Intelligent Retrieval and Clustering of Inventions

LIAQAT HUSSAIN ANDRABI

Master of Science Thesis
Stockholm, Sweden 2015
Intelligent Retrieval and Clustering of Inventions

Liaqat Hussain Andrabi <andrabi@kth.se>

Examiner: Mihhail Matskin
Industry Supervisor: Fredrik Egrelius
Academic Supervisor: Anne Häkansson
DECLARATION

This thesis is an account of research undertaken between February 2015 and August 2015 at Ericsson AB, Torshamnsgatan 23, Stockholm, Sweden. I hereby certify that I have written this thesis independently and have only used the specified sources as indicated in the bibliography.

Liaqat Hussain Andrabi

Stockholm, August, 2015
I would like to thank and express my deepest appreciation to my supervisors Fredrik Egrelius and Till Burkert, who gave me this opportunity to work in Ericsson and guided me throughout this work. I, not only, learned from them technically but their knowledge and experience has helped me progress professionally. I would also like to thank Rickard Cöster and Remi Tassing for their guidance, ideas, suggestions and valuable feedback during this period. I would also like to thank my examiner Mihhail Matskin and supervisor Anne Håkansson from the KTH Royal Institute of Technology. Their reviews, comments and feedback have helped me write this thesis report in the most professional way possible.

Tack så mycket!

Liaqat Hussain Andrabi

Stockholm, August, 2015
Ericsson’s Region IPR & Licensing (RIPL) receives about 3000 thousands Invention Disclosures (IvDs) every year submitted by researchers as a result of their R&D activities. To decide whether an IvD has a good business value and a patent application should be filed; a rigorous evaluation process is carried out by a selected Patent Attorney (PA). One of most important elements of the evaluation process is to find prior art similar, including similar IvDs that have been evaluated before. These documents are not public and therefore can’t be searched using available search tools. For now the process of finding prior art is done manually (without the help of any search tools) and takes up significant amount of time. The aim of this Master’s thesis is to develop and test an information retrieval search engine as a proof of concept to find similar Invention Disclosure documents and related patent applications. For this purpose, a SOLR database server is setup with up to seven thousand five hundred (7500) IvDs indexed. A similarity algorithm is implemented which is customized to weight different fields. LUCENE is then used to query the server and display the relevant documents in a web application.

**Keywords:** Information Retrieval, Solr, Lucene, Patents, Patent Search Engine, NoSql.
## NOMENCLATURE

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASF</td>
<td>Apache Software Foundation</td>
</tr>
<tr>
<td>IDF</td>
<td>Inverse Document Frequency</td>
</tr>
<tr>
<td>IPR</td>
<td>Intellectual Property Rights</td>
</tr>
<tr>
<td>IR</td>
<td>Information Retrieval</td>
</tr>
<tr>
<td>IvD</td>
<td>Invention Disclosure</td>
</tr>
<tr>
<td>PB</td>
<td>Patent Board</td>
</tr>
<tr>
<td>PU</td>
<td>Patent Unit</td>
</tr>
<tr>
<td>PA</td>
<td>Patent Attorney</td>
</tr>
<tr>
<td>PAR</td>
<td>Patent Archive</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>R&amp;D</td>
<td>Research and Development</td>
</tr>
<tr>
<td>RIPL</td>
<td>Region IPR and Licensing</td>
</tr>
<tr>
<td>TF</td>
<td>Term Frequency</td>
</tr>
<tr>
<td>VSM</td>
<td>Vector Space Model</td>
</tr>
</tbody>
</table>
CONTENTS

DECLARATION...................................................................................................................... I

ACKNOWLEDGEMENT........................................................................................................ III

ABSTRACT............................................................................................................................... V

NOMENCLATURE .................................................................................................................... VII

CONTENTS................................................................................................................................ IX

FIGURES.................................................................................................................................. XII

1 INTRODUCTION .................................................................................................................. 1

1.1 Background....................................................................................................................... 1

1.1.1 How Processing of IvD Works ..................................................................................... 2

1.1.2 Regional IPR & Licensing Tools .................................................................................. 2

1.2 Problem Statement .......................................................................................................... 3

1.3 Proposed Solution ............................................................................................................ 5

1.4 Objective ......................................................................................................................... 6

1.5 Project Limitations ......................................................................................................... 7

1.6 Research Methodology .................................................................................................. 10

1.7 Research Ethics .............................................................................................................. 11

1.8 Report Overview ............................................................................................................ 11

1.9 Author’s Contribution ..................................................................................................... 12

2 RESEARCH TECHNOLOGIES ............................................................................................ 13

2.1 Background ..................................................................................................................... 13

2.2 Apache SOLR ................................................................................................................ 13

2.2.1 Introduction to SOLR ............................................................................................... 13

2.2.2 Why SOLR? .............................................................................................................. 14

2.3 Apache LUCENE .......................................................................................................... 15

2.3.1 Why LUCENE? ........................................................................................................ 15
Figure 1: Example of an Invention Disclosure Template. .............................................8
Figure 2: Example of an Invention Disclosure Template. .............................................8
Figure 3: Example of an Invention Disclosure Template. .............................................9
Figure 4: Example of an Invention Disclosure Template. .............................................9
Figure 5: Architecture of Search Engine Application using TIKA. .............................17
Figure 6: Example of How SOLR can be integrated into application .........................27
Figure 7: SOLR Admin User Interface .....................................................................28
Figure 8: Defining fields with termVectors in SOLR schema. ..................................29
Figure 9: Defining Unique value field in SOLR schema. ...........................................30
Figure 10: Defining fields in SOLR schema. .................................................................30
Figure 11: Predefined requestHandlers in SOLR configuration file. ........................31
Figure 12: User-defined requestHandler in SOLR configuration file. ........................31
Figure 13: Defining dependencies in POM.xml file ......................................................32
Figure 14: Similarity score calculated for each term ...................................................36
Figure 15: Process of matching documents using relevance feedback .......................50
1 INTRODUCTION

The following section(s) provides an in depth view on the background of the approach, the existing problems, their proposed solution and the limitations faced during the course of this Master Thesis project.

1.1 Background

Ericsson AB is one of the leading providers of communication technology and services in the world. With an annual Research and Development (R&D) budget of more than 30 Billion SEK and 25,000 R&D employees, it is ranked 4th worldwide in terms of developing innovative technology through R&D. Having a huge R&D program also means that the company holds a considerable amount of Intellectual Property Rights (IPR). As of now, Ericsson has more than 37,000 granted patents to its name for inventions in the field of information and communication technology (ICT), especially within mobile communication technology across 2G, 3G and 4G wireless networks. Through licensing its patents, the company receives substantial licensing royalties. [1]

Patents are an investment that gives a good return to the company in the form of profit as a result of licensing agreements. In order to create a strong patent portfolio, intellectual property, research and development, and other core units need to develop a well-functioning cooperation. The Regional IPR & Licensing (RIPL) at Ericsson is one of the key components of the company’s global business. RIPL consists of:

- Patent Development
- Strategy and Portfolio Management
- Assertion
- Licensing and Commercialization
- Business Management
- IPR Policy and Communication

The role of the Patent Development organization is to generate and maintain patents in cooperation with R&D units. This includes, handling of invention disclosures, driving patenting processes, as well as supporting the company with providing patent knowledge
and creating IPR awareness. The Patent Development is divided into ten Patent Units (PU:s) across three continents (North America, Asia, Europe). [2]

The PU:s handle Invention Disclosures submitted by researchers as a result of their R&D activities. Every year the Patent Development organization processes over three thousand (3000) IvDs in collaboration with Ericsson R&D units, out of which only forty to fifty (40-50) percent are approved for filing a patent application.

1.1.1 How Processing of IvD Works

An Invention Disclosure (IvD) is submitted and distributed to one of the local Patent Units (PU), if the researcher thinks that the new technology is patentable. In the IvD, the invention is thoroughly described by using an IvD template. It is recommended that before the submission of the IvD, the inventor searches for prior art using Ericsson’s search tools for patent documents and scientific publications, for example Thomson Innovation or IEEE Explore.

Upon reception of the IvD, a unique identification number, a P-number is assigned to the IvD. The next step is to forward the IvD to PU for evaluation where a Patent Attorney (PA) assesses the inventions patentability and business value for the company. One possible outcome of the evaluation is that a patent application is filed. The patent application is drafted by an internal or external PA for submission in one of the patent offices of the world. Once this process is completed, it can take many years, up to 10 years in certain countries, until the patent office takes a final decision to grant or reject the patent application. The patent application is normally published by the patent office eighteen (18) months after filing, in accordance with local patent laws.

1.1.2 Regional IPR & Licensing Tools

During the evaluation phase, a Patent Attorney (PA) has access to several tools that aid in evaluation of an Invention Disclosure (IvD). These include a document handling system, the Patent Archive (PAR), and a Patent Management System called Winpat. These repositories contain data related to each and every received IvD, irrespective of whether a patent application is filed or not. Information such as P-number, application-number (if a patent application has been filed), inventor’s name/department/Identification number, title of the IvD, a brief summary, Patent Attorney (PA), technical classification of the invention etc. is stored in these repositories.
1.2 Problem Statement

Evaluation of an Invention Disclosure (IvD) is one of the most common jobs a Patent Attorney (PA) has to face in Ericsson’s PU:s. An important factor during the evaluation phase is to find related IvDs which have been submitted in the past by the R&D units and can be termed as ‘secret prior art’ since the IvDs themselves are not made public. Even though a Patent Attorney has different search tools available at his disposal such as Patent Management System (Winpat), Thomson Innovation and AutoMatch during the evaluation phase, these tools suffer from the limitation that they do not have access to Ericsson’s internal data since the IvDs are confidential information. Tools like Thomson Innovation and AutoMatch have access only to public patent documents. While the Patent Management System can be used to search the brief summaries and titles of the IvDs using Boolean search, that kind of search is considered by most PAs as too coarse a tool to be really useful and reliable, especially considering that there unfortunately is a fair amount of IvDs which do not have a brief summary. The Patent Archive is, even if it has some search functionality, not seen as useful and reliable either, since its search engine is seen as currently only useful for searching in file names and file types, and not technical descriptions included in IvDs. Hence it is difficult for the PA to find prior art among IvDs for which no patent application has been filed.

Having above implied the desire to be able to in a more accurate way find similar IvDs to a newly submitted IvD, the usefulness and importance of finding related IvDs is briefly mentioned:

- **The double registration problem:** Currently, IvDs are assigned to PUs based on the organization of the inventor(s), projects etc. For various reasons an IvD may end up in two or more PUs at different times or simultaneously. The former case may happen when an inventor aggrieved by a negative evaluation decision from one PA submits virtually the same IvD to another PU in an attempt to get a re-evaluation. Another reason could be that an IvD is sent by mistake to a PA, who may be on vacation, but manage to forward the IvD to a PU for registration, while the inventor, having received an out-of-office message from the PA, forwards the IvD to another email address for registration. The latter case, i.e. simultaneous submission, might happen when the IvD is submitted at the same time to a number of people/patent units in order to make sure (e.g. in the case of vacations/long-term leaves) that the IvD will be treated somewhere. Unaware of that another person in one patent
unit have registered and started to work on the IvD evaluation, another patent unit might register it later and hence cause double-work. While a title search in the Patent Management System might solve this in many cases, employees might forget to do the search and in case of a slight change of the title, a match would not be discovered.

• **The IvD distribution problem:** As mentioned above in section 1.1.1, the distribution to an appropriate PA is done based on a couple of parameters, whereof an important one is the technical experience of the PA based on previously handled IvDs. A PA used to handle IvDs within Data Analytics would simply handle the IvD evaluation of a Data analytics invention more effectively and with better quality than a PA specialized in server cooling systems. Easier recognition of PAs with experience from related IvDs as a newly submitted one would therefore facilitate and speed up the IvD distribution.

• **The experts and classification problem:** When an IvD is received, the PA has, as mentioned above, to find appropriate technical experts and business experts for the disclosed invention. Finding related IvDs to a newly submitted IvD therefore gives guidance on which experts that could be asked for input, since the names of the experts often are logged in previous evaluation decisions. Ericsson has also an internal Technology classification system for IvDs, and finding related IvDs might help the PA to correctly classify the new IvD in that Technology classification system. Moreover, it would also be good to use data analytics to find outliers with respect to how IvDs have been classified in the Technology classification system in order to reclassify them if they are classified wrongly. Furthermore, all IvDs that have led to filed patent applications also are attached in the Patent Management System to a Portfolio of patents, e.g. a group of patents related to Long-term Evolution (LTE) technology. Facilitating the finding of a Portfolio through the finding of related IvDs would also be desirable.

Moreover, other secondary problems and how they can be solved have been recognized with the current Ericsson systems:

• A search in the Patent Management System is considered by the PAs as very slow in comparison with modern search engines. The search is considerably longer if there are many hits or even longer if there are no hits at all. Hence it
would be desirable to find a system where an IvD search could be done not even more accurate but also faster than in the current Patent Management System.

- A search in the Patent Management System generates a plain hitlist of the P-numbers, but to see e.g. the brief summary of the IvDs the user needs to make a number of clicks. Hence it would be desirable to make a hitlist where visual and cognitive appreciation of the displayed data is tailor-made for the PAs when they during the IvD evaluation phase.

- As mentioned above, the searches in the Patent Management System are Boolean, so if a term does not exist at all, then no hits are provided by the search engine. Hence it would be desirable to have a more intelligent system which displays at least a handful of hits regarded as the most similar ones.

- A new Ericsson-internal database may of course not only comprise IvDs, but may be complemented by e.g. patent documents, Ericsson products information etc.

- An new Ericsson-internal database, may in the future, be used for more general invention landscaping in order to answer e.g. questions like: how many IvDs do we have in certain fields, in which sub-areas of the fields are Ericsson strong/weak and what is the progress over time in the respective areas. These analyses can be done already today based on e.g. the Technology classification, but an alternative tool would be good in order to discover sub-fields which are not targeted by the Technology classification.

Furthermore, the data available to build this type search tool is totally unstructured in the form of text and is difficult to manage. Keeping in mind that this kind of information does not fit traditional relational database systems, there is a need of a completely new search engine system design that can handle unstructured text data.

### 1.3 Proposed Solution

Considering the problems explained in section 1.2, the solution to these problems is to first investigate what kind of information is available and how to process it. Usually, unstructured data needs to be preprocessed first because of data quality issues. Next step is to determine the type of search functionality that is required. The final step is to implement an Invention Disclosure information retrieval engine primary targeted at the private data using open source tools.
The following open source tools are expected to be well suited for such kind of information retrieval system and are able to handle unstructured data:

- **Apache LUCENE**: LUCENE is an open source Java-based indexing and information retrieval software library created and managed by the Apache Software Foundation. It is coupled with features such as spellchecking, hit highlighting and advanced text analysis. LUCENE is very commonly used in applications that require full text search capabilities. [3]

- **Apache SOLR**: SOLR is a highly reliable, scalable, fault tolerant and standalone search server managed by the ASF. It provides distributed indexing, load-balanced querying, customized configuration and many more features that allow developers to easily create complex, high performance search applications. SOLR is an open source application built on top of LUCENE. [4]

- **Apache TIKA**: Apache TIKA is a content analysis toolkit created and managed by the Apache Software Foundation. TIKA helps detect and extract text and metadata from different file formats such as spreadsheets, text documents, images and PDFs. The ability to parse different file types using a single interface makes TIKA useful for search applications, which involves indexing and information retrieval and analysis. [5]

The above-mentioned tools are the most significant ones used during the implementation of this project. There are several other tools and technologies that have been used, explained in Chapter 2.

**1.4 Objective**

The purpose of this Master Thesis project is to investigate and develop an information retrieval search tool for unstructured data with an accurate, reliable and consistent similarity algorithm that can intelligently retrieve and cluster similar Invention Disclosures (IvD) or unpublished patent applications. This search engine is targeted towards the private data that is not accessible for any other search tool. The objective is to improve the effectiveness of the relevant retrieved information high by applying different techniques such as boosting. The overall aim is to aid Patent Attorneys in discovering prior art among Ericsson’s internal IvDs and unpublished patent applications. Also, to find different ways of dealing with the unstructured nature of text data considering its quality, lack of formatting and limitation of this project (explained in
section 1.5). In the end, the overall project should take care of most of the problems as explained in chapter 1.2.

1.5 Project Limitations

Because of the sensitive information which is disclosed in IvDs, they are classified as Ericsson internal documents, which results in certain limitations for this project.

First and foremost, the dataset of Invention Disclosures is very small, that is, only seven thousand five hundred (7500) document files are made available for this project. The main reason for this is the data privacy. Due to Ericsson’s confidential data privacy on highly sensitive data such as Invention Disclosures, only documents which are more than eighteen (18) months old are made accessible and most of them are very old. This could well prove to be one of the major factors in calculating similarity. Moreover, there are no electronic copies of older IvDs, many of them are not correctly tagged/classified as IvDs and are therefore difficult to retrieve from the Patent Archive (PAR).

One of the most important limitations is extracting text out of Invention Disclosure (IvD) documents, available in .DOC format. This is because several different templates have been used for the IvDs which have been made available for this project and sometimes inventors use old templates. Therefore, it is difficult to extract and index text from different sections separately. It is important to note that if the fields such as “Background” or “Problem” or “Solution” are indexed separately, it would help in determining a more accurate search result while searching. A primary reason of this is the boosting technique that is going to be used in this project. More on boosting is explained in the later chapters.

The figures below show different formats of Invention Disclosure templates. It is pretty evident that older templates do not have a common structure. Therefore, extracting text out of separate fields is not possible. For example, in case there is a need to extract “Background” text from the given Invention Disclosures, documents based on Figure 1 would return no value.
1  TITLE

2  TECHNICAL FIELD

3  TECHNICAL BACKGROUND
3.1  KNOWN SOLUTIONS
3.2  PROBLEMS WITH KNOWN SOLUTIONS

4  THE INVENTION
4.1  DESCRIPTION
4.2  ADVANTAGES
4.3  BROADENING

**Figure 1: Example of an Invention Disclosure Template.**

<table>
<thead>
<tr>
<th>ERICSSON</th>
<th>Confidential</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVENTION DISCLOSURE</td>
<td>1 (1)</td>
</tr>
<tr>
<td>1</td>
<td>TECHNICAL INFORMATION</td>
</tr>
<tr>
<td>1.1</td>
<td>Name of invention</td>
</tr>
<tr>
<td>1.2</td>
<td>Inventor(s)</td>
</tr>
<tr>
<td>1.3</td>
<td>Background</td>
</tr>
<tr>
<td>1.4</td>
<td>State-of-the-art</td>
</tr>
<tr>
<td>1.5</td>
<td>Problem</td>
</tr>
<tr>
<td>1.6</td>
<td>Solution</td>
</tr>
<tr>
<td>1.7</td>
<td>Merits of the invention</td>
</tr>
<tr>
<td>2</td>
<td>ADDITIONAL INFORMATION</td>
</tr>
<tr>
<td>2.1</td>
<td>Origin of the idea</td>
</tr>
<tr>
<td>2.2</td>
<td>Related documents</td>
</tr>
<tr>
<td>2.3</td>
<td>Reference people within Ericsson</td>
</tr>
<tr>
<td>2.4</td>
<td>Implementation</td>
</tr>
<tr>
<td>2.5</td>
<td>Date of disclosure to the public</td>
</tr>
<tr>
<td>2.6</td>
<td>Standard blocking</td>
</tr>
<tr>
<td>2.7</td>
<td>Markets for the invention</td>
</tr>
</tbody>
</table>

**Figure 2: Example of an Invention Disclosure Template.**
Figure 3: Example of an Invention Disclosure Template.

Figure 4: Example of an Invention Disclosure Template.
Furthermore, direct access to Patent Archive (PA) database and Winpat is not made available during this Master thesis project. One can have access to technical classification data by extracting it in form of an excel spreadsheet. Due to the data quality issues with the database, the excel also has similar data quality problems. Extracted data from patent databases (Patent Archive and Winpat) has certain issues with data quality in terms of formatting etc. Therefore, I have to manually format the data, which consumes a lot of time. However, it would not be a problem for implementing more features in future (explained in section 7.2).

1.6 Research Methodology

Considering the nature of this project, the research methodology that is used is more closely related to the Quantitative approach. The reason for this is that the overall quality and consistency of the similarity score of the Invention Disclosures will verify the end result. Whether or not the developed system correctly calculates similarity is critical. This also means that the philosophical assumptions for this Thesis project is Realism; credible facts and figures are going to be presented that exist in real. The procedure to conduct analysis and present the result is done through experimental research methods. Considering, the main goal is to develop an information retrieval system, therefore, the calculation of similarity score using different variables to see whether the system is performing efficiently or not is important. Moreover, the performance of the system is concluded and verified on basis of a deductive approach. For a large set of data as in this scenario, the only way to verify the research is to implement an experimental research strategy and design control for the system. This is best strategy that could be used to present results for an application like information retrieval system.

Meanwhile, the primary reason for collecting data is to define the problem in more details, as the data to be used for actual simulation is already available in the form of text documents. This is done mainly through questionnaires. For this every Patent Attorney is asked to answer a few questions related to what kind of problems they normally face during the evaluation. On the basis of collected data, it is concluded that need for a search engine tool is a top priority of people who evaluate IvDs. Furthermore, the analysis conducted on collected data is done by performing statistical analysis to see what kind of problems are the most important ones. Similarly for the overall result and performance of the system, a statistical analysis is used to present the overall significance of the system’s results. In the end, the quality of the system is to be measured using two
methods; validity, and reliability. This means that the presented results in the end show that the system is measuring what is expected of it and the results that are presented are consistent.

1.7 Research Ethics

To remain within the Ethical boundaries, all the data collected, analysed and used to represent the overall results during course of this project is kept anonymous and will not be shared with anyone under any circumstances. A non-disclosure agreement is also signed with Ericsson before the beginning of this Master Thesis project. I believe that ethics are a critical part of any research and to be within the defined boundaries is a duty for every researcher. In case of dealing with highly private data such as Invention Disclosures, the ethical part is even more important.

1.8 Report Overview

The thesis report is divided into 8 chapters.

Chapter 2: This chapter is a literature review.

Chapter 3: This chapter explains the technical theory required that helps carry out this project.

Chapter 4: This chapter focuses on the approach and methods implemented in the project.

Chapter 5: This is where the test result is presented.

Chapter 6: This chapter illustrates and discusses the results as well as the concluding remarks.

Chapter 7: This chapter discusses future recommendations and improvements for this project.

Chapter 8: This chapter lists all the reference material used during the course of this project.
1.9 Author’s Contribution

Liaqat Hussain Andrabi has done all work during the duration of this Master Thesis project. In case of occurrence of any problem, it was solved with consultation of my Advisors in Ericsson and also from support forums online.
2 RESEARCH TECHNOLOGIES

The following chapter describes the technologies used to carry out this Master Thesis project and explain why they were chosen to be part of it.

2.1 Background

Search engines nowadays are very effective in serving relevant information to the user, be it a basic keyword search or, for instance, more complex matching of an image. Matching a query term to the indexed data or “documents” may sound simple but in reality it is a lot harder, in particularly considering unstructured text data. Text based sources of information such as email or documents do not operate well with typical relational databases.

Moreover, one of the main reasons for underperformance of searching engines is the poor quality of content mainly because of lack of preprocessing of unstructured content before indexing. Also, when factors like autosuggestions and spelling checks are considered, a need of specialized and powerful data management tools arises. These tools help in understanding, analyzing and restructuring data.

2.2 Apache SOLR

SOLR is a highly reliable, scalable, fault tolerant and standalone search server managed by the Apache Software Foundation (ASF). It provides distributed indexing, load-balanced querying, customized configuration and many more features that allow enterprise developers to easily create complex, high performance search applications. SOLR is an open source application built on top of a Java library called, LUCENE (also managed by ASF).

2.2.1 Introduction to SOLR

One of the main challenges in software industry today is the handling and management of data. In general, with the evolution of Web Applications, Social Media, Cloud Computing and Big Data there’s a special interest in developing non-relational data
storage and processing technology, also known as NoSQL. Powerful data management tools like SOLR, ElasticSearch and CloudSearch have been built to simplify managing unstructured data. These technologies are created to handle this special type of data by keeping the requirements of modern web applications in mind, for example data handling, scalability and availability.

Apache SOLR is based on NoSQL technology made specifically to search large volume of text-centric data. The main properties of SOLR are:

- Highly scalable
- Easy installation and configuration
- Handles large volumes of data
- Fast search response
- Text-centric
- Results sorted on basis of relevancy of user’s query

In other words, SOLR is a powerful search tool for information retrieval for finding unstructured text data coupled with hundreds of useful features that can be used to retrieve top relevant documents. [4]

### 2.2.2 Why SOLR?

During the planning phase of this project, the following requirements were identified:

- Text-centric application.
- Document oriented.
- Flexible schema.
- Should be able to handle large amount of data.

**Text-centric**: Text-centric data means the application should be able to handle text that is extracted from the Invention Disclosures. Typically this type of data is “unstructured” because human-created text documents usually have no consistent structure.

**Document oriented**: In a search engine term, a document is a standalone collection of different fields that hold data. This type of approach is really good for processing Word and PDF files.
**Flexible schema:** Another requirement of the project is that it should have a flexible schema. This means that there’s no need to have a uniform structure, as it is in a relational database. SOLR provides a simple way to define the structure of how the data will be indexed and retrieved. Schema is explained in detail in section 4.2.2.

**Handling of Big Data:** Considering the huge volume of data, the application should be able to handle it in an optimized way.

Furthermore, SOLR is available under standard open source licensing which means that anyone can setup a SOLR cluster without paying any software licensing fees. SOLR is also being used within Ericsson by several departments.

Considering all project requirements, suggestions from other related departments within Ericsson, my supervisor and other stakeholders; it was decided to use SOLR for this project.

### 2.3 Apache LUCENE

LUCENE is an open source Java-based indexing and information retrieval software library created and managed by the Apache Software Foundation. It is coupled with features such as spellchecking, hit highlighting and advance text analysis. LUCENE is very commonly used in applications that require full text search capabilities. The relationship between SOLR and LUCENE is usually explained as a relationship between a car and its engine.

#### 2.3.1 Why LUCENE?

It can be argued that a SQL database could be used instead of LUCENE to achieve the same results. However, there are certain key features of LUCENE that distinguishes it from a traditional relational database.

LUCENE provides SOLR with the core infrastructure for indexing and searching to retrieve relevant information. SOLR exploits a specialized data structure of LUCENE called an Inverted Index for this purpose. An Inverted Index transforms a page-based (Page -> Words) data structure into a keyword-based (Words -> Page) data structure. This way, instead of searching whole index for a specific key term, the index is searched directly. Moreover, the search results returned in response to a LUCENE query are
ranked by their relevance. This means that a relevance score is calculated for each document. On the other hand, in traditional relational databases sorting of results can only be done on columns and scoring is usually trivial and limited. Other than that, performing a search using LUCENE does not require too much off CPU power or memory, except when there is a need to index data. However, this is not the case in traditional databases. [3]

### 2.4 Apache TIIKA

Apache TIIKA is a content analysis toolkit created and managed by the Apache Software Foundation. TIIKA helps detect and extract text and metadata from different files formats such as spreadsheets, text documents, images and PDFs. The ability to parse different file types using a single interface makes TIIKA useful for search applications, which involves indexing and information retrieval and analysis. The figure below explains the architecture of a search engine application where TIIKA is used as an extraction module.

#### 2.4.1 Why TIIKA?

For applications such as search engines and content management systems, Apache TIIKA is very commonly used library for easy extraction of data (mostly text) from various file formats. Specifically for search engine applications where extracted text and metadata is very useful, TIIKA helps index text data into a search engine from documents such as spreadsheets, Word documents and PDFs.

The figure below explains the architecture of a search engine application where TIIKA is used as an extraction module. [5]
2.5 SolrJ

SolrJ is a Java client for SOLR that offers a Java interface to add, update, and query the SOLR index. One of the biggest advantages of using SolrJ is that it hides many complexities such as connecting to the SOLR search server, with simple high-level methods. It uses Apache Commons HTTP Client to connect to the SOLR server. It is designed as an extendable framework to pass request to SOLR server and get a response from it – just like a client-server model. [6]

2.6 Maven

Maven is an Apache build manager for Java projects generally used for project management. Software projects involving Apache technologies recommend developers to use Maven. Maven’s main goal is to describe how the project is built and define its dependencies (Java library or a plug-in). At the same time Maven allow developers to take complete control of the project using minimum development effort in less time. It
also helps in organizing the project into separate folders that makes navigation between different sub-projects easier. [7]

**2.7 Other Tools and Technologies**

Tools and technologies other than the ones mentioned above are:

- **Eclipse:** Eclipse is an integrated development environment that provides a workspace to develop software.

- **Java Web Technologies:** Java web technologies provide a framework to develop web applications that can communicate through a network or server. In this project, three different Java web technologies are used; Java Web Servlet, Java Server Pages and Java Standard Tag Library.

- **Apache Tomcat Server:** Tomcat is an open source servlet container that implements Java servlet and Java Server Pages technologies. This transforms the workspace into a HTTP webserver environment to develop Java web applications.

- **Relational Database:** A relational database in the form of Excel spreadsheets are used to display corresponding Invention Disclosure information with the results. Due to data quality issues, a corresponding MySql database is created but not used as updating and formatting in a spreadsheet is easier.
3 THEORY

The following sections cover the different theories used to for this project and also explain the past research studies carried out in the field of Information Retrieval.

3.1 Background

Over the past few decades, the field of Information Retrieval (IR) has come a long way. From being a mere academic field, it has become one of the most common techniques to access information. A primary example of such systems is search engines such as Google, Bing and Yahoo. Although the origin of IR was to help scientists and engineers to find relevant data, recent evolution in the field of hardware has led to search engine applications. These systems have the capacity to process billions and billions of queries every day in very low response time. Ranking of results is one of the fundamental problems in IR. Ranking means sorting the result set in such a way that the “best” results for the particular user appear at the top. With time and introduction of new data formats, these search engines have evolved by implementing new ranked retrieval algorithms and user behaviour models. [8]

3.2 Ranked Retrieval

Before the evolution of Information Retrieval systems, Boolean expression of query terms was the most common way of retrieving desired information [8], also known as the Boolean Retrieval model. This made query construction a lot difficult as queries incorporated Boolean expressions such as OR, NOT, and AND. These types of systems require users to be specific with their query to limit the number of documents retrieved. Moreover, the corresponding results set documents were not ranked in order of any relevance to the entered query. A lot of training and learning with the system was also required to get necessary results from the system mainly because of the complex query model [9].

Ranked retrieval systems on the other hand sort the list of documents in the result set on the basis of how useful they are corresponding to the query term. A numeric relevancy score is assigned to each document for this purpose. Whenever a query is entered, each term of the query has a certain score (default is 1). The occurrence of query terms, also
known as the Term Frequency Factor, is also one of the factors in calculating the total relevancy score for a given query.

There are many different ranked retrieval models currently being used in Information Retrieval systems for ranking individual documents e.g. Vector Space Models and probability based models such as RF, BM25 and BM25F. However, VSM is the one which is most relevant to this Master thesis.

### 3.2.1 Vector Space Model

The Vector Space Model (VSM) or term query model is used for representing text documents as vectors. It is one of the most commonly used models and a fundamental concept in Information Retrieval. [10] [15]

Each dimension of the vector corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. Several different ways of computing these values, also known as (term) weights, have been developed. One of the best-known schemes is \( tf – idf \) weighting, is explained below. Typically terms are single words, keywords, or longer phrases. If words are chosen to be the terms, the dimensionality of the vector is the number of words in the vocabulary (the number of distinct words occurring in the corpus). Vector operations can be used to compare documents with queries.

**Vector**

We denote the vector of the document \( d \) by \( \vec{V}(d) \), where there is at least one component in the vector for each term. Every component has a term and its corresponding weighting score. The score is calculated on many different factors such as number of occurrences of the term in document \( d \) and the number of documents that have this term. Unless otherwise stated, the weighting-score is computed using the \( tf – idf \) weighting scheme (term frequency – inverse document frequency). \( tf – idf \) is the product of two statistical factors \( tf \) and \( idf \). \( tf_{t,d} \) denotes the frequency of the term \( t \) in document \( d \), whereas, \( idf_t \) represents inverse document frequency for a term \( t \). In mathematical form, the latter can be written as:

\[
idf_t = \log \frac{N}{df_t} \quad (3.1)
\]
The primary reason for using Inverse Document Frequency for calculating weighting score is that all words do not have equal significance. For example, two documents with almost identical terms can have a significant vector difference because one is longer than the other. Therefore, it is much better to calculate relative score rather than the absolute score.

Using Equation (3.1), the mathematical representation of $tf - idf$ (’−’ not to be confused with a ‘minus’) can be given as:

$$tf - idf_t, d = tf_t, d \cdot \log \frac{N}{df_t} \tag{3.2}$$

It is important to note that a set of documents represented in a vector space have one dimension for each term. Therefore, the phrase ‘Fernando is faster than Felipe’ is equivalent to ‘Felipe is faster than Fernando’. When both phrases are represented in a vector space, the relative ordering is irrelevant.

**Similarity**

To minimize the effect of length (number of words) of document $d$, the standard way of computing the similarity between two documents $d_1$ and $d_2$ is to calculate the cosine similarity of them. We denote vector representations of the two documents by $\vec{V}(d_1)$ and $\vec{V}(d_2)$. The cosine similarity is given as:

$$sim(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| |\vec{V}(d_2)|} \tag{3.3}$$

where numerator is the dot product of vector $\vec{V}(d_1)$ and $\vec{V}(d_2)$ and denominator is the product of their Euclidean distances.

In the same way, cosine similarity can be used to calculate the similarity between query and documents. This is considered to be a measure of score for that document relative to the corresponding query.

Meanwhile, we can represent a query as a vector the same way as we represent the documents. In that case, the cosine similarity between query $q$ and document $d$ is given as:
\[ sim(q, d) = \frac{\bar{v}(q) \cdot \bar{v}(d)}{|\bar{v}(q)| |\bar{v}(d)|} \]  

(3.4)

3.2.1.1 Why Vector Space Model?

After comparing different ranking models by conducting different experiments, Frakes and Ricardo Baeza-Yates [11] explain that there are certain trends that stand out. Term weighting based on frequency of the term within a document always improves the overall performance of the algorithm. The Inverse Document Frequency as explained in Section 3.2.1, is very commonly used for this purpose. When the IDF score is combined with the Term Frequency the results are often better. One of the reasons to be sure that this factor would improve the ranking is to use the normalization factor. Applying normalization helps to compensate the document length. It is also important to use different boosting techniques such as document level boosting or query level boosting to improve the overall ranking of the result.

3.3 Ranked Retrieval in LUCENE

LUCENE uses a combination of Vector Space Model (VSM) and Boolean Model to determine how relevant a document is to given query. The Boolean model is used to narrow down the documents according to logic specified in the query. The VSM is then applied to find relevant documents. In addition to this, LUCENE also refines the similarity score for better searching through normalization of vectors and boosting. For example, boosting one or more fields of the document can influence search results. There are three types of boosting techniques that LUCENE provides:

- **Document level boosting:** Boosting the document at the time of indexing makes it more important than others.
- **Document’s field level boosting:** Boosting specific fields of the document at the time of indexing.
- **Query level boosting:** Boosting query terms while searching.

It is important to note that in LUCENE, the scoring objects are called documents and a document is a collection of many different single or multivalued fields. The total is calculated after combining the score of all the fields present in the document. This is also an important advantage as two documents with same text, but one having all the content
in one field and the other in two will return same scores, however with individual field scoring the scores are different because of the normalization factor applied by LUCENE. [12]

For better searching and usability, LUCENE refines the VSM score by using the following conceptual formula for similarity:

$$sim(q, d) = coordFactor(q, d) \cdot queryBoost(q) \cdot \frac{\bar{V}(q) \cdot \bar{V}(d)}{|\bar{V}(q)|} \cdot docLenNorm(d) \cdot docBoost(d)$$  

(3.5)

- Normalizing $\bar{V}(d)$ to unit vector could be a problem, as it would remove all the document length information. This would be ok for documents with some duplicated text. But for documents that have absolutely no duplicate text this would be wrong. Therefore, a new document length normalization factor is introduced that normalizes to equal or larger than the unit vector, called $docLenNorm(d)$.

- $docBoost(d)$ is one of the boosting factors used to calculate similarity/relevance score. At the time of indexing a document, one can specify if one document is more significant than others.

- Similarly, $queryBoost(q)$ defines the boosting factors specified at the time of querying. This type of boosting can either be applied on the whole query or selective fields within a query.

- $coordFactor(q, d)$ is another scoring factor based on how many query terms are found in the specified document. If the document has more query terms, it will have a higher score than the one with fewer query terms.

For implementation, the conceptual formula given in Equation 3.5 is transformed into a more practical one.

$$sim(q, d) = coord(q, d) \cdot queryNorm(q) \cdot \sum(tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t, d))$$  

(3.6)
where \( tf(t \text{ in } d) \) is the term’s frequency, \( idf(t) \) is the Inverse Document Frequency, \( queryNorm(q) \) is the normalization factor, \( coordFactor(q, d) \) is a scoring factor based on how many query terms are found in the specified document, \( t.getBoost() \) is the boosting factor applied at the time of search and \( norm(t, d) \) is the boosting factor applied at the time of indexing.

As explained in section 3.2.1.1, factors such as Inverse Document Frequency, Term Frequency, normalization factor in addition to different boosting techniques can improve the relevance score considerably. This means that using the above-mentioned algorithm would be a good choice for this kind of system. [13]

### 3.4 Ranked Retrieval in SOLR

SOLR search server plays a key role in this Master Thesis project. It allows user to index information in its database as well as searching for relevant information. The most important part of SOLR search server is the eDisMax, an abbreviation for Extended Disjunction Max, query mode. The eDisMax query mode is able to process simple natural language terms across different fields in the document using different boosting values. It also supports boosting for word bigrams, which means that it does not require 100% of the word to be present in the document to boost it. One of the most important parameter of eDisMax is the Query Field \( (qf) \) parameter that contains the “list of fields” of the document and their boost values. The overall concept of boosting is to specify the importance of one or more fields in a query. This makes one field more significant than the other. [14]
The following sections cover the different methods that were used to solve the problem defined in section 1.2. The modeling, design, development and implementation of the application are explained in this chapter.

4.1 Modelling and Design

In general, there are two approaches (under consideration) for dealing with data, traditional relational databases and an unstructured-data approach using SOLR/LUCENE. Although relational databases with their advanced features like portability, availability of GUI-modeling tools and ease of querying are still popular, they are not well suited for all kinds of applications, especially for search engines. The following sections compare both approaches that were considered for the IvD search engine.

4.1.1 RDBMS Approach

While designing a data driven application, the first question that is to be answered is: How will the users find data? Depending on the nature of application, both structured and unstructured approaches can be used. However, if we consider an e-commerce application where the user needs to find, for instance, compatible part for his car, then an RDBMS based system is good because the results given to user are a result of fixed and structured queries. This type of application can work without full-text searching feature or filtering of certain products to help him find the right product. The relational database approach is easier to implement and useful when processing of the data to produce required results is not too complex. [17]

4.1.2 SOLR Approach

Using SOLR server as the central part of the application is altogether different from the traditional database approach. Compared to the relational approach where normalization (not to be confused with similarity normalization) of the data to overcome redundancy and building relations between different entities by using foreign keys (uniquely identifies a row of another table in MySql) is a requirement, there is no such restriction in
SOLR. As explained earlier, SOLR consists of documents based on single-value and multi-value fields that help in building search applications on top of it. This is mainly because of the non-uniformity of the data. SOLR is good at managing diverse unstructured data and also in scenarios where the developer is uncertain about what kind of information will be added to the database.

4.1.3 Using Best of Both Approaches

Both approaches explained above have their advantages and disadvantages. However, for a search engine application where there is a need to store only relevant information that helps in enhancing the overall search result, SOLR should be used. To provide additional information, a relational database can be queried. Overall, the best option is to combine features of both approaches.

The modeling and design of the IvD search engine application are based on the project requirements, recommendations from other related departments of Ericsson that are using SOLR for different projects and literature review. During the planning phase the following requirements were identified.

- Text-centric application.
- Document oriented.
- Flexible schema.
- Should be able to handle large amount of data, aka “Big Data”.

Also considering the nature and quality of data extracted from Ericsson’s internal database, the type of data which is to be processed (extracted text of Invention Disclosures), it was decided that the search engine application is to be based primarily on SOLR and LUCENE. For presenting additional information in the result set (to overcome project limitations) in search results a relational database is to be used. The following figure shows how SOLR can be integrated into an application.
The above figure shows how SOLR can run alongside other client-server applications. For example, the end user application can be considered as an online store application where users can buy different items, e.g. Amazon. The content management system depicts an inventory system for the employees of the store where all product information is stored that could well be based on relational databases. Products’ metadata, however, would be stored in SOLR to exploit its search capabilities.

4.2 Getting Started

The following sections explain the development and implementation process of the Master Thesis project.

4.2.1 Setting up SOLR Server

The first step during the development step is to install and configure SOLR server. The version used for this project is v5.0. There are two modes for running SOLR; Standalone and Cloud mode. The Standalone mode is used for this project. For installing SOLR, its Getting Started Guide is used as a reference. The reference guide [4] is available on the SOLR website. The following figure shows SOLR’s Admin User Interface:
Figure 7: SOLR Admin User Interface.
Once the SOLR server is up and running the next steps are to:

- Define SOLR schema: The schema tells SOLR what type of documents will be indexed. In case of an Invention Disclosure document, the schema fields are P-number (identification number), description, and other extracted metadata.
- Connecting SOLR to the application.
- Index documents in SOLR.
- Using SOLR as a search server for the application.

### 4.2.2 Defining SOLR Schema

The next step after successfully setting up the server is to define the schema. Defining a schema is a way to tell how it should build indexes on inserted documents. The `schema.xml` file contains all the information about the fields every document contains, and how these fields function while inserting documents to SOLR or at the time of querying these fields. By default, many different field types are available in SOLR however new fields can be defined as well. The following figures show some snippet from the `schema.xml` file of the IvD search engine:

The following fields store the id (P-numbers), description (extracted document content) and author (author name) of an Invention Disclosure. It is important to note that the `termVectors` value for these fields is set to `true`. This means that the SOLR server would store `termVectors` for the given field that helps in getting more relevant search result by using the given fields for similarity. `termVectors` can be defined as a data-structure which holds an array of all words in a field and their occurrences, excluding predefined `stopWords`. (example of `termVectors` is given in section 4.4)

```xml
<field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" termVectors="true" termPositions="true" termOffsets="true" />

<field name="description" type="text_general" indexed="true" stored="true" termVectors="true" termPositions="true" termOffsets="true" />

<field name="author" type="text_general" indexed="true" stored="true" termVectors="true" termPositions="true" termOffsets="true" />

<field name="category" type="text_general" indexed="true" stored="true"/>
```

*Figure 8: Defining fields with termVectors in SOLR schema.*
The unique field in this case is the id field which stores the P-numbers.

```xml
<uniqueKey>id</uniqueKey>
```

*Figure 9: Defining Unique value field in SOLR schema.*

There are certain metadata fields that are specifically named to match with metadata while parsing rich text documents such as Word, PowerPoint and PDF.

These fields include title, subject, comments, keywords etc. Some of these fields store multi-values because the text extraction API, Apache TIKA, may return multi-values for them.

```xml
<field name="title" type="text_general" indexed="true" stored="true" multiValued="true"/>
<field name="content_type" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="resource_name" type="text_general" indexed="true" stored="true"/>
<field name="subject" type="text_general" indexed="true" stored="true"/>
<field name="comments" type="text_general" indexed="true" stored="true"/>
<field name="keywords" type="text_general" indexed="true" stored="true"/>
```

*Figure 10: Defining fields in SOLR schema.*

### 4.2.3 Configuring SOLR

The `solrconfig.xml` file is used for configuring SOLR as per application needs. Important SOLR features such as requestHandlers, listeners, RequestDispatchers and Admin Interface can be configured through this file. The following figures show snippets from the solrconfig.xml file.

The default searchHandlers provided with SOLR are `/select` and `/query`. 
<requestHandler name="/select" class="solr.SearchHandler">
    <lst name="defaults">
        <str name="echoParams">explicit</str>
        <int name="rows">10</int>
    </lst>
</requestHandler>

<requestHandler name="/query" class="solr.SearchHandler">
    <lst name="defaults">
        <str name="echoParams">explicit</str>
        <str name="wt">json</str>
        <str name="indent">true</str>
        <str name="df">text</str>
    </lst>
</requestHandler>

**Figure 11: Predefined requestHandlers in SOLR configuration file.**

For IvD search engine application, a more robust requestHandler is defined called /mlt with default values.

<requestHandler name="/mlt" class="solr.MoreLikeThisHandler">
    <lst name="defaults">
        <str name="echoParams">explicit</str>
    </lst>
</requestHandler>

<lst>
    Query settings -->
    <str name="defType">dismax</str>
    <str name="mm">1004</str>
    <str name="q.alt">*:*</str>
    <str name="rows">10</str>
    <str name="fl">*,score</str>
    <str name="qf">id description author</str>
    <str name="mlt.qf">id description author</str>
</lst>

**Figure 12: User-defined requestHandler in SOLR configuration file.**

The /mlt requestHandler uses the built-in class called `solr.MoreLikeThisHandler` to return similar documents based on the query. The search results can be altered by changing default values of the parameters such as:

- **mlt.fl**: Defines the similarity fields to be used while searching.
- **mlt.maxqt**: Defines the maximum number of query terms to be included in a generated query.
- **mlt.boost**: Defines whether to boost the query or not.
- **mlt.qf**: Defines query fields and their respective boosting values. The default value is 1.0 if not defined in solrconfig.xml.
- **mlt.minwl**: Defines minimum word length in the document.
• mlt.interestingTerms: Defines the terms which are interesting for a specific query and also sets boosting value.

The `solr.MoreLikeThisHandler` is an implementation of `TFIDFSimilarity` [10] class implemented in LUCENE. It implements the relevancy using the Vector Space Model (VSM) and also defines the components of Equation 3.6. These methods can be overridden to alter the relevancy score.

During the course of this thesis project, there are number of different values used for common parameters to get the optimum search results using a heuristics approach. The application’s interface also allows the user to change the default values as per his/her needs.

### 4.2.4 Setting up Development Environment

Even though installing and running SOLR is pretty easy, setting up the development environment is one of the difficult modules of this Master Thesis project.

As there are many different technologies used in the project, Maven is used as the build technology as it automates the process of managing 3rd party dependencies. Using Maven, configuring dependencies is very easy. Maven’s `pom.xml` requires that all dependencies are defined in it and the rest of the process is purely automated. It is recommended to check the corresponding compatibilities of 3rd party technologies such as TIKA, SolrJ, and Tomcat with each other (only certain versions are compatible to run together). The following figure shows a snippet from the `pom.xml` file.

```xml
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.10</version>
</dependency>

<dependency>
    <groupId>org.apache.solr</groupId>
    <artifactId>solr-solrj</artifactId>
    <version>5.0.0</version>
</dependency>

<dependency>
    <groupId>org.apache.tika</groupId>
    <artifactId>tika-core</artifactId>
    <version>1.7</version>
</dependency>
```

*Figure 13: Defining dependencies in POM.xml file.*
4.3 Application Development Process

The first step in the application development process involves setting up a client-server model between application, the SOLR-server and the local Tomcat server on which the end-user application or webapp runs. The next steps are to index the data and query it to get relevant information. The following sections explain these steps in detail.

4.3.1 Using SolrJ to connect to SOLR server

SolrJ, a java based API for SOLR, is used to setup connection between the end-user application and the SOLR server. The library comes included with the SOLR package, however, the classpath and other dependencies are required to be setup.

The setting up process is not difficult as it hides most of the complexities. It creates an abstract SolrServer instance that can be used to request the SOLR server through a query and receive a response back over an HTTP connection similar to a client-server model.

```java
String urlString = "http://localhost:8983/solr";
SolrServer solr = new HttpSolrServer(urlString);
```

Creating a SolrServer instance doesn’t create a connection unless an operation of some kind is performed, for example requesting the server for information using the SolrQuery object. When this happens, a new connection is setup, client sends a query request to the server, it gets processed and the response is sent back to the client.

4.3.2 Processing Documents using TIKA

The next step during the application development process is to index the data into the SOLR server. Apache TIKA is used for processing of documents before indexing. TIKA provides a framework to incorporate rich text documents such as .DOC, .DOCX, PDF etc. into the SOLR server using the ExtractingRequestHandler. TIKA uses different file format parsers such as Apache PDFBox and Apache POI to extract text out of documents that can be indexed into SOLR. The extraction of text and metadata from rich documents is achieved through parse function.

```java
void parse( InputStream stream, ContentHandler handler, Metadata metadata,
            ParseContext context) throws IOException, SAXException, TikaException;
```
For extracting text and metadata out of Invention Disclosures, each IvD is passed to the `parse` method one-by-one and the output is indexed into SOLR.

To index documents into the SOLR server, the path of the root directory that contains all the documents to be indexed is passed in. A simple recursive loop takes each file one by one and parses it into plain text or other specified formats using Apache TIKA. This is managed based on the `ContentHandler` sent to the parser. However, using `BodyContentHandler` means that TIKA would return only the content within body of the document as plain text.

The following snippet of code shows how text is extracted from a file using Apache TIKA.

```java
InputStream input = new FileInputStream(new File(pathfilename));
ContentHandler textHandler = new BodyContentHandler(10*1024*1024);
Metadata meta = new Metadata();
Parser parser = new AutoDetectParser();
ParseContext context = new ParseContext();
parser.parse(input, textHandler, meta, context);
```

Each document’s text is then added to the SOLR server in addition to other metadata information and a commit function is called to save the changes.

### 4.3.3 Querying SOLR server

The next step is to query the SOLR server to find relevant documents. At the moment, user can enter the IvD identification number also known as P-number as well as set values for different parameters through the interface.

Once the query is submitted, a connection is created so that client application can communicate with the server. The next step is to create a query object using `SolrQuery` and set a query mode. SOLR has built in support for multiple query modes through its query parser plugin framework. The one used in this project is called eDisMax (explained in section 3.4). Query object allows setting values of different parameters such as number of rows in the result set, interesting fields to look for, setting boosting
values of specific fields or whole query. Following is an example of SOLR query that has been used.

\[
qt=%2Fmlt&mlt.match.include=false&mlt.boost=true&mlt.minwl=3&mlt.mintf=4&rows=10&q=id%3AP35167
\]

As a result, SOLR sends a response back in a QueryResponse object with a list of similar documents based on the similarity algorithm explained in the next section.

## 4.4 Similarity Algorithm

The similarity algorithms used to search for related Invention Disclosures is implemented using LUCENE. It works by comparing the document that is searched to the ones indexed in the SOLR database server. The first step is to tell LUCENE from where to find the related content. In this case, the similar content is already indexed Invention Disclosures. Otherwise, it could be an external file or even a URL.

Next step is to create a *query* instance. In this case, the *query* instance contains the document to be searched. The next step is to define for which *fields* of the document (similar to columns in a relational database) the similarity should be calculated. As per SOLR schema defined in section 4.2.2, the similarity fields are ID, Description etc. It is important to note that, only those fields can be used for calculating the similarity score for which the termVectors value is set to true. The termVectors are used to create a HashMap where key is the term and value is the occurrence of that term in that field. For the following document:

**Author:** Liaqat Hussain Andrabi

**Description:** My name is Liaqat and I am a Master thesis student.

The termVectors would look like:

**Author:**

*Term:* liaqat, hussain, andrabi

*Term frequency:* liaqat [1], hussain [1], andrabi [1]

**Description:**

*Term:* my, name is, liaqat, and, I, am, a, master thesis, student

*Term frequency:* my[1], name[1], is[1], liaqat[1], and[1], I[1], am[1], a[1], master[1], thesis[1], student[1].
For instance in “liaqat[1]” the key is the term “liaqat” and the value is the term frequency.

The next step is to go through all the documents present in the index and check, for each key-pair value, which documents in the index have this term the most. It should be noted that, specific keywords could be ignored using stopWords or setting the value for setMinTermFreq. After that, the Inverse Document Frequency is calculated as explained in section 3.1.1. Once both the scores, TF and IDF are calculated, the similarity score is determined that is a product of Inverse Document Frequency, number of times a term occurs in a document, boosting and the normalization factor. Equation 3.6 denotes the mathematical formula for similarity.

Given the term frequency and the IDF score, the similarity score could look like the following:

<table>
<thead>
<tr>
<th>TERM</th>
<th>NUMBER OF OCCURANCES OF TERM IN DOCUMENT</th>
<th>NUMBER OF DOCUMENTS MATCHING TERM</th>
<th>IDF VALUE</th>
<th>SIMILARITY SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>liaqat</td>
<td>10</td>
<td>52</td>
<td>3.212</td>
<td>32.12</td>
</tr>
<tr>
<td>hussain</td>
<td>5</td>
<td>3</td>
<td>4.141</td>
<td>20.70</td>
</tr>
<tr>
<td>master</td>
<td>4</td>
<td>19</td>
<td>6.898</td>
<td>27.59</td>
</tr>
<tr>
<td>thesis</td>
<td>3</td>
<td>5</td>
<td>4.632</td>
<td>13.89</td>
</tr>
<tr>
<td>student</td>
<td>3</td>
<td>4</td>
<td>2.717</td>
<td>8.15</td>
</tr>
</tbody>
</table>

*Figure 14: Similarity score calculated for each term.*

Once the similarity score is calculated, it can be boosted using different techniques provided by LUCENE, as explained in section 3.2. At the moment, boosting is applied as per user’s settings through the interface as it is difficult to define optimal boosting value for so many fields as default. Other heuristics parameters that a user can set through the application’s interface are minimum length of term, minimum length of words to be considered, minimum document’s frequency etc. Different ways through which the similarity scored can be made more accurate are explained in Chapter 7.

A similar algorithm is used to measure similarity between a query and the documents present in the index.
The results section of the thesis aims to answer the problems explained in section 1.2. Once these questions have been answered the test results of the algorithm will be discussed.

**Does this tool solve the problems with the Patent Management System?** As explained in section 1.2, the Patent Management system is limited in its search functionalities, having ability to search only on IvD’s summaries. Moreover it uses a Boolean search to display results that are not really useful. Patent Archive has similar issues. In contrast, this IvD search engine uses a state-of-the-art information retrieval system as its backend to show details like which unit it belongs to, inventors name, abstract etc. It is also considered to be comparatively reliable considering the test case result shown in the next few sections.

**Does it solve the double registration problem?** The IvD search engine calculates similarity on the basis of analyzing text within the document. If an IvD ends up in two or more Patent Units, the IvD solves this problem by letting the user know that it’s a duplicate document that is already indexed. If an IvD is duplicate the corresponding similarity score would be 1.0 as it would be an exact match, considering no boosting factor is applied. If there is a boosting factor then the similarity score would be more than 1.0. At the moment the system does not let index duplicate documents in the system with the same P-number (unique ID). Tests have been performed by changing the ID of the same IvD and the results were as expected.

**Does it cater for the IvD distribution problem and classification?** At the moment, one of the columns in the result set corresponds to the respective Patent Attorney. If the results show that a certain Patent Attorney has experience in evaluating similar IvDs then it would be better to allot the new IvD to him. However, in reality this process involves many other factors. Therefore a powerful statistical analysis is needed to choose a PA. The statistical analysis for this purpose is part of the future work. In addition to the IvDs the search engine also has a certain number of application documents indexed that are filed patents. If an application document ends up being similar to a search IvD then it would give the user an idea in which technology the IvD could be classified in as well as the Portfolio details.
**Does it solve other secondary problems?** SOLR is a very powerful and efficient search tool having a very low response time. Ever for very large datasets, the search speed is lightning fast. Other secondary problems that include a tailor-made interface for a PA and an intelligent algorithm running at the backend to display similar search results is already there. However, to make the interface more professional and better some work has to be done, which is part of the future implementations.

During the course of this project several different tests have been performed using many different Invention Disclosure documents. Each person in Regional IPR & Licensing at Ericsson was asked to provide at least two (2) groups of two (2) or more IvDs each; documents that they are sure off belong to the same group in terms of research domain, technology and other factors without revealing which group they belong to. The idea behind is to test the application in a totally anonymous way, where one does not know during testing what the search results would be. The following test results are retrieved as top documents for each test case that the system considered being the most relevant to the user query.

### 5.1 Similarity Test Result

The following figures show test results based on their relative similarity scores. For the sake of convenience the following figures only show three columns. The actual result set is somewhat different.

**Group 1:** P28157 and P34923.

Result set for finding similar documents for P28157.
Result set for finding similar documents for P34923.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P34926</td>
<td>I</td>
<td>0.84598243</td>
</tr>
<tr>
<td>P37541</td>
<td>I</td>
<td>0.7207655</td>
</tr>
<tr>
<td>P38277</td>
<td>I</td>
<td>0.6527613</td>
</tr>
<tr>
<td>P38277</td>
<td>A</td>
<td>0.62833086</td>
</tr>
<tr>
<td>P31692</td>
<td>I</td>
<td>0.5852408</td>
</tr>
<tr>
<td>P13570</td>
<td>A</td>
<td>0.55471325</td>
</tr>
<tr>
<td>P36073</td>
<td>I</td>
<td>0.55091935</td>
</tr>
<tr>
<td>P28157</td>
<td>I</td>
<td>0.54145455</td>
</tr>
</tbody>
</table>

It is important to note that a specific IvD could be similar to some other IvD in terms of relative similarity score that is not part of the group. They were later examined to check if the result set is correct or there is a case of False Positive. Furthermore, it can be seen that the similarity score of the documents are not the same in both figures. This is due to the normalization being applied prior to calculating the relevance score. The search results are shown for similarity greater than 0.3(30%).

**Group 2:** P36073 and P31692.

Result set for finding similar documents for P36073.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P31692</td>
<td>I</td>
<td>0.58205503</td>
</tr>
<tr>
<td>P34923</td>
<td>I</td>
<td>0.45474672</td>
</tr>
<tr>
<td>P38277</td>
<td>I</td>
<td>0.39899138</td>
</tr>
<tr>
<td>P38277</td>
<td>A</td>
<td>0.37098864</td>
</tr>
</tbody>
</table>

Result set for finding similar documents for P31692.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P36073</td>
<td>I</td>
<td>0.8670483</td>
</tr>
<tr>
<td>P34923</td>
<td>I</td>
<td>0.3654067</td>
</tr>
<tr>
<td>P38277</td>
<td>I</td>
<td>0.33234242</td>
</tr>
<tr>
<td>P34926</td>
<td>I</td>
<td>0.30111563</td>
</tr>
</tbody>
</table>
**Group 3:** P28698 and P29908.

Result set for finding similar documents for P28698.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P28698</td>
<td>I</td>
<td>0.5041015</td>
</tr>
<tr>
<td>P28089</td>
<td>I</td>
<td>0.42651165</td>
</tr>
</tbody>
</table>

Result set for finding similar documents for P29908.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P28698</td>
<td>I</td>
<td>0.5729396</td>
</tr>
<tr>
<td>P28089</td>
<td>I</td>
<td>0.22402732</td>
</tr>
</tbody>
</table>

**Group 4:** P47231 and P46891.

Result set for finding similar documents for P47231.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P46891</td>
<td>I</td>
<td>0.51991206</td>
</tr>
<tr>
<td>P37260</td>
<td>I</td>
<td>0.344493</td>
</tr>
</tbody>
</table>

Result set for finding similar documents for P46891.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P47231</td>
<td>I</td>
<td>0.92337796</td>
</tr>
<tr>
<td>P40697</td>
<td>I</td>
<td>0.5200025</td>
</tr>
</tbody>
</table>

**Group 5:** P44609 and P44667.

Result set for finding similar documents for P44609.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P44667</td>
<td>I</td>
<td>0.6859234</td>
</tr>
<tr>
<td>P47321</td>
<td>I</td>
<td>0.38868675</td>
</tr>
<tr>
<td>P42852</td>
<td>I</td>
<td>0.30333778</td>
</tr>
</tbody>
</table>

Result set for finding similar documents for P44667.
Group 6: P46560, P46621 and P47402.
Result set for finding similar documents for P46560.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P46560</td>
<td>I</td>
<td>0.6995286</td>
</tr>
<tr>
<td>P47321</td>
<td>I</td>
<td>0.4749532</td>
</tr>
<tr>
<td>P42852</td>
<td>I</td>
<td>0.4114688</td>
</tr>
<tr>
<td>P45699</td>
<td>I</td>
<td>0.3124611</td>
</tr>
</tbody>
</table>

Result set for finding similar documents for P46621.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P47402</td>
<td>I</td>
<td>0.5815847</td>
</tr>
<tr>
<td>P46621</td>
<td>I</td>
<td>0.45657673</td>
</tr>
<tr>
<td>P45389</td>
<td>I</td>
<td>0.36362505</td>
</tr>
</tbody>
</table>

Result set for finding similar documents for P47402.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P46560</td>
<td>I</td>
<td>0.68300426</td>
</tr>
<tr>
<td>P47402</td>
<td>I</td>
<td>0.5724527</td>
</tr>
<tr>
<td>P45389</td>
<td>I</td>
<td>0.3241395</td>
</tr>
</tbody>
</table>

Group 7: P47321 and P35167. (A case of False Positives)

Result set for finding similar documents for P47321.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P47402</td>
<td>I</td>
<td>0.40399215</td>
</tr>
<tr>
<td>P22078</td>
<td>I</td>
<td>0.39242992</td>
</tr>
<tr>
<td>P42852</td>
<td>I</td>
<td>0.38339806</td>
</tr>
</tbody>
</table>
Result set for finding similar documents for P35167.

<table>
<thead>
<tr>
<th>PID</th>
<th>Type</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>P35840</td>
<td>I</td>
<td>0.5142482</td>
</tr>
<tr>
<td>P39371</td>
<td>I</td>
<td>0.46116644</td>
</tr>
<tr>
<td>P34133</td>
<td>I</td>
<td>0.45377594</td>
</tr>
<tr>
<td>P36260</td>
<td>I</td>
<td>0.41343385</td>
</tr>
<tr>
<td>P32318</td>
<td>I</td>
<td>0.40536907</td>
</tr>
<tr>
<td>P25066</td>
<td>I</td>
<td>0.40073448</td>
</tr>
</tbody>
</table>

Observing results of Group1-7, it is clearly evident that IvD search engine gives pretty much the expected results. However, there are some cases where the result contains a *False Positive(s)*. These results are pretty normal and expected from a new search engine where there is always room for improvement. With more data indexed in the database, the overall performance and accuracy would definitely improve.

Overall about 75 tests for different groups of Invention Disclosures were done, out of which 61 show correct result. Out of 14 groups having incorrect results, 10 had false positives while the other 4 were completely incorrect. This shows that more than 81% of the test cases have shown correct results. 13% of the test cases have had a *False Positive(s)* in the search result while a mere 5% of the test cases have incorrect result. It must be noted that at a bigger level where the total number of indexed IvDs would be more than the current count of seven thousand five hundred (7500), the probability of having a false positive or a completely incorrect result would decrease. However, the overall accuracy and performance above 70% is considered to be very good.

### 5.2 Time and Cost Comparison

Of course, it is difficult to calculate the amount of time or the total cost that can be saved using an Invention Disclosure search engine. On typical IvD, a Patent Attorney spends between one (1) to two (2) hours searching for related cases; mostly publically available prior art, that is important for patentability. Finding related IvDs using a search engine would make the evaluation process more efficient. In terms of money, drafting and filing a patent application costs around one hundred thousand (100,000) Swedish Krona (SEK). Now the question arises, what if a very similar IvD already exists? It could be either granted, filed for application or rejected in the past. By using a search engine to find
similar documents from confidential information would definitely have a significant effect on the overall cost.

### 5.3 Survey Results

Once the test cases are done, every person in the Regional IPR & Licensing department is asked to complete a survey. The motivation behind this is to let them individually evaluate the overall performance of the system. The survey results are as follow:

---

**How beneficial is the idea of having an IvD Search Engine to you?**

- **Not beneficial**
- **Somewhat beneficial**
- **Beneficial**
- **Very beneficial**

Answered: 21  Skipped: 0
How easy is it to understand the information on the results page?

Answered: 21  Skipped: 0

- Extremely easy
- Very easy
- Moderately easy
- Not so easy
- Not at all easy

How much do you trust the results? (Similar lVDs)

Answered: 21  Skipped: 0

- A great deal
- A lot
- A moderate amount
- A little
- Not at all

44
In accordance with the research methodology stated in section 1.6, a quantitative approach is used to present the results of this Master Thesis project. On the basis of collected data through questionnaires to define the problem, the overall system does
resolve most of the issues. The application has correctly identified more than 80% of the related Invention Disclosures for different test cases. This can be verified through test results presented above. It also shows that the system is reliable and consistent with its results. Moreover, to verify the given results even more a survey is conducted to quantize the quality of the application. The overall results of the survey are pretty clear. It is safe to say that, the search engine application works according to the expectations of the RIPL department, however, there are certain areas where it can be improved.
6 DISCUSSION AND CONCLUSION

The following sections discuss the results and their conclusion that have been drawn during this Master Thesis project. The conclusions are based on carrying out analysis with an intention to solve the problems mentioned in Chapter 1.

6.1 Discussion and Conclusion

The purpose of this Master thesis project is to create an information retrieval tool to search similar non-public Invention Disclosure documents. At the same time, this would help ease the Invention Disclosure evaluation process and make it more efficient. However, behind the scenes it involves research on how to deal with unstructured data and use it to build an information retrieval system. Managing unstructured data is very difficult especially at an enterprise level. As explained in section 4.1, modeling and design of these kinds of systems is not easy. Using a powerful data management tool like SOLR proved to be really beneficial in handling text data as well as designing the overall architecture for the system. It must be noted that certain different data management tools have been studied as part of technical background for this Master thesis project. But with the given requirements SOLR in combination with LUCENE seemed a perfect fit.

Similarly Vector Space Model, the chosen algorithm for information retrieval or retrieving similar IvDs, also proved to be a great choice. As explained in Chapter 5, more than 80% of the test cases have been proved to be correct. The RIPL staff also verifies these results. Of course to implement such a search engine that is based on just text data is going to have its flaws at the same time. But this means that, the overall accuracy of the system can still be improved using a different strategy, choice of algorithm etc. The similarity algorithm, as explain in section 4.4, used for this Master thesis project is very robust according to the previously published researches and performed experiments. It is also important to consider the limitations of the project, explained in section 1.5. The limitations significantly affect the performance of the system in terms of similarity score as well as the overall efficiency. If in some way these limitations can be overcome the overall results could definitely be improved.
Moving on the result analysis part, more than 80% system accuracy is very good considering all other factors that hinder the overall performance. This means that it could still be improved. Furthermore, the accuracy of the system is just one issue of the overall problem. The system is developed to ease the overall evaluation process. This means that the overall time and cost is also important. As it is impossible to calculate how much time can be saved on individual basis using the application, it can be said that it would still be significant. Furthermore, evaluating the business value of an IvD to conclude whether it is patentable or not could prove to be very beneficial for the company in terms of cost, as explained in section 5.2. This is a very important factor that could save a lot of money for the organization.

Furthermore, the system also takes care of most of the problems explained in section 1.2. Majority of the problems have been solved but at the same time this could be improved by introducing new features in the future. The objective that was defined in the beginning of this thesis project is almost achieved.

The overall conclusion is that this is still a very special type of Information Retrieval system specifically made to process private data such as Invention Disclosures. Although the similarity score can be altered using different similarity algorithms and boosting parameters, given the limitations of implementing this project, it has proved to be working very efficiently in both searching and information retrieval as shown in Chapter 5. Although there are some test cases where the system does not seem to give correct results or show a False Positive, it is safe to say that the overall performance of the application is good. The naive answer to the question "how accurate the system’s accuracy is in terms of results?" is “it can be better”. Of course there is always room for more improvement. If in some way the limitations of the system such as a relatively small data set, can be removed the efficiency is assumed to increase.
7 RECOMMENDATIONS AND FUTURE WORK

In this chapter, recommendations on more detailed solutions and future work/improvements in this Master thesis project are presented.

7.1 Recommendations

It is recommended to use an iterative and incremental development model for further implementation of this application. One suggestion to re-collect requirements is to make a survey to gather more information about existing problems and investigate if they are possible to implement within Regional IPR & Licensing department at Ericsson. Also, collect opinions and suggestions from other relevant departments.

Considering the limitations explained in section 1.5 do influence this project in some way, it is recommended to find a solution in order to minimize their effect. Doing so would not only make it more efficient but also increase the overall performance of the system in terms of accuracy.

7.2 Future Work

Considering this Master thesis project is a proof of concept, there are a number of possible features that can be integrated in future.

Determining Similarity: For now the similarity score is determined by using a similarity model based on different fields of the Invention Disclosures. However, one future improvement could be using a different way of calculating the similarity between Invention Disclosures. Many different aspects affect the overall similarity, for instance, having more fields in the index. At the moment, due to data extraction issue (because of different IvD templates) there are not too many fields to calculate similarity. Thus, it would not be logical to use a totally different algorithm without trying the current one with more data fields. Another improvement for a greater similarity would be to find an optimal boost values for different fields using some statistical analysis.

Data Accessibility: As explained in section 1.5, one of the main limitations of this Master Thesis project is the accessibility of data. This is mainly because of Ericsson’s data privacy policy. However, to scale this project to a greater level in terms of more indexed Invention Disclosures, data has to be made available to the application
developer. This also proves to be one of the factors that affect the similarity score. The similarity score and the likeliness of having relevant documents in the top-10 list of search results will be increased if there is more relevance data indexed.

**Extracting Relevant information out of Text documents:** Another factor that affects the similarity score is the number the fields in the SOLR schema. Extracting content out of similarly structured documents is easier, however, in this case there are many Invention Disclosure templates used. Considerable amount of the total time for this project is spent on trying to extract field’s data separately instead of all content in one field. However, this proved to be unsuccessful. A suggestion for future improvement is to somehow extract content from different IvD templates into separate SOLR fields. This could prove to be an important factor in determining an accurate similarity score to get relevant search results.

**Relevance Feedback:** Choosing a user’s search experience on the basis of relevance feedback could also be an important feature for this kind of search engines. The type of Invention Disclosures a user is searching could be used to create a user specific behavioral model to present him a result according to his needs. Using a relevance feedback system would also improve the accuracy of the search results. This would also help in making the system a user-centric. This means that separate behavioral models are generated for different users to increase the overall accuracy of the system. The following figure shows a process of matching documents using relevance feedback.

![Figure 15: Process of matching documents using relevance feedback.](image)

where Q₀ is user entered query and Q₁ is the combination of user query and the relevance feedback.
Another advantage of having a feedback system is to remove any *False Positives* from the result set. For example, if a user thinks that a certain document is not relevant to the search result he can reduce its significance or completely remove it from search results so that it does not show up next time in the search result.

**Recommendation System for Patent Attorney:** At the moment, when a new Invention Disclosure is submitted it is allotted to a Patent Attorney on the basis of organization of the inventor(s), projects etc. One way this system can help in selecting a relevant PA is by recommending someone who has worked on a similar IvD before. For this, a suggestion for a future feature is to recommend a Patent Attorney on the basis of some statistical analysis. A simple way of doing this is that a new Invention Disclosure is sent to the person who shows up at the top of the search result on basis of similarity score or if someone has evaluated more IvDs in the past even though their similarity score is low. However, this might not be a good solution in the long run. Therefore, by applying some kind of statistical analysis on the result set, if the system can find to whom the new IvD should be sent for evaluation could prove to be a great feature.

**Moving to a Distributed Setting:** To have a better performance for large data sets, in the future the system can be moved to a distributed setting using distributed feature of SOLR, called SolrCloud. It enables to set up a highly available and fault tolerant cluster of SOLR. SolrCloud also automates the process of master-slave communication that can save a lot of time.

**More Advanced User Interface:** To give the application a more professional look, it is recommended to have a user interface that can offer a simple, intuitive and responsive design with more advanced features.


The appendix or appendices is the natural place for detailed or supplementary information that would make the thesis less easy to read if they were given in the previous chapter. Please, give each appendix a suitable title.