basic requirement of recruitment for enterprises to retrieve the talent information that fulfills the requirement quickly and without omission. Based on the framework of SpringBoot and Lucence full-text search engine, this paper implements a resume intelligent filtering algorithm, which improves the query speed of the system by establishing an index database. At the same time, the scoring function improves the accuracy of the filtering results, reduces the pressure of high concurrency of the database, improves the work efficiency of the Human Resources Department, and avoids the talent loss.

Key words: Full text search, index, resume, lucence.

1. Introduction

With the continuous improvement of computer technology, the way of recruitment is changing gradually from paper resume to electronic resume; sending the resume from face-to-face to online. Nowadays, there are many problems in the recruitment process of HR department in major enterprises, such as the resumes cannot be reused, brain drain and so on. For this reason, large enterprises usually have their own talent retention resume database. However, in order to select the appropriate talents, HR often needs to rely on manual search and preliminary screening in the resume database, to push it to the relevant departments, and need manual tracking and response to all aspects of recruitment. In order to improve the work efficiency of HR department and solve the problems in the recruitment (e.g. complicated process and lack of real-time feedback of information), this article proposes an AI based intelligent resume screening method, which ensures the intelligent resume screening is based on the requirement, and prevent the impact and loss of brain drain caused by the resume not being reused in the resume database [1]-[3].

This paper takes SpringBoot framework and Lucene full-text retrieval technology as the core, designs and implements intelligent resume screening algorithm based on Java language and MySQL data, and provides a platform for enterprises to find talents and push resumes, so as to release the pressure of HR department and create higher social benefits.

Lucene is a sub project of the Apache Software Foundation program. It is an open source toolkit, which is a full text search engine. Lucene uses the method of inverted index. Before the query, it extracts all the content of the query, constructs the text document (body), cuts the content of the document into words, and forms the index (contents). There is an association between the contents and the body. To find the body through the contents is to query the index first and find the document through the index. Remove the repeated words and revoked words, remove punctuation, and convert upper letters to lower letters to reduce the query content [4]-[6]. Full text retrieval is mainly to create an index and then start query, the

specific process is shown in Fig. 1.

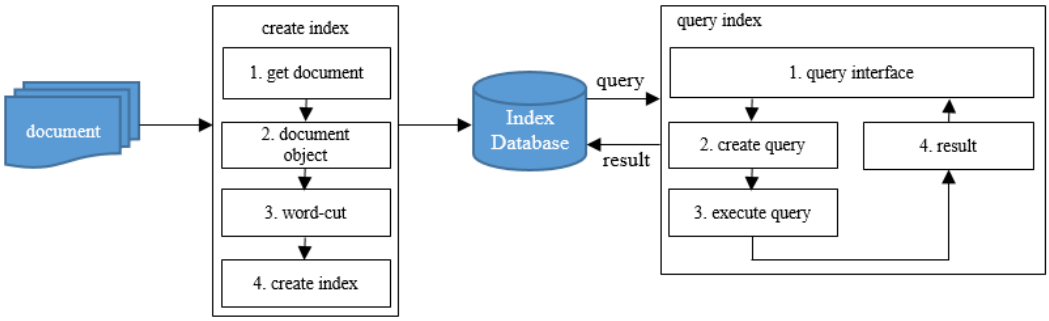

Fig. 1. Lucene full text index workflow.

1.1. Get Original Content

The original content refers to the content that needs to be indexed and retrieved. The original content mainly includes web pages, data in database, files in computer, etc. During the process of data collection, documents can be read through IO stream and imported into index library; data is put into Lucene index library and directly queried from index database, which reduces high concurrent pressure of database; web pages can be crawled and web pages information can be crawled into index library.

1.2. Building Documents

After the first step, the content index is needed. First, a document is created for the original content. There are many fields in the document, and different fields store different contents. Converts the index to data and stores it in the index library, which is created on the local computer disk.

1.3. Analyze Documents

To create the original content as a document in the domain, we need to analyze the contents of the domain again, and then divide the words in the domain.

1.4. Index Document

Index the words analyzed in the previous step, search the keywords, and find the documents for storing relevant keywords.

1.5. Create Query

Before the query, users should create a query object, which are the keywords, query statements, etc. Then the system will generate a query method to search the object.

1.6. Perform Search

According to the query method, find the index related to keywords by inverted index in index database, so that to find the corresponding document, and finally obtain the data in the document.

2. AI Based Intelligent Resume Retrieval

2.1. System Architecture

In the intelligent resume retrieval, the system architecture is mainly contains the controller, data storage object (Dao), service (server), serviceImpl (interface), mapper configuration file [7], [8]. Dao package mainly stores the data objects and writes the interface information used in the system. Service package is an independent module, which is mainly responsible for the business logic, application design and interface

design of the system, and realizes the interface design in serviceImpI. The business implementation in serve layer need to first calls the Dao interface to handle the logic of the business layer. Controller layer is the controller in the system, which controls the system by calling the interface of the business layer. Mapper configuration file is mainly to write SQL statements to query data in database. The whole system hierarchy is shown in Fig. 2.

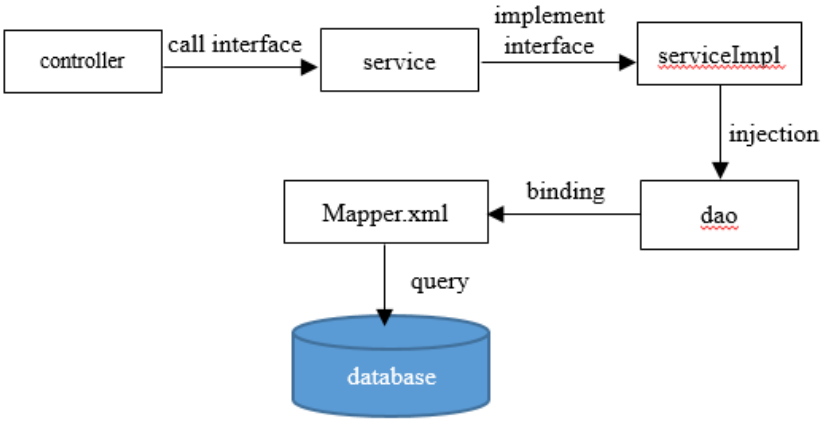

Fig. 2. System architecture of intelligent retrieval.

The system adopts a three-tier architecture. According to the hierarchical structure, the system contents structure is shown in Table 1. Service. ImpI Layer is the interface and implementation of business logic layer, which plays a key role in the system structure. It exists between the control layer and the data access layer, and plays the role of data exchange between the upper and lower layers. Map the data in the database in mapper, and convert the records in the database into objects. Dao layer calls SQL statement in mapper file to query database. The controller control layer integrates the four basic operations of adding, deleting, modifying and checking each module in the system to return the data to the front end. 'Application. properties' define the port number of the system and connect to the database. In the system, SpringBoot framework, MySQL database connection and full-text search Lucene technology are used pom.xml File can be configured to achieve automatic download of jar package.

Table 1. System Organization Division

Directory name		explain
SpringBootDemo		Project name
src/main/java		Contains the main source code
	com.project	Springboot bootloader
	com.project.controller	Business process controller
	com.project.dao	Used to store data access layer
	com.project.entity	interface
	com.project.service.impl	Used to store entity classes
	com.project.util	Business logic layer interface and
	mapper.xml	its implementation
	application.properties	Multifunctional, tool based package
/webapp		Used to store SQL statements

	admin.jsp	Spring configuration file
	css	Main JSP interface files
	images	Store. CSS style sheets
		Used to store system pictures

2.2. Algorithm Design

Lucene mainly realizes the function of intelligent search. It processes the information of recruitment positions by segmenting words, and the segmented words form an index. According to the index, it searches the talent pool to match talents' majors and skills; after matching, it scores the results and ranks them from high to low. The matching keywords are highlighted in red, and the algorithm is shown in Fig. 3.

Input: recruitment position information
Output: matching resume results
1) Create word segmentation: use analyzer word segmentation to create index;
2) Create query object: create a QueryParser query object, and set two parameters. The first parameter is the specified query field, namely special and skill; the second parameter is the word splitter, which is used to segment the "specialty" and "skill" of recruitment information, and match with the index of recruitment information;
3) Create input stream object and search object: write index information into memory to search content;
4) Return result: use Boolean query to traverse all contents and return the query object;
5) Scoring: score the results with scorer algorithm, and display them in order according to the score;
6) Highlight: highlight the matching information in red by using the highlight algorithm.

Fig. 3. Intelligent search algorithm.

2.3. Index Management

In order to speed up the intelligent search of resume content, it is necessary to create and manage the index. The specific steps are as follows:

2.3.1. Create index

First, to create Lucene index, you need to specify a folder to store index files. This article defines a constant variable, and the index created and modified each time is stored in this dictionary. When creating an index and updating an index, firstly load the index into the index reader, then add a document object to the index reader, and then close the index reader to release the related resource information.

2.3.2. Clear index

Before rebuilding the index, you need to clear all the index contents.

2.3.3. Re-index

When it is necessary to rebuild the index of all the data, you can restart the index, send an Ajax request, delete the index in the current index, and then add the resume data to the index again through the loop.

3. The Realization of Intelligent Retrieval

3.1. Lucene Environment Construction

In this article, Lucene version 7.6.0 is adopted, and four related dependencies need to be introduced,

namely lucene-core，lucene-queryparser，lucene-highlighter，lucene-analyzers-smartcn. The functions are Lucene core dependency package, parser, highlight and Chinese intelligent word segmentation pom.xml. The configuration is as follows:

```
<dependency>
        <groupId>org.apache.lucene</groupId>
        <artifactId>lucene-core</artifactId>
        <version>7.6.0</version>
</dependency>
<!--Query analysis of word segmentation index-->
<dependency>
        <groupId>org.apache.lucene</groupId>
        <artifactId>lucene-queryparser</artifactId>
        <version>7.6.0</version>
</dependency>
<!--General word segmentation, suitable for English word segmentation-->
<dependency>
        <groupId>org.apache.lucene</groupId>
        <artifactId>lucene-analyzers-common</artifactId>
        <version>7.6.0</version>
</dependency>
<!--Key highlights -->
<dependency>
        <groupId>org.apache.lucene</groupId>
        <artifactId>lucene-highlighter</artifactId>
        <version>7.6.0</version>
</dependency>
<!-- smartcn Chinese word segmentation -->
<dependency>
        <groupId>org.apache.lucene</groupId>
        <artifactId>lucene-analyzers-smartcn</artifactId>
        <version>7.6.0</version>
</dependency>
```

3.2. Lucene Create Index

To create an index, Lucene needs to specify a folder to store index files. This time, a constant variable is defined, such as:

```
public static final String storepath ="D\\lucene\\IndexDir";
```

The index created and modified each time is stored in this directory. When creating and updating an index, first load the index into the index reader, and then add a document object to the index reader. After adding, remember to close the index reader and release the relevant resource information.The specific process is as follows:

(1) Create index library

First, create an index library on disk

```
public static final String storepath="\\luncence\\IndexDir";
```

Index library is mainly used to store indexes, and storepath indicates the location of index library.

(2)Create an input stream object and read the index information on the disk into memory

IndexReader reader=DirectoryReader.open(dir);

(3)Create an IndexSearcher search object to search for content:

IndexSearcher is=new IndexSearcher(reader);

(4)Create an analyzer word segmentation and segment words according to the query statements

BooleanQuery.Builder booleanQuery= new BooleanQuery.Builder();
SmartChineseAnalyzer analyzer=new SmartChineseAnalyzer();

(5)Create QueryParser query object

The first parameter is query domain, the second parameter is word segmentation. According to the recruitment requirements, match the two functions of specialty and skill in the talent index

QueryParser parser=new QueryParser("special",analyzer);
Query query=parser.parse(requirement);
QueryParser parser2=new QueryParser("skill",analyzer);
Query query2=parser2.parse(requirement);

(6)Search and return results:

booleanQuery.add(query,BooleanClause.Occur.SHOULD);
booleanQuery.add(query2,BooleanClause.Occur.SHOULD);
TopDocs hits=is.search(booleanQuery.build(),100);

(7)According to the matching degree, we recommend talents from large to small

QueryScorer scorer=new QueryScorer(query);
Fragmenter fragmenter=new SimpleSpanFragmenter(scorer);

(8)According to the recruitment needs and professional and skills matching part highlights

SimpleHTMLFormatter simpleHTMLFormatter=new SimpleHTMLFormatter("'","");
Highlighter highlighter=new Highlighter(simpleHTMLFormatter,scorer);
highlighter.setTextFrgmenter(fragmenter);

According to the recruitment requirements, the system uses word segmentation to divide words and match them in talent index. The system sorts from high to low by using the scoring mechanism brought by itself. The recommended resume interface is shown in Fig. 4.

Recruitment requirements: Basic knowledge of network;Skilled Linux operation;Middleware Technique

operate	name	sex	professioal	tel	address	email	resume	score
☑	david	M	Computer software development	15643452314	Beijing	david@163.com	download	9.158452
☐	Tom	M	Computer software development	12812245231	Beijing	tom@163.com	download	7.034293
☐	Lili	F	Computer softwaretesting	15893452313	Beijing	Lili@163.com	download	6.4300396
☐	Sarah	F	Computer science	15878522312	Shanghai	Sarah@163.com	download	5.932408

Fig. 4. Resume screening result interface.

4. Conclusion

Based on the framework of SpringBoot and the configuration of Lucence full-text search engine, the intelligent resume retrieval system realizes the intelligent retrieval of the enterprise internal resume database. According to the publishing needs of recruitment in enterprises, it recommend resumes according to their needs. It can design a resume recommendation and management system to meet the recruitment process of modern enterprises. In the function of intelligent recommendation resume, the query speed of the system is significantly increased by establishing index database instead of SQL query in the database; at

the same time, the scoring function improves the accuracy of the screening results, reduces the pressure of high concurrency in the database, and improves the work efficiency of the Human Resources department.

Author Contributions

Jianping Du conducted the research with Dongping Ma. And Jianping Du wrote the paper. All authors had approved the final version.

Authors Contributions

The author declares no conflict of interest.

References

[1] Ran, L., & Enqiong, M. (2014). Personalized search engine based on conceptual model. *Computer and Digital Engineering*, *42(3)*, 475-477, 516.

[2] Zhiyang, J. (2009). Identification of the Semi-structrured Text. *Beijing University of Posts and Telecommunications*, 20–23, Beijing.

[3] Zhe, F. (2016). Research on personnel resume intelligent extraction system based on conditional random field. *XiAN PolyTechnic University*, 13–16. Xian.

[4] Yong, Y., Dunlap, R., Rexroad, M., & Cooper, B. F. (2006). Performance of full text search in structured and unstructured peer-to-peer systems. *Proceedings of IEEE INFOCOM* (pp. 534–538).

[5] Lux, Z. A., Beierle, F., Zickau, S., & Göndör, S. (2011). *Full Text Search. SQL Server Transact SQL Recipes*.

[6] Zoltán, A. L., Felix, B., & Sebastian, Z. (2019). Full-text search for verifiable credential metadataon. D*istributed Ledgers. Decentralized Identity Foundation*.

[7] Yong, P. X. (2019). Analysis and research of application development technology based on springboot framework. *Computer Knowledge Technology*, *15(36)*, 76-77.

[8] Qian, W., Jie, Y., Xu, H. (2019). Design and implementation of high concurrency e-commerce platform based on SSM framework. *Electronic Commerce*, 63-66.

Jianping Du is an associated professor. Her research interests include data ming, computer application. Now, she majors in algorithm research.