Università degli Studi di Padova

Department of Information Engineering

Master Thesis in ICT for Internet and Multimedia

Functionality Analysis and Information Retrieval in Electronic Document Management Systems

Supervisor
Professor Nicola Ferro
Università degli Studi di Padova

Master Candidate
Abdurahman Sadeg T. Kahil

Dedication.

I would like to dedicate this moment to thank those who helped me in this journey; my family, friends, colleagues, and educators. Thank you all for your love and support throughout this journey.

Abstract

A document management system (DMS) is nowadays one of the most impactful organisational tools that an enterprise may be dependent on. De Angeli Prodotti (DAP), a manufacturer for overhead conductors, wanted to implement an opensource DMS with functionalities that best fit their needs. We took this opportunity to also test and evaluate the state of information retrieval capabilities of electronic DMSs. A list of possible candidates was collected and each DMS was individually evaluated. The final 5 candidates were then extensively tested for their available functions and their search capabilities. For IR testing, some of the state-of-the-art metrics were used to evaluate the DMSs based on target document retrieval testing. The obtained results were compared to confirm the presence of significant difference. Alfresco Community Edition was the selected DMS for implementation; where several plugins were also installed to to add more desirable features to the system. A backup plan was also implemented to reduce the risk of data loss. IR testing was conducted to compare the implemented Alfresco solution with the search process used by DAP, a simple file system search. The conducted tests showed an increase of 47.3%, 64.4%, 42.3%, 52.8%, 25.0% in the mean reciprocal rank (MRR), mean average precision, average normalized discount cumulative gain (DCG), average expected reciprocal rank (ERR), and the average percentage of retrieved documents respectively. Overall, this means that the adaptation of Alfresco DMS should speed up and enhance the precision and accuracy of the IR search process conducted at DAP.

Contents

Listing of figures

Listing of tables

1 Introduction

Document management systems (DMSs) are nowadays considered an essential organisational resource for medium & large enterprises. They allow institutions to control the out and in flows of documentations regarding their day to day functional procedures. They also help with organizing teamwork oriented tasks by creating a shared workspace for teams and individuals to ease collaborations. There are wide collection of enterprise-ready DMS out there these days and choosing a one that best fits the requirements of a given institution can be a length task; especially given that migrating from one DMS variation/brand to another can be an exhaustive and rather tedious procedure. For that reason, an extensive research and testing are required before deciding on an implementation of a given DMS.

DMSs have many functionalities that can assist an enterprise as a whole or individuals in a given workspace. They can reduce clutter and the need of storage space, enhance document search capabilities, assist in document versioning, ease communications between employees, and much more. But one of the most important functions is mainly related to the document search engine and how well it helps in information retrieval. That is, given that a document is to be the requested result from a search query, how well would a given DMS be able to distinguish (retrieve) the document and other relevant ones within the stored archive?. This research question alongside the research procedure to determine the DMS with the best functionalities for given requirements by an enterprise were the main push fot performing this dissertation.

The enterprise in discussion is De Angeli Prodotti Srl (DAP), an innovator in the area of

electrical and aerial conductors manufacturing. They aim to be the manufacturing leader, in terms of quality and innovation, of both insulated and enamelled wires for electromechanical applications and overhead power line conductors. To assist in achieving that aim, an organisational element is needed, and for that an implementation of a DMS is a necessity. DAP also had an additional requirement for their DMS, it has to be opensource, that is, completely free to use and modify to satisfy their current needs and potentially some of their future ones too. Luckily, the opensource community in enterprise tools is rather big and resourceful, thus, many great options are readily available to use and modify to satisfy their needs.

One of the main reason behind the availability of such tools and resources is that DMSs are not a new thing. The earliest versions of DMSs were software-based systems that helped in managing paper based documents. The ones we have nowadays and the ones considered in this dissertation are electronic document management systems (EDMSs) thus dealing with non-hard (digital) copies and the transformation of hard one into digital formats. The first versions of such software have been introduced in the early 1990s. Of course, more improvements have been made from their early days till now, mostly focusing on collaboration functionalities and deeper focuses on enterprise solutions.

The International Organization for Standardization (ISO) has previously released several of standards regarding document and records management. Some of which were withdrawn in place of newer one to keep with the state of the art of the industry. The ones that were considered to be mostly relevant to the application in hands are **ISO 23950:1998**, **ISO 15489-1:2016**, and **ISO 9001:2015**

- ISO 23950:1998 deals with the application service definition and protocol specification for information retrieval. The protocol mostly focuses in defining the specifications for IR including the formats and procedures that should govern the communication between the client and the server holding the required information. These communications enable the client to request the search process within the server based on a given criteria and the to retrieve some or all of the identified records [ISO 23950:1998, 1998].

- ISO 15489-1:2016 defines the concepts and principles from which approaches to the creation, capture and management of records are developed. These include the records, the relevant policies, and the identification of records requirements and control, and the process of creating and managing records [ISO 15489-1:2016, 2016].

- ISO 9001:2015 specifies requirements for a quality management system for enhancing the ability of an organization to meet customer requirements and provides customer satisfaction. All the requirements of ISO 9001:2015 are generic and are intended to be applicable to any organization, regardless of its type or size, or the products and services it provides [ISO 9001:2015, 2015].

In answering the research question regarding the efficiency of retrieving, an Information Retrieval (IR) experimentation was required. The results from this experiment were evaluated based on some of the state of the art metrics collected from recommendations of IR field researchers and previous published works in the field. In doing so, we are evaluating the coherence and the functionality of the search framework applied in the DMS application and how well it performs. The DMS with the best integration and the best search capability was determined based on the results from these tests. In addition to the results from the search process, other functional requirements were also evaluated and scored based on performance that was mostly set by evaluations collected from the potential end-users.

At the end of the evaluation procedure, the selected DMS was implemented and adopted by DAP. The implementation process was also described in this dissertation alongside the proposed maintenance and backup plans. The technical office at DAP supervised my work throughout my stay at their premises to ensure that implementation and testing procedures were up to their standards. Their quality control sector also provided the document dump used in the IR experimentation. The functional requirements by which the DMS selection took place were collected from quality control sector which in place collected them from representatives of each of DAP's main sectors.

One of the other aspects that led to this implementation in DAP is to try to define a workflow method. The management at DAP also wanted to implement a project management system (PMS) later on in the near future and they would like to know how well their employees respond to management-digitizing workflows that were previously handled by word of mouth or emails. In fact, after implementing the DMS selected from this research, I worked on characterizing the available PMS solutions to state my recommendation for their future implementation. The PMS in quest had to be opensource and free to use, same as in the case of the implemented DMS.

This dissertation document starts with a background chapter where we discuss the related definitions required for defining the tasks in hand, then going through a brief history report regarding IR and related conferences and events, some common metrics used in IR experiments, and lastly the objectives and requirements of the requested system. Following the background chapter, we would present the experimentation sections where the list of the tested DMSs is presented alongside the reasoning behind not choosing some DMSs over others, this is followed by brief yet exhaustive description of their installation process in a production environment (not for Mayan-EDMS). Further functionality tests are also described and discussed and the metrics used to run the experimentation are explained in further details. The results chapter, chapter 4, would show the data collected from the testing. The collected data from each of the tested DMSs was statistically analyzed to examine the presence of significant difference. The DMS with the highest scores in functionality and IR tests would be implemented by DAP. The implementation process is explained in details alongside some backup plans put into place (chapter 5). Finally, the dissertation process is concluded by a brief summary of the objectives, methods, and results in the sixth chapter.

2 Background

2.1 Related Definitions

In this section, we will define some of the most common terms that would appear in this dissertation work. All of the upcoming terms are directly related to the main goal of the project, that is choosing, testing, and deploying a Document Management System for De Angeli Prodotti Srl.

2.1.1 Information Retrieval (IR)

IR can be defined as the process of acquiring information resources that are admissible to an information requirement from a set of those resources. This process is made by the means of data searches, these in turn can be based on full-text or other content based indexing techniques of the set of data. In other words, IR is the science of information searching, this includes searching for information inside a document, searching for the document itself inside a database or a file system, and also searching within the metadata for a set of needed data. IR can be automated when a reduction in information overload is required [Manning et al., 2008].

The IR process starts when a query is inserted and a search process is initialized inside the index of the database. It is important to note that an IR query may result in several hits. That is, several objects inside the information set matched the properties made in the query with different degrees of relevancy. It is also important to note that IR queries oppose the concept of classical SQL based queries in the sense that the returned results may or may not match the search query, and for that reason, a ranking system based on relevancy is required. The underlying concept of ranking is that all results retrieved do not have equal value based on a metric, such as relevance [Jansen & Rieh, 2010]. The type of data object is sensibly dependent on the application, it can be a text document, image, audio, ... etc.

The ranking is based on a numeric score calculated by the IR system. This score is a measure on how well an object in the database or the file system matches the queries set by the user. The objects with the highest score for matching are shown to the user. The process of refinement is done by changing some of the parameters set in the queries to iterate the ranking system and thus the shown results. The following figure briefly explains the process of IR. This figure will also be related to later on during this dissertation for further elaboration.

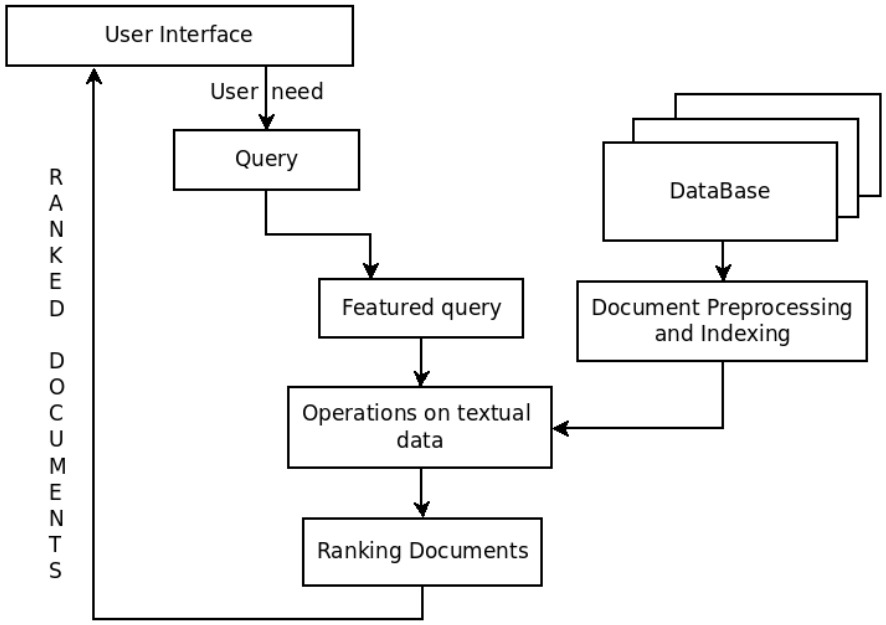


Figure 2.1: The process of information retrieval from text documents.

2.1.2 Document Management System (DMS)

A document management system that is used to reduce the needs of storing hardcopy paperwork by the means of tracking, managing, and storing document digitally. Most of modern DMS can keep records of previous version of the same documents that were modified and created by different system users. The latter is commonly referred to as history tracking. Modern DMS use users' local file system, file systems of remote servers, or database systems to store electronic documents usually in a number of possible file formats. Earlier models of DMS were known as document imaging systems as they were mainly focused on the capture, storage, indexing and retrieval of image file formats.

Nowadays however, DMS applications are highly evolved as they introduced collaboration tools, security and authorization techniques, workflow management systems, and the ability to audit versions of the proposed document before finalizing the submission. Such systems have numerous of other capabilities, those may include, saving copies of documents

in image-like formats, storing the image files in secure repository, capturing received faxes and forms, and are also capable of providing a quick retrieval procedure for requested document. The usage of (Optical character recognition) OCR is exploited in the latter. The retrieval is made possible as the system handles the extraction of the text formatted data from the captured images. Some DMS applications are made to be web-based to ease the access for users not connected to the local network. For these applications, the file format for storage is in the form of html. These are known as policy management system, that require the content to be imported into the DMS for faster retrieval. Therefore, DMSs that are based on html format storing allow for better application of search capabilities such as full-text searching and stemming.

2.1.3 Data Retrieval

Data retrieval in the world of databases simply means the process of obtaining data from a database management system. The data representation in the DBMS is formatted in a structural way while preventing ambiguity in the represented data. For retrieving the data, one must present criteria for the desired data by the means of query. The DBMS selects the desired data from the database using the command query. The retrieved data is projected by the means of storing it in a file, printing a hardcopy, or viewing it on a screen. Queries are prepared by means of a query language such as Structured Query Language (SQL) which was developed by the American National Standards Institute (ANSI) is now the standard language for writing database queries.

The retrieved data are mainly in the form of reports or queries. Overlapping may appear between these two but it is important to note that queries generally select a smaller portion of the database to be the result of data retrieval. Other than having a larger amount of data, queries tend to present the resulting data in its standard format.

2.1.4 Enterprise Search

Is the act of changing a format of contents of databases and intranets from enterprise institutions into a searchable form. In other words, it is the use of information retrieval technology to find information within organisations. It is an area that is of huge importance for businesses, yet, unfortunately, has attracted relatively little academic interest [Kruschwitz & Hull, 2017]. ES can be compared with both web and desktop search as both apply search technologies to find contents of files on the web or in the file system of a single computer respectively. The index data used by ES includes the data from various systems such as file systems, intranets, DMS, e-mails, and databases.

2.2 Brief History of IR and Evaluation Metrics

2.2.1 The Cranfield Tests

After the Second World War, there was a huge increase in the number of publications regarding scientific papers. Scientists who wanted to stay updated about researches in their fields had to use manually created indexes. Of course, at that time, the existing forms of access methods were not electronical and indexing methods were highly dependent on the the massive manually produced indexes provided by specialist librarians. Some of these massive publication indexes still exist till this day, an example of which would be the Medical Subject Index, or the Engineering Index. These indexes however were very expensive to produce and maintain [Manning et al., 2008] [Harman, 2011].

The Cranfield tests were experiments conducted in the field of computer information retrieval in the 1960s whose main purpose was to set an evaluation method for the efficiency of indexing systems. These were conducted at the College of Aeronautics at Cranfield by Cyril W. Cleverdon [Regazzi, 1980]. These experiments led to the establishing of a paradigm for evaluation that is cited regularly to be a "standard" in the evaluation of information retrieval. Cleverdon, a librarian at the College of Aeronautics, Cranfield, England and his staff conducted 2 separate experiments, the first of which started in 1958 and ended in 1962. The set task was to test four different forms of manual indexing techniques. Cleverdon submitted a proposal to the National Science Foundation to work on creating an evaluation method for these indexes. His proposal summarized the numerous aspects required for consideration, these included the number and type of documents which were to be indexed, the indexing systems, the indexer's subject knowledge of the documents and how familiar he or she is with using indexing system, and the type of question to be searched inside the index. He also proposed to test the overall efficiency of the tested systems. He would do so by measuring the time cost to prepare the index and to locate required information via searching through its content as well as measuring the probability of producing the required results while minimizing the irrelevant results [Harman, 2011].

Figure 2.2: Cyril W. Cleverdon, a British librarian and computer scientist who is best known for his work on the evaluation of information retrieval systems.

Cleverdon was assigned with indexing an estimated number of 18,000 scientific papers, most of which were published in the field of aerodynamics. The indexing methods put to test were an alphabetical subject catalogue, a faceted classification scheme, the Universal Decimal Classification and the Uniterm system of co-ordinate indexing; all of which represented the major types of used manual indexing schemes at that time of history. Helping Cleverdon in that process were 3 indexers with variant degrees of expertise. Rotation among the indexers, indexing methods, and papers to be indexing was done regularly and uniformly to eliminate bias. The indexing process was finally accomplished after 2 years of work [Harman, 2011].

The second stage, the searching stage, required thoughtful planning. Many previous experiments failed to answer the question regarding the preferred indexing method in terms of evaluated parameters. To avoid falling in the dilemma, he decided to use what is now known as the *known-item searching*, where he would require to generated indexes to produced the one document that is known to have the required information asked for by the search question (nowadays more relevant to the term query). 1600 search questions were used to maximize the number of data samples for the later stages in significance testing. The search questions were designed by the actual authors of the 18,000 indexed documents to increase the probability that the search questions would generate the required document. In all he received 1500 such questions, which were then subsetted into random batches for various stages of testing. As for the normalization needs, he submitted a batch of 400 questions to a committee of evaluators who verified that these were indeed typical user questions [Harman, 2011].

The searching process was repeated for all 4 types of the created indexes. The search time and the success rate of the search were recorded accordingly. The searching procedure failed an average 35% to generate the target document. No significant differences amongst the indexing systems were found. The recorded reason for the failure was "human indexing error"

which was also reported to not show significant differences between indexing techniques. This led Cleverdon to re-think the decisions regarding the content descriptors that were used for each document. He noted that exhaustive indexing (larger number of descriptors for each document) yielded a better recall however, reducing precision due to the repetition of several descriptors within documents. Exhaustive indexing can also be achieved by involving a method of showing relationships between terms. However, the selection of the descriptors themselves was still an issue. The problem of how to select content descriptors for indexing, and an increasing interest in evaluation issues, led Cleverdon to continue his investigations in the second set of experimentation that lasted from 1962 to 1966 [Harman, 2011].

Prior to starting the Cranfield 2 experiment, Cleverdon tested the indexing technique by using a sample of a smaller number of of documents (only 1000). This was done to examine whether the usage of test questions through source documents functioned properly, the results stated that they did. For that, Cleverdon sat a target of 1200 (he ended up using 1400) documents with 300 search questions (he ended up using 221) to be tested in Cranfield 2. He decided that the experimentation model should be modelled to fit realistic situations, where the selected documents would be ones that users naturally search for, the search questions must be similar to the ones usually asked, and the evaluation parameters needed to be relevant to how researchers would evaluate the resulting documents from their search process. He used the source document method for search question collecting but modified the evaluation to count also all the resulting documents relevant to a given question [Harman, 2011].

Unlike Crandfield 1, Cranfield 2 didn't only focus on a selected set of indexing systems, but rather examined in more depth the various properties of index methodologies. Cleverdon and his team wanted to build a "complete" set of index variations for each document using 33 different index type. Afterwards, they also wanted to manually test these variations through the search question experimentation [Harman, 2011].

	Relevant	**Non Relevant**	**Total**
Retrieved	A	B	$A+B$
Non Retrieved	C	D	$C+D$
Total	$A+C$	$B+D$	$A+B+C+D=3094$

Table 2.1: Document count distribution for recall and precision value calculations.

At the end of the indexing process, they ended up with 3094 unique single terms in the indexed 1400 documents. They also used exhaustive indexing by assigning weights to simple concepts based on their importance inside a document, 9-10 for the main general theme of the document, 7-8 for a major subsidiary theme, and 5-6 for a minor subsidiary theme. The weighted concepts themselves are later broken down into single terms with weights for each of the terms, again based on their importance, but this time, in the simple concepts. The later was also influenced by the indexer's view of the term's concreteness and potency. The

indexing was also repeated manually on the full document as a contrast and also to provide two additional levels of index exhaustion, namely, the title only, and the title in addition to the abstract of the document. For evaluating the results, 2 particular metrics were used, the **recall ratio**, and the **precision ratio**. These 2 were chosen based on the Cleverdon characterization for them being simple and descriptive for the user experience. Table 2.1 shows symbolic representation on how components for these metrics are denoted. Precision is calculated as *A/(A+B)* whiles the recall ratio is calculated as *A/(A+C)* [Harman, 2011] [Keen, 1967].

The results from this extensive experimentation was a ranking of a list of the used index terms. The ranking was based on the *Normalised Recall Score* All 7 of the highest ranking index methods used only single terms. The highest ranking method was found to be the one using the word forms (stems) of these single terms [Harman, 2011].

There were 2 very important contributions made by the Cranfield 2 in the field of IR. The first of which was that it has conclusively shown that using the actual terms in a document in indexing them yielded the best searching performance. This outcome was later used by the SMART project to guide them in their research. The second significant outcome was the Cranfield paradigm for evaluation. Today this is generally taken to mean the use of a static test collection of documents, questions, and relevance judgments, often with standard recall and precision metrics [Harman, 2011].

2.2.2 The SMART System

SMART (System for the Mechanical Analysis and Retrieval of Text) was started by Gerard Salton at Harvard University in 1961. He later moved the project with him to Cornell University in 1965 where he it started to take shape into its final more evolved form. Salton was a pioneer in the field on information retrieval and is famously nicknamed as the the father of that field. Salton's initial interest was in the structure used by manual indexers in the indexing of documents. However, suggestions by several researchers at the time that using words of document for indexing intrigued him. The SMART framework allowed for the insertion of several software modules to test the effects of various indexing methods on their ability to retrieve target documents. These modules included citation indexing, thesaurus coding and the use of simple words. The SMART project continued to produce a large amount of research published in journals, proceedings and in reports to National Science Foundation. Salton's input into evaluation metrics for IR systems are critical for nowadays usage of these systems [Harman, 2011].

Searching for documents using SMART framework produced a ranked list of documents. This prevented the indexers from using single point recall or precision metrics to evaluate the indexing method. SMART's developers proposed two additional sets of metrics called rank recall and log precision and normalized recall and normalized precision. These metrics measured the difference between the actual ranked positions of the relevant documents resulting from a search query and their "ideal" positions [Harman, 2011]. In [Keen, 1967]

and [Salton, 1971], the following equations for normalized recall and precision were mentioned.

$$Normalized\ recall = 1 - \frac{\sum_{i=1}^{n} r_i - \sum_{i=1}^{n} i}{n(N-n)} \tag{2.1}$$

$$Normalized\ precision = 1 - \frac{\sum_{i=1}^{n} \log r_i - \sum_{i=1}^{n} \log i}{\frac{\log N!}{(N-n)!n!}} \tag{2.2}$$

where the denotations represent,
n = number of relevant documents,
N = number of documents in collection,
r_i = rank of ith relevant document,
i = ideal rank position for the ith relevant item.

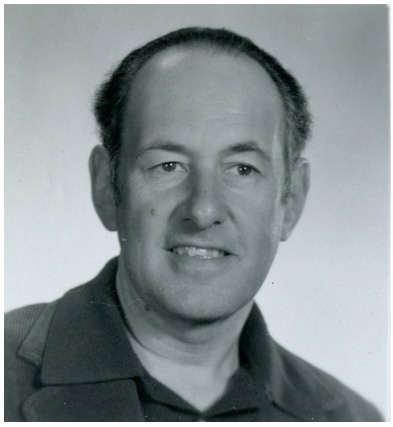


Figure 2.3: Gerard Salton, a German professor at Harvard and Cornell Universities. The developer of SMART Information Retrieval System.

As for [Keen, 1967], the author, also a contributor in the SMART project also tried to obtain separate recall and precision performances. He did so by the means of creating a recall to precision curve. Where recall is the fraction of retrieved instances to all relevant instances and precision is the fraction of retrieved instances that are relevant. These two can be compared using the above mentioned curve. Luckily, discussions regarding the IR field at the time were mostly in accordion with each other regarding the forms of metrics to be used, and the field was able to move forward without many metrics arguments [Harman, 2011].

Another important addition SMART made to the field was a large set of new test collections built according to the Cranfield paradigm. The test collections were based on query searches through a specified list of documents alongside the ideal search question that provided the source document as a result. The collection they provided also specified the reasoning for using the query and how they modified the search question to better fit the document.

This later on gave newer thoughts on how search methodologies should work, and how to index documents in a manner that would result in a specified set of outputs. Another important addition SMART made to the field was outside the metrics domain. They allowed for an easy experimentation by allowing users to perform minimal recoding to change the parameter settings in their queries [Harman, 2011].

2.2.3 Evaluation Metrics Improvements Up To 1992

The metrics developed during the Cranfield tests and the developing of the SMART system were extensively used by researchers in the IR field. One major issue that these metrics shared was their lack of an easily-interpretable single measure of performance. Due to that, two other metrics were developed who had an impact in the IR field of studies. The first of which was the expected search length, and the second was the van Rijsbergen's E measure [Harman, 2011].

The expected search length metric was developed by William Cooper [Cooper, 1970], who argued that the number of relevant document in the search that the user actually desired was an important input into the evaluation metric. So his idea of the expected search length was to measure the "cost" to the user to find their desired number of relevant documents. In the case of having the results as a set of ranked documents, the cost would be the number of the non-relevant resulting documents. This metric is used often in the cases of known item searching [Harman, 2011].

The second metric, the E-measure, is defined for set-based retrievals, like the Cranfield tests. However, it can also be extended to ranked lists by calculating recall and precision at document cutoff levels. The E measure, like the expected length metric, allowed the user to be a part of the evaluation. This was done through introducing the α value explained in the equation below [Harman, 2011].

$$E = 1 - \frac{1}{\alpha \frac{1}{P} + (1 - \alpha)\frac{1}{R}} \tag{2.3}$$

where the denotations represent,
α = variable controlling the emphasis on precision versus recall,
R = recall value,
P = precision value.

2.3 TREC and Batch Evaluation

The Text Retrieval Conference is a series of workshops co sponsored by the National Institute of Standards and Technology (NIST) and the Intelligence Advanced Research Projects Activity focusing on the research development and evaluation of different information retrieval (IR) methodologies. This series of workshops began in 1992 as a part of a bigger program (TIPSTER Text) that focused on improving the information retrieval research field for

the usage of governmental agencies. TREC's main purpose was to provide a fundamental pathway and the infrastructure required for large-scale evaluation of text-retrieval methods. The approach set by NIST was to provide each of the participating groups with a set of data and test problems for evaluation purposes. NIST also decided on a uniform scoring technique to fairly evaluate the approaches taken by the participating groups. The workshop provided for the groups took place after the evaluation of the results to get the participants together for sharing their thoughts and methods as well as to set track for future research work.

2.3.1 The TREC Ad Hoc Tests

The ad hoc system refers to the search process against a list of a given documents in the hopes of obtaining a fixed document collection. This was heavily impacted by the Cranfield paradigm and the known target document search methodology, which is still used today for evaluating retrieval system. TREC focused on this track of research from 1992-1999. The data collection used in the search process was created based on realistic user model, that is, documents to be used by individuals with heavy search based jobs. Examples of those are journalists, legal workers, and scientific researchers. The data collection was large set of varying text lengths, writing styles, variations in vocabulary, and had to be of different topic fields. However, they all shared the attribute of being full-text documents. The following table lists the sources from which the documents were obtained during the the first 8 years of the ad hoc based testing [Manning et al., 2008] [Harman, 2011].

Due to the availability of multiple fields within the provided documents, a wide range of query construction methods was applicable. Also, test collections during the early years were followed with sentence-length requests for document retrieval, leading to consistent judgment criteria for the effectiveness of the retrieval process. All topics were designed to mimic a real user's need, although the actual topic writers, the topic format and the method of construction evolved over time. By the third TREC, the number of fields within the search sentences underwent some changes. The most important of which was the addition of a narrative section to address fully how to separate a relevant document from a non-relevant document. The assessment was based on whether or not the retrieved document would be used in some manner in writing a report about the subject matter in the search question. This led to a binary relevance judgment, the retrieved document is either relevant or not. For each search question/topic, single assessor was assigned to determine the relevance of a document. This was done to maintain a single user interpretation of the topic in hand. Since emphasis has been set on having real-user topics for search, no attempts were made to build topics to match any particular characteristics [Harman, 2011].

The problems arising from indexing the documents were mostly related to scaling issues due to the heavy expenses of obtaining enough storage for all the required indexes. The largest number of relevant documents were from the Wall Street Journal and Associated Press as they covered the larger domain of fields with respect to the other sources. Length-

Source	# of Documents	Mean # of Words
Wall Street Journal	173,252	466.0
Associated Press	242,918	473.6
Computer Selects	293,121	371.3
Federal Register	101,450	960.0
Abstracts from Department of Energy	226,087	120.4
San Jose Mercury News	90,257	453.0
U.S. Patents	6,711	5391.0
Financial Times	210,158	412.7
Congressional Records	27,922	1373.5
Foreign Broadcast Information Services	130,471	543.6
Los Angeles Times	131,896	526.5

Table 2.2: The document sources for the first 8 years of the ad-hoc testing

ier documents showed difficulties in screening at earlier TREC events, however by the third TREC, the term weighting algorithms were improved and the difficulties seemed to disappear [Harman, 2011].

A measure called topic *hardness* was established for each search topic. This is given as the average over a set of run of the precision at R. R being the number of relevant documents for that topic. The hardness was that, OR, he precision at 100 if there were more than 100 relevant documents. The hardness measure is oriented towards obtaining a high recall and measure of the ability of systems to retrieve a higher number of relevant documents per search topic. It is important to note that there is a huge variance in the hardness of the systems developed by the participating groups when testing various search topics. However it is critical that the average performance measure truly reflects the differences rather than just random performance points [Harman, 2011].

To guarantee consistency in the relevance judgment, 2 additional judgments were carried out by 2 different assessors for each search topic. This yielded 72% agreement in the resulting relevance. Unfortunately, most of this was due to the large number of irrelevant documents. Around 30% of the initial relevant documents as set by the first assessor, were marked as irrelevant by both other assessors. This average shows a large variability across topics, where 12 of the 50 (on average) topics for each of the 3 judges had more than 50 % disagreement ratio on relevant documents. This can be due to human error or human variation in judgment to what is relevant and what is not [Harman, 2011].

TREC Ad Hoc Metrics

TREC followed the evaluation metrics set by Cleverdon in the Cranfield tests and the metrics developed by Keen and Salton in the SMART project. Researchers in the field at this time were using various metrics from SMART, but with different implementations and with dif-

ferent choices of which metrics to report. The availability of a single implementation however, provided ease in comparison across systems. The biggest addition is the R-precision value for better measurement of the high-recall tasks.

$$R\,Precision = \frac{r}{R} \tag{2.4}$$

R = total number of relevant documents,
r = number of retrieved documents that are relevant in the first R list of ranked documents.

Text Retrieval from OCR Outputs

Part of TREC agenda was to test the performance of IR systems when a noisy text was used. An example of such text is the one produced by an Optical Character Recognition (OCR) system. In TREC-4 for instance, ad-hoc designed searched topics were tested for retrieval within WSJ texts that has been artificially degraded to 10 and 20% character errors. In TREC-5, actual OCR collected data from the 1994 Federal Register was used alongside degraded scanned copies with 5 and 10% character error rates. Each of the search topic was created with the intentions of retrieving one target document. Nowadays, OCRs are very precise and accurate and there is no need to design ad-hoc experiment specifically for them rather than ones for an entire collection of documents. The metric used to measure the performance of the TREC systems was a modified version of the *expected search length*, and is based on the rank of the target document in the list of the retrieved documents. Averaging of the task with respect to the searched topics was carried out using the *mean-reciprocal-rank* measure. This is the mean of the reciprocal of the rank where the target document was found calculated as an average across all the topics searched through in a system [Harman, 2011].

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i} \tag{2.5}$$

N = number of search topics,
$rank_i$ = the rank of the target document in the list of results for the search topic i.

Enterprise Searching in TREC

In 2005, TREC changed its track focusing on web based retrieval to enterprise track that was more focused in intranet searching. The user model in this case is someone who is trying to retrieve in-organization information. The dataset used in searching was a crawl of the W3C site that included emails, discussions, web pages, and text documents in various formats. The search topics were based on three tasks; target email message, ad-hoc search for emails discussing a specific topic, and an internal search for experts in specific fields. In 2007 and 2008 TREC, the data and the topics came from within one organization (Australian Commonwealth Scientific and Industrial Research Organization) to improve the authenticity of the

performance measure of the built IR systems. The task for the track in those 2 years was to create a new "overview" page for a specific topic. The page is to include key links from the website of the organization along with key experts within the organization itself. 50 search topics were created by the organization's staff and the track participants performed the relevance judging themselves on the outputs of the developed IR systems [Kruschwitz & Hull, 2017] [Harman, 2011].

2.3.2 TREC 2017 & 2018 Tracks

However, nowadays, several new tracks have been added and identified for future research needs. The following is a list of some research track goals sat for TREC 2017 and 2018:

1. CENTRE Track: Developing and optimizing a reproducibility evaluation test for IR. The main purpose is to reproduce best results possible for the interesting systems shown in previous CLEF/NTCIR/TREC by using standard open source IR systems and to contribute back to the community the additional components and resources developed to reproduce these results [Zhang & Huang, 2019].

2. Complex Answer Retrieval: Providing longer answer for complex information needs. The goal set for this track shows resemblance with Wikipedia pages and how they synthesize knowledge that is globally distributed. The groups participating in this track stated that they envision systems that collect relevant information from an entire corpus, creating synthetically structured documents by collating retrieved results [Dietz et al., 2017].

3. Common Core Track: Designing an ad hoc search methodology for IR from a set of news-related document. This track also coincides with another track sponsored by The Washington Post that aims to provide test collection sets for supporting the search queries made by news readers and news writers [Bondarenko et al., 2018].

4. Incident Streams Track: The goal here is to to automate the processing of social media video streams during emergency and life threatening situations for the purpose of categorizing information. The categorized and filtered data would help the aid and emergency providers in delivering the required help [Choi et al., 2018].

5. TREC Precision Medicine / Clinical Decision Support Track: The larger part of the work made in this track focuses on the development of new treatments based on an individual's genetic, environmental, and lifestyle profile. The IR relation here is to link the metabolic behaviour and the oncological data of clinical test subjects' with their test trials. The result is a data-driven approach investigating the best treatment for an individual patient [López-García et al., 2017].

6. Real-Time Summarization Track: The goal of this track to develop techniques that can be used in real-time analysis of social media based data streams. The analysis can be used to summarize activity to give a larger overview on tending subjects [Lin et al., 2017].

2.4 Related Works

In this section, the dissertation discusses the effects of deploying a DMS in an enterprise or an enterprise-like institution. For that, reference to existing literature was made, mostly from surveys, case studies, and project proposals.

In [Ahmad et al., 2017], the authors conducted a survey in which they investigated the existing electronic and paper based DMSs in a sample of small size contracting companies in Jordan. They carried out interviews with officials and representatives of these companies to get a better overview on how a DMS helped to improve the efficiency of construction projects and how they are managed. One of the conclusions of the study was that the lack of an electronic DMS cause difficulties in storing, categorizing and retrieving required information or documents. The need for storage space is also a major issue in addition to the potential loss of important data that has been published internally in the last 5 years. The authors also discussed the reasoning behind not implementing an electronic DMS; the need a major investment of time, effort and money, while benefits may need time to be noticed, was the one that shone the most. Similarly, in [Sprague, 1995], discussions regarding possible application areas of DMS are made from the point of view of researchers in 1995. The author was successful in predicting the challenges to build a compatible technical infrastructure to support document management from the points of view of institutions/individuals who are used to the culture of having a paper based file system.

Joseph et. al [Joseph et al., 2013] discussed the process by which users search for information from EDMS to accomplish their work tasks. This process is also applicable in enterprise search as one of the main purpose of implementing DMS in DAP is to increase the efficiency in retrieving target documents. The authors also developed a model for the search procedure carried out by users in which they separated the search process into seven stages.

1. Starting the search process
2. Formulating the strategy for the search and where to search for the target documents
3. Executing the search
4. Process and evaluate the results from the search
5. Acessing the search results

6. making a descion regarding the search results

7. End of search

Further elaboration of these steps is shown in the figure below that was captured from the paper itself. it is important to note that this search pattern is viable in records management domain and may differ from the search when it is library-based or Web-based.

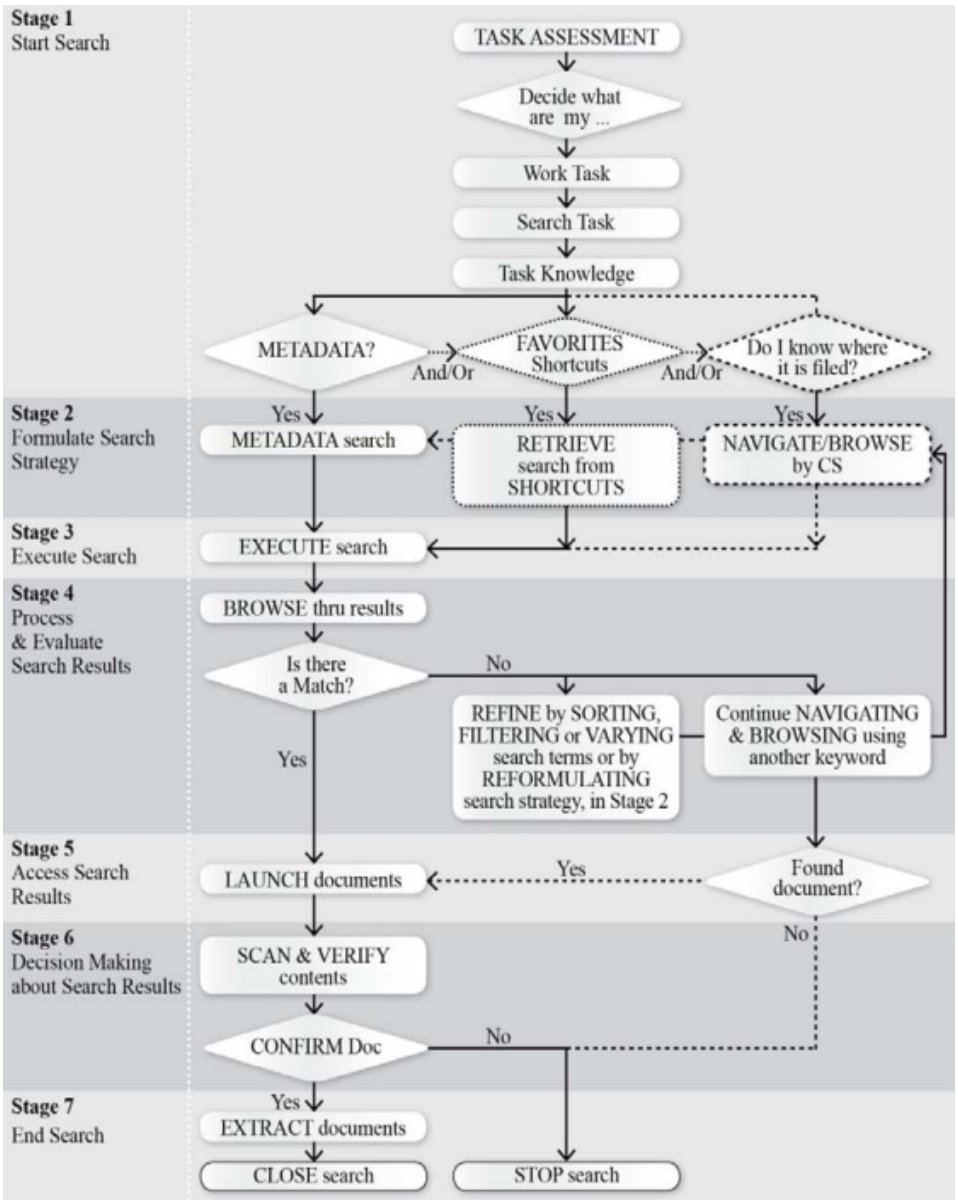


Figure 2.4: Search behaviour model for EDMS.

As in [Ahmad et al., 2017], in [Alshibly et al., 2016], a study in Jordan was conducted. This time however it was regarding the critical success factors for implementing EDMS in governments. This is relevant because a government architecture when it comes to document storage and search capabilities match that of a large enterprise.

Factor groupings	CSFs
Technological readiness	1 Architecture readiness
	2 Infrastructure readiness
	3 Process readiness
Top management support	4 Top management, leadership, and commitment toward EDMS
	5 Top management encouragement toward utilization of EDMS
	6 Clear mission developed regarding business objectives
	7 Top management encouragement toward formal/informal communication
	8 Gaining commitment and support of chief executive officers
	9 Planning and project management
Training and involvement	10 Providing the employees with adequate information of EDMS-related principles through training
	11 Adequate training and support for users
	12 Employees are trained on EDMS job-specific skills
	13 Involving EDMS end users
	14 Management always updating their knowledge
	15 Involving all levels within the organization and external stakeholders
	16 Actively encourage employee participation in EDMS-related decisions
Resource availability	17 Prior existence/development of necessary infrastructures
	18 Sufficient financial resources provided to support EDMS implementation
	19 Human resource availability
	20 Technical resources (e.g., software, equipment) are provided
	21 Requirement-driven procurement planning
System-related factors	22 EDMS functionality
	23 Effectiveness of EDMS
	24 Efficiency of EDMS
	25 User friendliness of EDMS
	26 Usability and understandability of output
	27 Integrating systems and technology
	28 Demonstrating benefits
	29 Piloting and testing
Work environment and culture	30 Policies and guidelines
	31 Communication
	32 Aligning projects with business objectives
	33 Ensuring a project has a clear agenda
	34 Change management
	35 Sharing expertise
	36 A spirit of cooperation and teamwork
	37 Supporting team-based approaches to problem solving

Figure 2.5: Critical Success Factors for the implementation of EDMS.

In this article, Alshibly et al, listed 37 factors that were considered as prerequisites of successful electronic document management system implementation. The factors were separated into 6 groups based on functionalities and are shown in the figure above. These were studied, filtered, and added to, to create the list of functional requirements for choosing the DMS for DAP. As shown in points 4,5, and 7, it is important that the top management is actively participating in the implementation of the DMS in the enterprise. This was also

recommended in [ElShobaki, 2017], where ElShobaki argued that an increase in interest by upper management ensures the success of the electronic document management system.

In [Dizon et al., 2017], the authors were tasked with building a DMS for the Faculty of Medicine at the University of Santo Tomas in the Philippines. The researchers used an agile approach in the software development to accommodate this particular client's need. The primary focus of the study is to develop a system that will minimize paper usage and quickly generate reports that can be used for critical decision making. From this article, I can understand how they reported the client's requirement into process mapping to determine what DMS functionality is needed. Not all the functional requirements are reported by the stakeholders, some are derivatives of other requirements and mapping the process by which the user utilizes the system is important to determine those missing functional requirements.

In [Ugale et al., 2017], the authors address some of the technologies that are helping professionals shift toward a paperless business world. The most important of which is OCR. This paper describes different steps for processing different documents using scanning, tagging, and indexing for effective data retrieval with OCR and Indexing techniques. The authors tested several OCR software to determine which had the best functionalities with respect to their requirements. From this reference, I used their evaluation methodologies for the available free software in deciding which DMS I would recommend to DAP at the end of my testing.

An important part of this dissertation is focused on the evaluation techniques as well. This ranges from metrics with performance points to human-factor related material such as ease of use and simplicity of the presented software options. In [Manning et al., 2008] Manning et al, discussed many attributes regarding IR. One of which was the evaluation methods. The book recommended various measurements and their relevant formulas, many of them are used in my dissertation. What I found particularly useful in this book was that it showed examples on how to use these formulas. The book also discussed the concepts behind relevance and how to consider if a search query result in the expected results. Many documents share the same feature with a target document aimed from a search procedure, thus relevance is an objective concept from the user point of view with respect to a computer following commands.

2.5 Full-text Search Frameworks

There are several frameworks used in nowadays local search engines. In this section, we will highlight two of the most common ones (Elasticsearch and Solr) which can be used in full-text searching and data parsing of documents. We will also briefly explain the process by which those frameworks function. Each of the deployed web servers tested in this project used either of those frameworks or one of the lesser famous ones mentioned at the end of this section.

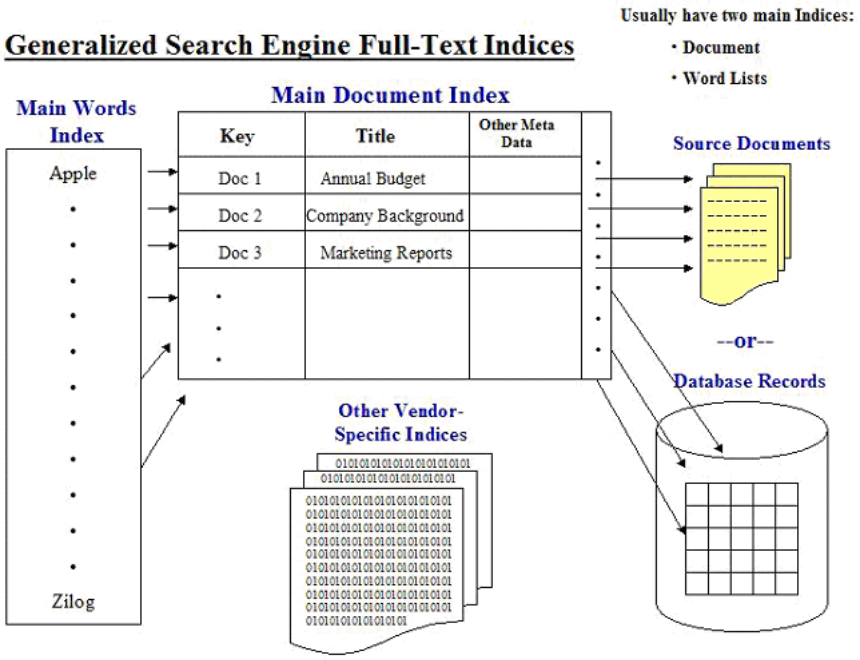


Figure 2.6: A generalized architecture for a full-text search engine.

Full-text index is a type of indexing technique that allows the processing of full-text queries. In a full-text search, a search engine examines all of the words in every stored document as it tries to match search criteria (for example, text specified by a user). To apply that, a special querying method is used where search component is matched against a given set of characters or binary column data gained from the documents stored inside the database. Full text indexing can be thought of as a description of an arbitrary input by textual words. Identity function can be used when the input consists of simple words [Misutka & Galambos, 2008]. This special index operates by breaking the indexed columns into *tokens* which in turn is considered to be the data index [Snaidero, 2018]. The indexer will make an entry in the index for each term or word found in a document, and possibly note its relative position within the document. Usually the indexer will ignore stop words (such as *the* and *and*) that are both common and insufficiently meaningful to be useful in searching. These words when it comes to TREC based research for example, are provided as an addition to increase the performance of the search engine. Some indexers also employ language-specific stemming on the words being indexed. For example, the words *eat*, *ate*, and *eaten* will be recorded in the index under the single concept word *eat*.

Building up the index starts by creating a *FULL TEXT CATALOG* where the full-text index data is stored. The catalog can hold many indexes but each index has to be a part

of one catalog. The updating procedure of a full-text index is time consuming due to the intensive resource consumption required for building. A table in the catalog can only contain one index, thus if more than one column is needed, new tables are needed for hosting them. There are several types of full-text based indexes, in TREC based research for example, XML indexes are common as they are specifically built to handle indexing XML type columns [Snaidero, 2018].

Elasticsearch

Elasticsearch is an open-source (under Apache 2 license) search engine developed by Elastic NV. The engine is built on top of Apache Lucene, arguably the most advanced search engine library available for developers nowadays. Lucene is integrated into a Java application and is used for indexing and searching. What makes Elasticsearch unique and powerful is that it indexes every field inside the input document making the entire document searchable. Depending on the applications that use Elasticsearch, the communication port is interchangeable, however, the application API is always reachable through port 9200 regardless of the application language.

Elasticsearch is document oriented, meaning that it stores entire objects or documents. Elastic search use an approach of document indexing, searching, sorting, and filtering rather than applying the process to rows of columnar data, this allows it to perform complex full-text search. Elasticsearch uses JavaScript Object Notation (JSON) as the serialization format for documents. Due to JSON's simplicity and ease of read for the user, converting an object to JSON for indexing was the selected methodology in implementation.

It is important to note that in Elasticsearch, a document is set t have a type which in turn is stored in an index. A cluster is a group of one or more instances (nodes) running concurrently in Elastic search. Each cluster can contain multiple indexes, these may contain many types which hold documents in rows. Each of the held documents can have many fields (columns).

Elasticsearch ⇒ **Indices** ⇒ **Types** ⇒ **Documents** ⇒ **Fields**

In full-text search, [Gormley & Tong, 2015] suggest that for the sake of simplicity Elasticsearch stores the document's text in one or more of the fields (metadata) of a stored input (document). The search query is matched against any of the fields inside the stored documents. If the search query consists of more than one word, the engine will try to find a document containing both words preferably in the same order by which they were entered inside the query. If not found, each word is searched for separately. At the end, the engine will output a list of matched documents alongside a score which is used to rank the outputs based on predicted relevance.

A simple IR request sent to Elasticsearch is in essence a simple HTTP based request.

```
curl -X <VERB> ' <PROTOCOL>://<HOST>/<PATH>?<QUERY_STRING> ' -d ' <BODY> '
```

- VERB: HTTP method defining the required action, such as, GET, POST, PUT, HEAD, or DELETE

- PROTOCOL: Define the protocol, either http or https
- HOST: The hostname of the node in a cluster. This can be defined as *localhost* or *127.0.0.1* for request meant for nodes inside the local machine
- PORT: The port running the Elasticsearch service based on the selected protocol. By default, the http protocol is ran at port 9200.
- QUERY_STRING: An optional query string parameter.
- BODY: A JSON-encoded request body

Apache Solr

Solr is an open source enterprise search server. Many of the famous public site out there use solr for their search needs, such as CNET and Netflix. Alike Elasticsearch, it is also written in Java, is built on top of Apache Lucene, communicates via HTTP/HTTPS, and uses JSON to communicate with applications written in other languages. Solr has many useful features such as query spell correction, query completion, and a *more like this* feature for finding similar documents. In Lucene, tokens are transformed by eliminating word stems and substituting synonyms. The resulting processed tokens are called **terms** [Shahi, 2015].

Solr has the ability to store several caches for faster search responses. Solr also has additional contrib modules that make it special compared to other engines, **DataImportHandler** (DIH) data crawler for extending its import capability, and **Solr Cell** an adapter for extracting text from numerous file types [Shahi, 2015].

Just like what you would expect from an index based search engine, Solr functions by processing textual data. The firsts of 3 main concepts in this processing are the field analyzers. These are used during the the document ingestion, indexing, and in the querying duration. An analyzer examines the text of fields and generates a token stream. Analyzers may be a single class or they may be composed of a series of tokenizer and filter classes. This brings us secondly to tokenizers whose function is to break field data into tokens. Lastly, the built in filters examine the token stream to determine what to keep, transform, or discard. This is done by matching the wording to the built it in stopwords and dictionary of similar terms and verb tenses. Tokenizers and filters may be combined to form pipelines, or chains, where the output of one is input to the next. Such a sequence of tokenizers and filters is called an analyzer and the resulting output of an analyzer is used to match query results or build indexes [Apache-Software-Foundation, 2017].

Search queries in Solr are processed through a **request handler**. This is a plug-in used for defining the logic behind the input query. The search query processing will then be passed to a query parser that in turn interprets the input terms. There are different parsers available in

Solr, each correspond to different syntax as judged by the request handler. The query parser input may include search *strings* (search terms), *fine-tuning parameters*, and/or *presentation controlling paramters*. The figure below highlights the most important steps in the search queries as followed by Solr [Apache-Software-Foundation, 2017].

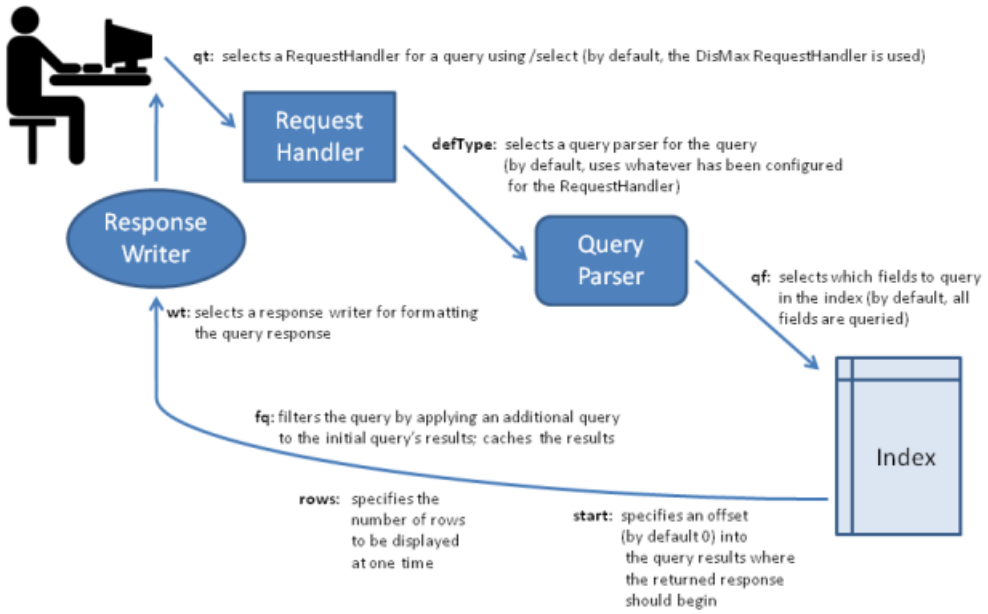


Figure 2.7: Summary of key elements of the search process followed by Apache Solr [Apache-Software-Foundation, 2017].

Another great feature of Solr is its capability to detect the language of a document. This is done to ease the process of indexation and searching as stopwords for example differ from one language to another [Apache-Software-Foundation, 2017].

Zend Framework

The Zend Framework is a web-application framework written in PHP-7. It is an open-source and built to be an object-oriented application. The framework is built on the grounds of having a professional collection og PHP-based packages worked coherently together. The coherence is achieved by the usage of Composer to be a part of the package dependency managers. Using the Zend Framework, Zend_Search_Lucene was built. Much Like ES and Apache Solr, ZSL is a derivative of the Apache Lucene project libraries. ZSL is a text search engine programmed entirely in PHP5. It has the capability to add search functionalities to almost any PHP-driven application. This is due to it storing its index on the filesystem rather than in a database server. ZSL supports the following features [Zend, 2016]:

- Ranked search, this where the best results from a search query are returned first in a list.
- Many powerful query types: phrase queries, boolean queries, wildcard queries, proximity queries, range queries and many others.
- Search by specific field and metadata. (e.g., title, author, contents)

OpenSearchServer

This is a bit different because it is a fully implemented software. It is a server based system that allows for testing and development of index based applications. What makes it unique is that it offers a series of full text lexical analyzers.

3

Experimentation and Methodology

In this chapter we will discuss the list of functional requirements and system objectives. We will also discuss the list of the tested opensource DMSs and the chosen ones for the testing phase. In choosing the final candidates, the functional requirements were taken into consideration. Not all of the initial functional requirements were satisfied by the chosen candidates. In facts, even the majority of the paid DMSs don't satisfy all of the requirements. We will also discuss the installation methodology and how to deploy the webservers for testing purposes. Secondly, the chapter will also discuss the list of documents uploaded to the DMSs and the initial fulltext search tests performed in them. This will be followed by the list of query topics used in the known-item search testing alongside with the metrics to be used in classification of how well they perform in information retrieval. The latter will be discussed in further details in the results chapter.

3.1 System Objective

Managing documents at enterprise level can be very complicated and chaotic due to having many document types and formats. Also, the need to have a unified system for document storage at all the departments of an enterprise would add complications regarding document versioning and naming. As discussed in chapter 2, DMS tends to help in solving such issues and would reduce the complications resulting in electronic storage of documents. A typical DMS embedded in an enterprise would be able to store, version, and organize documents submitted by users and administrators. It would help in maintaining billing information, inventory, workflow in production plants, employee information, and much more.

The system has to satisfy the stakeholders' needs while keeping in mind the constraints of such process. The stakeholders have a set of requirements that the system should be capable of providing. One important requirement is the ability to backup the data on remote locations to guarantee their storage in case of DMS malfunction. Luckily, many of the tested

opensource software, readily store the document inside the document server which can be located internally inside the enterprise's premises. Another important feature requested by the stakeholders is the ability to have a full-text search. Opensourced DMS provide full data indexing and thus that requirement can be fulfilled by many of the pre-existing systems.

The main objectives of the system is to help department managers and employees in storing their documents while keeping search procedures easy and efficient by applying data indexing and enabling full text search. The system should be accompanied with the ability to perform Optical Character Recognition (OCR) to allow the electronic storage of scanned documents for performing full text search on them as well. Other DMS functionalities are also explained in detail in the functional requirement section in this chapter.

3.1.1 System Stakeholders

The stakeholders are defined as the individuals or the administration bodies who requested the implementation of the DMS. In this case, they are the heads of the departments and the employees at De Angeli Prodotti, a manufacturer of insulated and enamelled wires for electromechanical applications and overhead power line conductors. The system admins and users are also defined as stakeholders and they would be the ones responsible of maintaining the system after deployment. The stakeholders are divided into several classes explained in the following section.

User Roles and Privileges

The tested opensource software allowed for role creation and privilege editing. This is vital as the set of functions assigned to each user role may vary from one enterprise to another. The privilege ladder follows the inheritance concept, that is, each user can perform procedures privileged to his/her user role as well as the ones assigned for the roles lesser than him/her in the privilege ladder. The user roles are briefly described based on a generic need of an enterprise. In this application, mainly 4 user roles are selected.

1. Administrators: The ones responsible in maintaining the system, scheduling and performing data backups, assigning user roles and privileges, creating user accounts and doing password retrievals in case of a forgotten/lost password, monitoring the DMS performance, troubleshooting and problem solving for other users, and security control.

   System admins are also the ones who test any new feature implemented in the DMS prior to releasing it on the DMS running in the main webserver. They are the ones receiving feedback on the system and other inputs from the users such as recommendations and complaints.

2. Heads of Departments: Users who are capable of monitoring workflows, document versioning, and the operation procedures regarding document creation for all employ-

ees and users under their respectful department. These users keep track of documents belonging to classes/tags assigned to their departments. They can also monitor user logins and have control on user privileges for employees working in their departments. Head of departments can also lock documents to disallow editing and can also set file locations where submission review is necessary before a document is published on the DMS. They can also assign tasks regarding the upload, edit, and creation of documents for team members under their command.

3. Users/employees: Regular employees follow the command of one or more heads of department and have their privileges set by the head of the department or the system admin. Employees are the ones responsible for document upload, versioning and editing, and new document creation. Employees report directly to their chiefs or admins and can see document submissions of other employees at the same department if the submitted document is set on public status. Employees can check in/out documents for review and edit and can also lock their own documents to disallow editing/deletion/download/preview of other users other than the admins and the head of the respectful department.

4. Guest users: Type of users with very limited privileges to download/preview certain types of documents meant for review. These users can be individuals invited by system admins/heads of departments/regular employees to join the DMS webserver in order to preview set of documents. These users will have accounts with expiration capabilities, that is, their account privileges are temporary. This is done to reduce the security risk of exposing the enterprise's webserver to non-employees.

3.1.2 Functional Requirements

The following is a list of the functional requirements by which the DMS was selected. The list of tested DMS had many of the following requirements satisfied in the available features, but only few of them had all the required functions already built in the application.

1. Data Indexing to enable full text search inside the stored documents in the DMS.

2. The ability to upload documents from different sources and users.

    - The ability to have a webserver accessible via internet/intranet with user authentication.
    - The ability to submit documents into the DMS via mobile devices such as smartphones.

3. Italian language support in the Graphical User Interface (GUI) of the DMS.
4. Document versioning.
    - The capability to store multiple versions of the same document.
    - The capability to revert back to previous versions of the same document stored in the DMS.
    - The capability to download all the versions of the document as stored in the DMS.
    - The capability to check in/out a document or any of the its versions.
5. Embedding the names of the document uploaders and authors for the purpose of verification and document tracking.
6. The ability to create rules on user roles and privileges.
7. The ability to approve/deny document submissions by users with certain roles.
8. The ability of privileged users to post comments on stored/submitted documents.
9. The DMS should support multiple document file extensions alongside the capability for data indexing for each extension. (PDF, doc, docx, csv, xls ... etc).
10. The ability to set defined metadata fields to require matching to a document type to fit a structural document requirement.
11. Bulk import of pre-existing documents stored in data servers alongside their folder directory in the original electronic storage location.
12. The ability to set dynamic values for metadata to ease search procedures inside documents.
13. The ability to preview several document types and extension formats as stored in the DMS without the need to download into local machines.
14. OCR capabilities to enable data indexing of scanned documents that are not electronically editable.
15. The OCR should have multilingual support.
16. Document grouping capabilities to enable automatic or manual linking of certain documents based on set criteria or properties.

17. The ability to tag documents based on set properties or document type.

18. Document workflow for ease of document state tracking and role assigning in the creation of new documents.

19. The chosen DMS should have pre-existing user manuals and developer documentation for ease of troubleshooting and bug detection.

20. The ability to perform data backups to ensure long term storage of documents.

21. The DMS should also be atomic and provide the ability to isolate departments from each other when needed to maintain secrecy and classification of the stored documents.

22. The ability to migrate the database on which the documents are stored in case of a need to change the DMS or upgrade the system.

Note: this list can be edited in the future if new functionalities are to be required by the stakeholders of the DMS.

3.1.3 Nonfunctional Requirements

These are sets of requirements that are expected from any DMS and are not related on the functional procedures of the system itself but rather other components exterior to the DMS.

1. Preservation of data integrity for the stored information including uploaded documents, system logs, and user settings.

2. Ease of usability to allow non-experts to use the system.

3. The DMS should be efficient and consistent to allow the handling large amount of data with quick response times. For that, the requirement will also be dependent on the specifications of the host machine from which the webserver is deployed.

4. The system should have high tolerance for failure due to excessive load to minimize data loss in cases of high traffic.

5. The system should have capability of staying online as much as needed as it might be deployed on the web.

6. The capability to add 3rd party plugins for ease of functional expansion.

7. The deployed DMS should be highly secure to prevent unauthorised access to sensitive enterprise data and to allow the managers and staff to log-in and access the allowed services depending on their authorities.

8. The DMS should be a part of an un-dead project with long term support to ensure constant security upgrades and the addition of new functionalities.

Constraints

The constraints by which the selected system must comply with:

1. Should use MySQL, Postgresql, or SQLite databases.
2. The server side's operating system has to be a Linux based distribution (preferably Ubuntu/Debian based).

3.2 List of Tested DMSs

The following are the list of tested webservers. Next to each entry, you will find the reasoning behind its neglect. I ended up with a good selection of DMSs from which I selected 5 to run the tests.

Deployed Webservers

1. SeedDMS
2. LogicalDOC Community Edition
3. OpenKM Community Edition
4. Kimios
5. Mayan EDMS
6. OnlyOffice
7. Teedy
8. Krystal DMS
9. NextCloud with ElasticSearch Plugin
10. Redmine with DMSF Plugin
11. Alfresco Community Edition
12. Nuxeo (Content Management System)

Rejected Webservers

1. Sentrifugo – Only for HR purposes, no viable DMS
2. OpenDOCman – Limited functionalities for free version
3. Maarch – Dead project
4. EpiWare – Dead project
5. LetoDMS – Project transferred to SeedDMS
6. Bitfarm-archiv – No Italian language support & limited functionalities
7. DocMGR – Out of support
8. NemakiWare – No Italian language support & out of support
9. OrfeoGPL – No Italian language support & limited functionalities
10. Openprodoc – No Italian language support & out of support
11. Projeqtor – No indexing
12. SuiteCRM – Low performing file indexing in initial testing
13. Paperless – Low performing file indexing in initial testing
14. Muk – Low performing/Lack of proper documentation /No Italian language support
15. Phase – Low performing/Lack of proper documentation /No Italian language support
16. KloudSpeaker – Low performing/Lack of proper documentation /No Italian language support
17. Kala – Low performing/Lack of proper documentation /No Italian language support
18. AdLibre – Low performing/Lack of proper documentation /No Italian language support
19. Plone (Content Management System) – Undocumented error in deployment
20. Casebox – Undocumented error in deployment
21. Xinco – Undocumented error in deployment

22. dotProject – Low performing/Lack of proper documentation/mainly focused on project management.

From the list of deployed webservers, 5 where chosen to continue testing based on their conformity with the larger number of functional requirements. These FIVE are SeedDMS, MayanEDMS, OpenKM CE, Alfresco CE, and NextCloud alongside with the ES plugin. With those 5 DMSs, opensearcserver was used as a form of a control test due to its reputation as a professional search engine. Also, traditional search methods such as system file search was used for comparison.

3.3 DMS Functionality Testing

In this section, we will discuss the list of documents used to test the systems and briefly go over how they were imported. We will also discuss the search queries used and the evaluation metrics for the testing procedure. Chapter 4 will contain the results from these search queries as well as some statistical analysis that would help in characterising the significance between them.

The document repository used in testing was a document dump of 5000 documents. These include the following types of files: pdf, xls, xlsx, ppt, pptx, doc, docx, pub, html, txt, jpg, png, and eml. The dump was collected from the data backups of the quality control department of DAP. 50 files were chosen at random from the document dump. These files were carefully analyzed to form a search question that if asked, will result in that target document as well as other relevant document to the search. Basically, the test performed was the known-item test. The target documents were kept in their original directories to emulate a real search experience that can be done by any of the enterprise employees. Part of the testing phase included measuring the time required to import the document repository as well as the time required to find the target document when the search is initiated.

3.3.1 Search Queries

50 search questions (search topics) that would normally emulate the searching procedure carried out by users were selected. The following questions were translated from Italian to English for the purpose of display. The target documents were kept classified to prevent any violations of privacy. Also, some client names and product types were not displayed for the same reason. Some words were added to validate a search questions, however, the search topic is unchanged. Due to the presence of stop-words, this didn't affect the results nor the search query.

Query No.	Search Query	No. of Relevant Documents
1	Pareto analysis for May of 2013	3
2	Claim by Client-X for wire type Y	5
3	The corrective actions taken to make products more easily traceable	9
4	Commercial business planning for Client-X in 2011	2
5	Company's values and vision for 2017	3
6	Environmental risk factors for fire hazards	6
7	Quality management system certificate ISO 9001:2015	4
8	Environmental management system certificate for 2009	6
9	2012 sampling and feasibility summary	3
10	Enamelling department user manual	2
11	The CTC samples for Client-X in 2011	3
12	2015 clients for aluminum cable products	3
13	Laboratory trials for Client-X in 2006	2
14	Declaration of conformity for Client-X	4
15	Declaration of conformity for Client-Y (OCR required)	5
16	Suppliers from Country-X	11
17	Fiberglass Utilization market in 2011 for Client-X	7
18	Factsheet for the toner used in Printer-X	2
19	Internal non-conformity trends in 2017	4
20	Quality management system certificate for Client-X (OCR required)	1
21	The 2015 qualitative isolated conductors report	3
22	Isolated conductor copper plates in 2015	3
23	How the hardening rollers work	4
24	PFC product description	3
25	Calibration temperature for the thickeners	2
26	Flat furnace coil discharge process	2
27	Instructions for using the wire rod of the copper trolley	3
28	Production organizers in 2017	2
29	First aid for a state of shock	4
30	Enamelling department failure control plan	3

Query No.	Search Query	No. of Relevant Documents
31	Internal audit plan in 2018	3
32	Control process for server failure emergency	2
33	Quality audit process procedure	3
34	The manual for environmental safety	4
35	The enamelled copper analysis September 2017 report	2
36	Available equipment in the laboratory	2
37	Process sheet for Chimney-X	2
38	The process reports for Item-X	2
39	Process audit for the chimney valve	2
40	Maximum allowed value for surface oxide	3
41	The compacted oxide layer glaze technical specification	5
42	Order process for Client-X	5
43	The statistical indicators for Client-X in 2017	2
44	Enamelling process mechanical characteristics sheet	4
45	Characteristics of enamelled aluminium	5
46	Security form for Client-X	1
47	Defective material list of 2012	8
48	Reasoning for monitoring the EMS	2
49	The discarded product reels	1
50	The customer focus action plan	6

Table 3.1: List of search queries and the number of available relevant documents per query.

3.3.2 Evaluation Metrics

Many of the search queries produced no results or didn't produce the target document as a result. For these cases, the rank for the output of the DMS was set to infinity, treating them as non retrieved by the DMS. I chose to search for the retrieved documents within the first 15 output results appearing in any ranking of results from the search query. In the result tables shown in chapter 4, these instances are referred to as *NF* or *not found*. This step was needed opposing to leaving the ranking to blank, as it would have lower the average retrieval rank and would result in biased results. The number of relevant documents was collected by analyzing the first 15 outputs (if any) of a search query and analyzing those documents to find the number of relevant documents. The maximum number of those relevant documents was the number released by the DMS which produced the most amount of relevant documents in the first 15 results. The MRR value (equation 2.5) and the average precision were used as metrics to compare the results from the each DMS. The MRR is the mean of the sum of the reciprocal ranks of the retrieved documents and the mean number of retrieved documents is the average percentage of the number of relevant documents to the target documents (including the target document) retrieved in the first 15 results.

$$Average\ Number\ Retrieved\ Documents = \frac{1}{N}\sum_{i=1}^{N}\frac{r_i}{R_i} \tag{3.1}$$

Where N is the total number of queries, r_i is the number of relevant documents in the first 15 results, and R_i being the maximum number of relevant documents for the queries including the target document.

$$Average\ Precision = \frac{1}{N_r}\sum_{R_i=1}^{R_i=15}\frac{n_i}{R_i}rel(i) \tag{3.2}$$

Where N_r is the total number of relevant documents at a given query, $n_(i)$ is the sequence of relevant documents found by a DMS for a given query, and R_i is the rank at which the relevant document was retrieved, and $rel(i)$ is the binary value {1,0} to be active if the document is relevant.

Another metric used for comparison was the Discounted Cumulative Gain (DCG). This metric uses a graded relevance scale of resulting documents from a search query. Their are several advantages for using DCG, namely: it combines the degree of relevance of a document with its respectful rank, not heavily affected by outliers, and it realistically interpret the loss from having target/relevant documents being later down in the results [Järvelin & Kekäläinen, 2002]. A score is given based on the relevance of the first 15 outputs (if any) from each query at a given DMS. The target document is given a score of 2, a relevant document is given a score of 1, while as a non relevant document is given a score of 0. The following

equation was used to calculate the DCG value for a given query [Ferro & Peters, 2019].

$$DCG = \sum_{i=1}^{15} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{3.3}$$

To better understand how well should a search engine perform, the *Ideal DCG* was calculated for each query. The ideal DCG is built on having the results that would resemble an ideal search procedure; having the target document as the first output (rank 1) and all the relevant documents proceeding it. Proceeding the relevant documents, you may have the irrelevant documents finishing at a total number of 15 outputs. If a query has 1 target document, 8 relevant documents (with the target counted), then an ideal result would have a relative score matrix of [2 1 1 1 1 1 1 1 0 0 0 0 0 0 0]. The normalized DCG was also calculated by dividing the DCG for a query by the ideal result.

OpenSearchServer and File System Search were also used in the search test and the results alongside those from the DMSs were recorded in a spreadsheet for ease of later analysis.

Part of the evaluation was also evaluating the functionalities of the DMSs. For that, a table was created having the main functionalities required by the DMS and a score was given to each of the chosen 5 DMSs to show how well they perform in those functionalities. The maximum score is 5, showing that the DMS did excellent in that requirement, while the lowest was 1 to which is shows that the DMS didn't have the requirement or performed exceptionally poorly in. The IR results were also a part of this table; where the DMS with the best performance was given a score of 5, and the lowest performance was given a score of 1. These can be seen in table 4.8.

Another metric proposed by Chapelle et al is the Expected Reciprocal Rank (ERR) which is a method to evaluate search processes by giving a graded relevance to the output. ERR works by giving a probability parameter in the metric. ERR uses a hypothetical assignment of a probability that a user stops at a given result. The following equation was used to calculate the value at each query and DMS.

$$ERR = \sum_{r=1}^{15} \frac{1}{r} \frac{2^g - 1}{2^{g_{max}}} \tag{3.4}$$

Where r being the rank of the document in question and g $\in \{0 \ldots 5\}$. Non relevant documents are given a g score of 0 Then, the probability that a user stops at a non-relevant document is then surely a zero. Relevant documents are given a score of 2 while target documents are assigned a score of 4. This would of course increase the probability that a user stops the search at a relevant or target document. As always, the first 15 results where considered for each query.

As for functionality testing, some of the tested DMSs had to OCR functionalities, additional solutions were needed in case that the preferable DMS lacked it. For that, another open-source software was found. **NAPS2** is a software that allows the user to upload a PDF

file of a scanned document and perform an OCR operation. The output file is a searchable and indexable text. Basically, text in an image is converted into a machine-encoded text where other basic text processing procedures can be performed. However, to evaluate the DMSs, NAPS2 wasn't used. This was done to give fair advantage to the DMSs that provided that service. From the 50 search topics, only 2 referred to documents that would require an OCR operation to retrieve. Thus, also bias regarding the lack of OCR was also somewhat eliminated.

4
Results and Analysis

In this chapter, the table containing the scores from the main functional requirement analysis is shown. Following these, the spreadsheet containing the results from the IR tests are shown. This would be followed by some statistical analysis to verify if the results from the DMSs are significantly different or not. The DMS with the highest score was later used in implementation procedure described in chapter 5.

4.1 Functionality Testing Results

Listed in the table below, the results from the functionality analysis testing. The max score for each test criteria is 5 while the lowest possible is 1. As for the OCR availability, please refer to the last paragraph of section 3.3.2. Red colored cells in OCR were assigned a score of 3 where an external software for OCR was utilized, yellow (a mediocre OCR) a score of 4, while green cells are given a score of 5.

Functional Property	DMS Name				
	SeedDMS	*Mayan-EDMS*	*OpenKM CE*	*Alfresco CE*	*NextCloud w/ ElasticSearch*
Document Versioning	5	5	5	5	5
Document Info	5	4	5	5	4
Document Preview	3	5	3	5	5
OCR					
Customizable User Permissions	4	4	4	4	4
Italian Language UI	5	5	4	5	5
IR Effectiveness	3	1	2	5	4
Document Assignment	5	5	5	5	3
Collaboration	3	3	3	3	3
Document Check in/Check out	5	5	5	5	3
System Logs	3	5	5	5	5
IR Indexing Procedure	4	2	4	5	5
Revision by Privileged Users	5	5	4	5	3
Supported Formats	3	3	5	5	5
Bulk Import	5	3	5	5	4
Maintaining Directory	5	1	5	5	5
Documentation and Support	3	4	4	5	4
Data Backup Capabilities	5	5	5	5	5
Access Permission to User Data	5	5	5	5	5
Guest Accounts	5	5	5	5	5
Document Tagging	5	5	5	5	5
Ability to add stop words	5	1	5	5	1
IR Incremented Indexing	2	3	2	5	2
Average score	4.17	3.87	4.30	4.87	4.04

Table 4.1: Results for the functional properties testing conducted on the DMSs.

4.2 IR Search Test Results

The results for each query is shown for each DMS. The results are shown in x pages to allow the production of visible results in the paper copy of the submitted thesis. The time required to produce the output of the search query (if the DMS had that functionality) was also recorded, however it was eliminated from the shown results as it serves no purpose in our testing.

4.2.1 Relevant Document Ranking Search Test Results

The results from the ranking tests are shown in tables 4.2-4.5. Each table has 2 DMSs/control search environments. They were separated in this manner to allow for a better visual display in a hard copy format.

Search Query	SeedDMS			Mayan EDMS		
	Rank of Highly Relevant Doc	*# of Retrieved Relevant Docs*	*Average Precision*	*Rank of Highly Relevant Doc*	*# of Retrieved Relevant Docs*	*Average Precision*
1°	1	1	0.3333	1	1	0.3333
2°	12	5	0.3496	NF	0	0
3°	3	9	0.9060	NF	0	0
4°	3	2	0.8333	NF	0	0
5°	1	2	0.6667	5	1	0.0667
6°	1	4	0.6667	NF	0	0
7°	5	4	0.6792	2	2	0.2917
8°	7	6	0.9107	3	3	0.5000
9°	3	1	0.1111	NF	0	0
10°	2	2	1.0000	NF	0	0
11°	6	3	0.5556	NF	0	0
12°	NF	0	0	NF	0	0
13°	15	1	0.1667	1	1	0.5000
14°	NF	0	0	NF	0	0
15°	1	5	1.0000	NF	0	0
16°	1	11	0.8638	NF	0	0
17°	1	1	0.1429	NF	0	0
18°	1	1	0.5000	1	1	0.5000
19°	3	3	0.3021	NF	0	0
20°	5	1	0.2000	NF	0	0
21°	NF	0	0	NF	0	0
22°	NF	0	0	NF	0	0
23°	2	2	0.5000	NF	0	0
24°	3	3	0.3611	1	1	0.3333
25°	1	2	0.6250	NF	0	0
26°	1	2	1.0000	NF	0	0
27°	1	3	0.5195	NF	0	0
28°	7	1	0.0714	14	1	0.0357
29°	4	2	0.3750	15	1	0.0167
30°	1	2	0.4444	NF	0	0
31°	8	3	0.2421	NF	0	0
32°	1	2	1.0000	NF	0	0
33°	4	3	0.7000	15	1	0.0222
34°	1	4	1.0000	1	1	0.2500
35°	1	1	0.5000	NF	0	0

Table 4.2: Relevant document search test results for SeedDMS and Mayan-EDMS.

Search Query	SeedDMS			Mayan EDMS		
	Rank of Highly Relevant Doc	*# of Retrieved Relevant Docs*	*Average Precision*	*Rank of Highly Relevant Doc*	*# of Retrieved Relevant Docs*	*Average Precision*
36°	1	2	0.7000	NF	0	0
37°	NF	0	0	NF	0	0
38°	1	1	0.5000	NF	0	0
39°	11	1	0.0455	NF	0	0
40°	NF	0	0	1	1	0.3333
41°	1	1	0.2000	NF	0	0
42°	1	5	0.8100	NF	0	0
43°	NF	0	0	NF	0	0
44°	15	1	0.0625	NF	0	0
45°	1	5	0.7254	3	1	0.0667
46°	2	1	0.5000	1	1	1.0000
47°	14	2	0.0595	14	8	0.3988
48°	4	2	0.7500	NF	0	0
49°	NF	0	0	1	1	1.0000
50°	1	6	1.0000	1	2	0.3333

Search Query	OpenKM Community Edition			Alfresco Community Edition		
	Rank of Highly Relevant Doc	*# of Retrieved Relevant Docs*	*Average Precision*	*Rank of Highly Relevant Doc*	*# of Retrieved Relevant Docs*	*Average Precision*
1°	2	1	0.3333	1	3	0.9167
2°	NF	0	0	1	1	0.2000
3°	2	5	0.4870	1	9	1.0000
4°	NF	0	0	1	1	0.5000
5°	1	2	0.6667	2	1	0.1667
6°	NF	0	0	1	6	1.0000
7°	1	1	0.2500	3	2	0.2917
8°	1	2	0.3333	3	3	0.5000
9°	1	2	0.6667	1	1	0.3333
10°	3	2	0.2667	2	2	1.0000
11°	1	1	0.3333	2	1	0.1667
12°	NF	0	0	NF	0	0
13°	NF	0	0	1	1	0.5000
14°	NF	0	0	3	4	1.0000
15°	15	2	0.4000	1	1	0.2000
16°	1	10	0.8421	1	11	1.0000
17°	6	7	0.7333	1	3	0.3095
18°	15	1	0.5000	1	1	0.5000
19°	1	3	0.7500	2	2	0.5000
20°	NF	0	0	1	1	1.0000
21°	15	1	0.0556	1	2	0.6667
22°	NF	0	0	1	1	0.3333
23°	2	4	0.6042	2	2	0.5000
24°	NF	0	0	1	1	0.3333
25°	NF	0	0	1	2	1.0000
26°	1	2	0.8333	15	1	0.5000
27°	1	3	0.8667	1	1	0.3333
28°	NF	0	0	1	1	0.5000
29°	1	4	1.0000	2	2	0.5000
30°	1	1	0.3333	1	1	0.3333
31°	15	1	0.1667	1	2	0.5556
32°	1	1	0.5000	1	1	0.5000
33°	2	3	0.6389	1	3	1.0000
34°	1	1	0.2500	3	4	1.0000
35°	1	2	1.0000	1	1	0.5000

Table 4.3: Relevant document search test results for OpenKM Community Edition and Alfresco Community Edition

Search Query	OpenKM Community Edition			Alfresco Community Edition		
	Rank of Highly Relevant Doc	*# of Retrieved Relevant Docs*	*Average Precision*	*Rank of Highly Relevant Doc*	*# of Retrieved Relevant Docs*	*Average Precision*
36°	1	1	0.5000	NF	0	0
37°	1	2	1.0000	2	2	0.5833
38°	NF	0	0	NF	0	0
39°	2	2	0.5833	1	1	0.5000
40°	1	3	1.0000	3	2	0.5556
41°	1	5	1.0000	2	3	0.6000
42°	4	5	0.5629	2	5	0.8767
43°	1	2	1.0000	1	1	0.5000
44°	1	3	0.6875	6	3	0.2405
45°	1	3	0.4462	1	1	0.2000
46°	1	1	1.0000	1	1	1.0000
47°	NF	0	0	3	8	0.4386
48°	2	2	1.0000	2	2	1.0000
49°	1	1	1.0000	1	1	1.0000
50°	1	1	0.1667	1	5	0.8333

Search Query	NextCloud w/ ElasticSearch			OpenSearchServer		
	Rank of Highly Relevant Doc	*# of Retrieved Relevant Docs*	*Average Precision*	*Rank of Highly Relevant Doc*	*# of Retrieved Relevant Docs*	*Average Precision*
1°	1	2	0.6667	1	1	0.3333
2°	1	1	0.2000	1	1	0.2000
3°	7	7	0.7778	4	5	0.5556
4°	1	1	0.5000	1	1	0.5000
5°	13	3	0.4658	1	2	0.6667
6°	1	1	0.1667	1	4	0.5694
7°	1	2	0.5000	1	3	0.7500
8°	3	3	0.5000	3	2	0.2778
9°	NF	0	0	1	1	0.3333
10°	6	2	0.1667	NF	0	0
11°	4	2	0.2167	1	2	0.6667
12°	3	3	0.4778	NF	0	0
13°	8	2	0.3750	NF	0	0
14°	1	4	1.0000	2	4	1.0000
15°	1	1	0.2000	NF	0	0
16°	1	7	0.5217	1	7	0.5969
17°	1	2	0.1905	1	1	0.1429
18°	1	2	1.0000	1	1	0.5000
19°	1	4	0.8500	1	2	0.5000
20°	NF	0	0	NF	0	0
21°	1	3	0.9167	NF	0	0
22°	2	3	0.8333	NF	0	0
23°	3	1	0.0833	1	2	0.5000
24°	NF	0	0	NF	0	0
25°	1	1	0.5000	1	1	0.5000
26°	2	2	1.0000	NF	0	0
27°	10	2	0.1333	1	2	0.6667
28°	1	2	0.7500	NF	0	0
29°	4	1	0.0625	NF	0	0
30°	3	3	0.6984	1	1	0.3333
31°	2	3	1.0000	2	1	0.3889
32°	1	2	1.0000	1	1	0.5000
33°	5	1	0.0667	1	2	0.6667
34°	1	4	1.0000	1	4	1.0000
35°	3	1	0.1667	1	2	0.8333

Table 4.4: Relevant document search test results for NextCloud with the ElasticSearch plugin and OpenSearchServer.

Search Query	NextCloud w/ ElasticSearch			OpenSearchServer		
	Rank of Highly Relevant Doc	*# of Retrieved Relevant Docs*	*Average Precision*	*Rank of Highly Relevant Doc*	*# of Retrieved Relevant Docs*	*Average Precision*
36°	3	2	0.3333	NF	0	0
37°	NF	0	0	NF	0	0
38°	7	2	0.1484	NF	0	0
39°	3	2	0.3333	NF	0	0
40°	9	3	0.2074	1	3	1.0000
41°	1	5	0.7962	1	1	0.2000
42°	2	5	1.0000	2	5	0.8767
43°	1	1	0.5000	1	1	0.5000
44°	5	2	0.1625	3	4	1.0000
45°	1	4	0.6800	1	2	0.4000
46°	4	1	0.2500	1	1	1.0000
47°	1	6	0.7500	1	8	1.0000
48°	NF	0	0	2	2	1.0000
49°	NF	0	0	1	1	1.0000
50°	9	6	0.7474	1	2	0.3333

File System Search							
Search Query	*Rank of Highly Relevant Doc*	*# of Retrieved Relevant Docs*	*Average Precision*	**Search Query**	*Rank of Highly Relevant Doc*	*# of Retrieved Relevant Docs*	*Average Precision*
1°	1	1	0.3333	26°	NF	0	0
2°	1	1	0.2000	27°	2	1	0.1667
3°	13	6	0.2099	28°	NF	0	0
4°	1	1	0.5000	29°	8	2	0.0982
5°	1	2	0.6667	30°	1	1	0.3333
6°	1	1	0.1667	31°	2	3	1.0000
7°	1	3	0.7500	32°	1	1	0.5000
8°	2	2	0.3333	33°	9	3	0.1513
9°	1	2	0.5556	34°	1	3	0.7500
10°	6	2	0.2262	35°	1	1	0.5000
11°	6	1	0.0556	36°	NF	0	0
12°	NF	0	0	37°	1	1	0.5000
13°	NF	0	0	38°	2	1	0.2500
14°	NF	0	0	39°	9	1	0.0556
15°	NF	0	0	40°	1	3	1.0000
16°	7	8	0.6595	41°	2	2	0.4000
17°	3	1	0.0476	42°	4	5	0.8767
18°	1	1	0.5000	43°	2	1	0.2500
19°	6	2	0.1131	44°	8	4	0.2815
20°	NF	0	0	45°	2	4	0.2533
21°	2	2	0.3889	46°	1	1	1.0000
22°	12	1	0.0278	47°	NF	0	0
23°	6	2	0.1333	48°	1	2	1.0000
24°	15	1	0.0278	49°	1	1	1.0000
25°	2	1	0.2500	50°	1	3	0.5000

Table 4.5: Relevant document search test results for File System Search.

4.2.2 DCG and ERR Values

The ideal DCGs and ERRs are shown next to each query. The normalized DCG are also listed as well as the scored ERRs. The results are shown in tables 4.6 and 4.7.

Search Query	Ideal DCG	SeedDMS		Mayan EDMS		OpenKM CE		Alfresco CE	
		nDCG	*ERR*	*nDCG*	*ERR*	*nDCG*	*ERR*	*nDCG*	*ERR*
1°	4.1309	0.7262	0.4688	0.7262	0.4688	0.7262	0.4688	0.9832	0.5391
2°	4.9485	0.4567	0.1077	0	0	0	0	0.6062	0.4688
3°	6.2545	0.8175	0.3762	0	0	0.6502	0.3859	1.0000	0.6402
4°	3.6309	0.6885	0.2500	0	0	0	0	0.8262	0.4688
5°	4.1309	0.8790	0.5156	0.2809	0.0938	0.8790	0.5156	0.4582	0.2344
6°	5.3047	0.8599	0.5703	0	0	0	0	1.0000	0.6047
7°	4.5616	0.5968	0.1953	0.5245	0.2656	0.6577	0.4688	0.4671	0.2031
8°	5.3047	0.7309	0.2740	0.5902	0.2969	0.6845	0.5156	0.5902	0.2969
9°	4.1309	0.3631	0.1562	0	0	0.8790	0.5156	0.7262	0.4688
10°	3.6309	0.7967	0.3281	0	0	0.4927	0.1656	0.7967	0.3281
11°	4.1309	0.5325	0.1562	0	0	0.7262	0.4688	0.4582	0.2344
12°	4.1309	0	0	0	0	0	0	0	0
13°	3.6309	0.1377	0.0312	0.8262	0.4688	0	0	0.8262	0.4688
14°	4.5616	0	0	0	0	0	0	0.7808	0.3203
15°	4.9485	0.7522	0.2891	0	0	0.3296	0.1406	0.6062	0.4688
16°	6.8225	0.9625	0.6325	0	0	0.9433	0.6394	1.0000	0.6581
17°	5.6380	0.5321	0.4688	0	0	0.6798	0.2727	0.6840	0.5156
18°	3.6309	0.8262	0.4688	0.8262	0.4688	0.2754	0.0938	0.8262	0.4688
19°	4.5616	0.4924	0.1914	0	0	0.9056	0.5469	0.6342	0.3281
20°	3.0000	0.3869	0.0938	0	0	0	0	1.0000	0.4688
21°	4.1309	0	0	0	0	0.0862	0.0156	0.8790	0.5156
22°	4.1309	0	0	0	0	0	0	0.7262	0.4688
23°	4.5616	0.6342	0.3281	0	0	0.6881	0.3008	0.6342	0.3281
24°	4.1309	0.5328	0.1875	0.7262	0.4688	0	0	0.7262	0.4688
25°	3.6309	0.9131	0.4805	0	0	0	0	1.0000	0.5156
26°	3.6309	1.0000	0.5156	0	0	0.9639	0.5000	0.2754	0.0938
27°	4.1309	0.8744	0.4907	0	0	0.9726	0.5344	0.7262	0.4688
28°	3.6309	0.2754	0.0670	0.2115	0.0335	0	0	0.8262	0.4688
29°	4.5616	0.5025	0.2109	0.0548	0.0063	1.0000	0.5703	0.6342	0.3281
30°	4.1309	0.8125	0.4844	0	0	0.7262	0.4688	0.7262	0.4688
31°	4.1309	0.3827	0.0824	0	0	0.1527	0.0469	0.8473	0.5000
32°	3.6309	1.0000	0.5156	0	0	0.8262	0.4688	0.8262	0.4688
33°	4.1309	0.6485	0.2297	0.1816	0.0312	0.6835	0.2891	1.0000	0.5469
34°	4.5616	1.0000	0.5703	0.6577	0.4688	0.6577	0.4688	0.7808	0.3203
35°	3.6309	0.8262	0.4688	0	0	1.0000	0.5156	0.8262	0.4688

Table 4.6: nDCG and ERR values in IR testing for SeedDMS, Mayan-EDMS, OpenKM CE, and Alfresco CE.

Search Query	Ideal DCG	SeedDMS		Mayan EDMS		OpenKM CE		Alfresco CE	
		nDCG	*ERR*	*nDCG*	*ERR*	*nDCG*	*ERR*	*nDCG*	*ERR*
36°	3.6309	0.9328	0.4875	0	0	0.8262	0.4688	0	0
37°	3.6309	0	0	0	0	1.0000	0.5156	0.6590	0.2656
38°	3.6309	0.8262	0.4688	0	0	0	0	0	0
39°	3.6309	0.2305	0.0426	0	0	0.6590	0.2656	0.8262	0.4688
40°	4.1309	0	0	0.7262	0.4688	1.0000	0.5469	0.6052	0.2500
41°	4.9485	0.6062	0.4688	0	0	1.0000	0.5891	0.8348	0.5469
42°	4.9485	0.9445	0.5578	0	0	0.5797	0.1962	0.8218	0.3859
43°	3.6309	0	0	0	0	1.0000	0.5156	0.8262	0.4688
44°	4.5616	0.0944	0.0234	0	0	0.8904	0.5391	0.3921	0.1103
45°	4.9485	0.9339	0.5551	0.3031	0.1562	0.7868	0.5228	0.6062	0.4688
46°	3.0000	0.6309	0.2344	1.0000	0.4688	1.0000	0.4688	1.0000	0.4688
47°	5.9535	0.2130	0.0647	0.4734	0.1019	0	0	0.5981	0.2271
48°	3.6309	0.6313	0.2109	0	0	0.7967	0.3281	0.7967	0.3281
49°	3.0000	0	0	1.0000	0.4688	1.0000	0.4688	1.0000	0.4688
50°	5.3047	1.0000	0.6047	0.6845	0.5156	0.5655	0.4688	0.9329	0.5891

Search Query	Ideal DCG	NextCloud w/ ES		OpenSearchServer		File System Search	
		nDCG	*ERR*	*nDCG*	*ERR*	*nDCG*	*ERR*
1°	4.1309	0.8790	0.5156	0.7262	0.4688	0.7262	0.4688
2°	4.9485	0.6062	0.4688	0.6062	0.4688	0.6062	0.4688
3°	6.2545	0.6883	0.2967	0.6091	0.3078	0.3586	0.0839
4°	3.6309	0.8262	0.4688	0.8262	0.4688	0.8262	0.4688
5°	4.1309	0.4645	0.1142	0.8790	0.5156	0.8790	0.5156
6°	5.3047	0.5655	0.4688	0.8328	0.5547	0.5655	0.4688
7°	4.5616	0.7960	0.5156	0.9056	0.5469	0.9056	0.5469
8°	5.3047	0.5902	0.2969	0.4713	0.2500	0.5453	0.3281
9°	4.1309	0	0	0.7262	0.4688	0.8473	0.5000
10°	3.6309	0.3687	0.0859	0	0	0.3861	0.0915
11°	4.1309	0.4064	0.1359	0.8790	0.5156	0.2587	0.0781
12°	4.1309	0.5610	0.1984	0	0	0	0
13°	3.6309	0.4344	0.1055	0	0	0	0
14°	4.5616	1.0000	0.5703	0.8382	0.3828	0	0
15°	4.9485	0.6062	0.4688	0	0	0	0
16°	6.8225	0.7947	0.5979	0.8159	0.6110	0.6582	0.2953
17°	5.6380	0.5953	0.4844	0.5321	0.4688	0.2661	0.1562
18°	3.6309	1.0000	0.5156	0.8262	0.4688	0.8262	0.4688
19°	4.5616	0.9690	0.5563	0.7960	0.5156	0.3073	0.0915
20°	3.0000	0	0	0	0	0	0
21°	4.1309	0.9832	0.5391	0	0	0.5792	0.2656
22°	4.1309	0.7865	0.3438	0	0	0.1963	0.0391
23°	4.5616	0.3288	0.1562	0.7960	0.5156	0.3191	0.0969
24°	4.1309	0	0	0	0	0.0654	0.0078
25°	3.6309	0.8262	0.4688	0.8262	0.4688	0.5213	0.2344
26°	3.6309	0.7967	0.3281	0	0	0	0
27°	4.1309	0.3036	0.0656	0.8790	0.5156	0.4582	0.2344
28°	3.6309	0.9448	0.4922	0	0	0	0
29°	4.5616	0.2832	0.1172	0	0	0.2805	0.0720
30°	4.1309	0.6859	0.2634	0.7262	0.4688	0.7262	0.4688
31°	4.1309	0.8213	0.3594	0.5792	0.2656	0.8213	0.3594
32°	3.6309	1.0000	0.5156	0.8262	0.4688	0.8262	0.4688
33°	4.1309	0.2809	0.0938	0.8790	0.5156	0.3411	0.0650
34°	4.5616	1.0000	0.5703	1.0000	0.5703	0.9056	0.5469
35°	3.6309	0.4131	0.1562	0.9639	0.5000	0.8262	0.4688

Table 4.7: nDCG and ERR values in IR testing for NextCloud w/ ElasticSearch, OpenSearchServer, and Windows File System search.

Search Query	Ideal DCG	NextCloud w/ ES		OpenSearchServer		File System Search	
		nDCG	*ERR*	*nDCG*	*ERR*	*nDCG*	*ERR*
36°	3.6309	0.5112	0.1719	0	0	0	0
37°	3.6309	0	0	0	0	0.8262	0.4688
38°	3.6309	0.3477	0.0742	0	0	0.5213	0.2344
39°	3.6309	0.5112	0.1719	0	0	0.2487	0.0521
40°	4.1309	0.3728	0.0771	1.0000	0.5469	1.0000	0.5469
41°	4.9485	0.9513	0.5634	0.6062	0.4688	0.5846	0.3281
42°	4.9485	0.8508	0.4016	0.8218	0.3859	0.7408	0.2922
43°	3.6309	0.8262	0.4688	0.8262	0.4688	0.5213	0.2344
44°	4.5616	0.3488	0.1172	0.7808	0.3203	0.4099	0.0918
45°	4.9485	0.8932	0.5563	0.7337	0.5156	0.5521	0.2607
46°	3.0000	0.4307	0.1172	1.0000	0.4688	1.0000	0.4688
47°	5.9535	0.8910	0.6047	1.0000	0.6298	0	0
48°	3.6309	0	0	0.7967	0.3281	1.0000	0.5156
49°	3.0000	0	0	1.0000	0.4688	1.0000	0.4688
50°	5.3047	0.6797	0.2424	0.6845	0.5156	0.7787	0.5469

4.2.3 Search Test Results Mean Values

Equation 2.5 and 3.2 were used to calculate the MRR and average precision values for each DMS. The constraints regarding the number of results considered (15) was applied at all queries and DMSs. As seen from the results, Mayan-EDMS struggled to produce relevant results for many search queries. The search capabilities are very limited but the functionalities provided as a DMS are quite good (section 4.1). The average produced rank for target documents, MRR, and average precision were calculated and can be seen in table 4.8 in the next page.

A lower value a DMS obtains in average ranking of the higher relevant document (target documents), the better its search capabilities. Alfresco has the lowest average ranking by far and thus would be easily acknowledged as the best in this category (The means where not mentioned in the table since these can be replaced with the MRR). It is important to note that if the search system didn't retrieve the higher relevant document in the first 15 results, a ranking score of infinity (the higher relevant documents where not retrieved) was assigned for that query. The higher the MRR the better it is in IR evaluation terms. Again, Alfresco had the best results, this time with OpenSearchServer, the search engine used as a control being the second best with a difference of 0.1064 in the score, mounting to a 17.2% increase in the value of the metric between the best and the second best system with respect to the metric. The system that produced the best results in obtaining the highest percentage of relevant documents was NextCloud with SeedDMS being a close second (0.0178 difference in score, only 2.68% of the total value scored by NextCloud).

The normalized DCG show how close a system gets to the ideal value per query. A nDCG value of 1 signifies that the results from a search system was identical to the ideal one for a given query. In [Chapelle et al., 2009] , the author didn't use ideal ERR values since it would be difficult to suggest that a user would stop the search process with certainty at a given relevant result. Thus, ERR values are shown as is, with no normalization. A higher nDCG and ERR values suggest a higher IR performance.

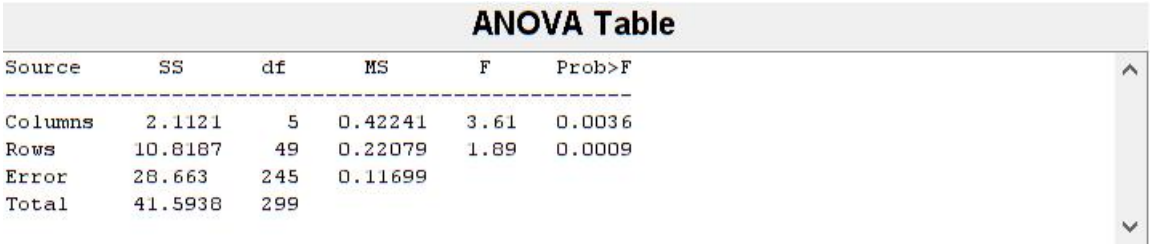
Evaluation Metric	*DMS / Control Systems*						
	Alfresco	OpenKM	NC w/ES	SeedDMS	MEDMS	OSS	FSS
Mean Average Precision	0.5594	0.4552	0.4579	0.4576	0.1196	0.4258	0.3402
Mean Reciprocal Rank	0.7247	0.5603	0.5292	0.5055	0.2129	0.6183	0.4918
Mean nDCG	0.7121	0.5538	0.5844	0.5597	0.1959	0.5599	0.5003
Mean ERR	0.3932	0.3054	0.2980	0.2785	0.1050	0.3284	0.2574
Mean % Retrieved Relevant Documents	0.6109	0.5293	0.6630	0.6452	0.1640	0.4344	0.4887

Table 4.8: Mean values for the evaluation metrics for the tested systems.

4.2.4 Analysis of Variance (ANOVA)

An ANOVA table with 2 variable assessments was produced for the percentage of retrieved documents, nDCG, ERR, and the average precision for each query per each DMS except for Mayan-EDMS. Mayan-EDMS was kept out of the assessment as it scored very badly in all of the IR related experiments and thus would certainly show significant differences in each category. We want to know if there is any significant difference between the tested systems and having one of the systems being very different in such an obvious manner would defeat the reasoning behind the testing. The ANOVA table was generated comparing the rest of the 4 DMSs against each other and against the other 2 control systems. The input into the MATLAB function was columns containing the results from the search testing ordered from the best to worst for each evaluation metric. For our testing, we used a level of significance of 0.05 (5%). That would define the confidence level to be 0.95.

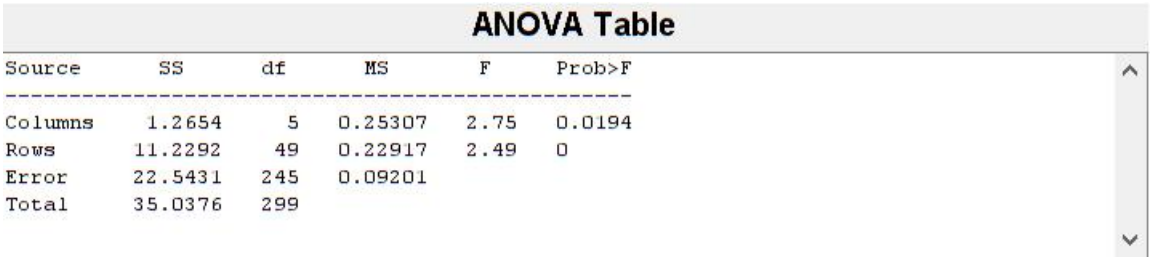
The function *anova2* was used in MATLAB to build the ANOVA figures. If the p value would state that the means are significantly different, the *multcompare* (Tukey HSD test) function would be used to compare the sample means individually to determine which of them have significant difference. The data input into the *multcompare* is sorted in descending order of sample means. This would help the user to better visualize the overlapping that happens between the data produced by the systems. This would also ease the process of distinguishing the outlier systems (significantly different) for each metric.



ANOVA Table

Source	SS	df	MS	F	Prob>F
Columns	2.1121	5	0.42241	3.61	0.0036
Rows	10.8187	49	0.22079	1.89	0.0009
Error	28.663	245	0.11699		
Total	41.5938	299			

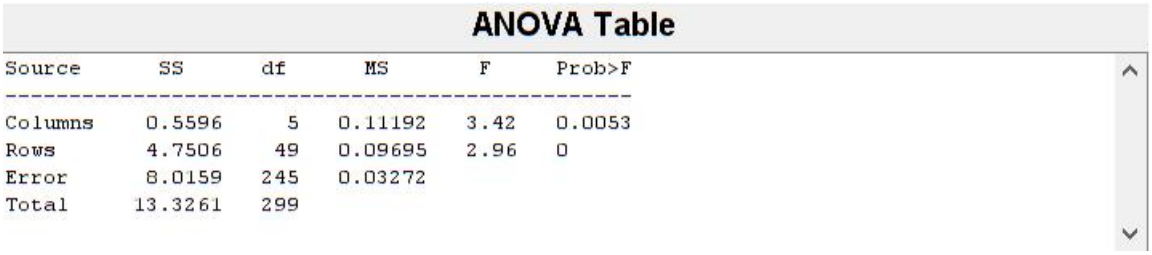
Figure 4.1: ANOVA Table for percentage of retrieved documents per query.



ANOVA Table

Source	SS	df	MS	F	Prob>F
Columns	1.2654	5	0.25307	2.75	0.0194
Rows	11.2292	49	0.22917	2.49	0
Error	22.5431	245	0.09201		
Total	35.0376	299			

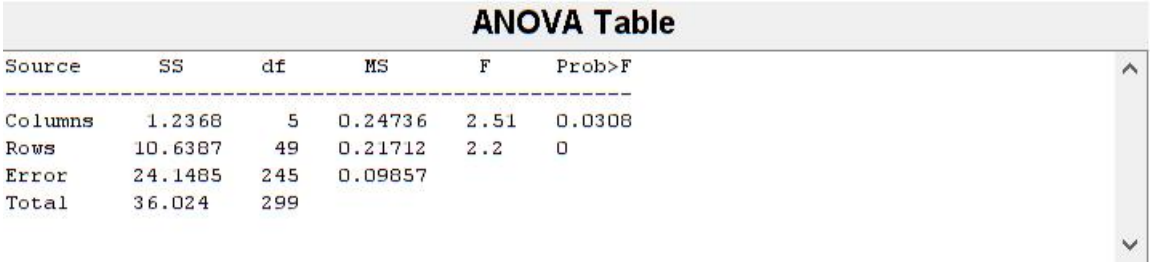
Figure 4.2: ANOVA Table for the nDCG values for the tested systems.



ANOVA Table

Source	SS	df	MS	F	Prob>F
Columns	0.5596	5	0.11192	3.42	0.0053
Rows	4.7506	49	0.09695	2.96	0
Error	8.0159	245	0.03272		
Total	13.3261	299			

Figure 4.3: ANOVA Table for the ERR values for the tested systems.



ANOVA Table

Source	SS	df	MS	F	Prob>F
Columns	1.2368	5	0.24736	2.51	0.0308
Rows	10.6387	49	0.21712	2.2	0
Error	24.1485	245	0.09857		
Total	36.024	299			

Figure 4.4: ANOVA Table for the average precision values for the tested systems.

From the P-values in figures 4.1-4.4, we can conclude that the at least one of the system samples at each evaluation metric produced results that are significantly different from those

produced by other systems. If our null hypothesis states that H_o: μ_1=μ_2=μ_3=μ_4=μ_5=μ_6, then we reject it and accept the alternative stating that H_1: $\mu_1 \neq \mu_2 \neq \mu_3 \neq \mu_4 \neq \mu_5 \neq \mu_6$. If the produced P-values was higher than the significance level, we would fail to reject the null hypothesis stating that the means of the tested groups are equal for that metric measure. Thus no significant difference is present between the tested groups. The Tukey HSD test was therefore also conducted to distinguish the systems producing significantly different results from the rest.

Figure 4.5, displayed sideways in the next page, shows the Tukey HSD plot for each metric. [A] Shows the plot for the samples tested for the percentage of retrieved documents. Data from OSS was inferior and showed significant difference against data collected from NextCloud and SeedDMS. [B] Shows the plot for the samples tested for the ERR. Data from Alfresco was superior and showed significant difference against data collected from FSS. [C] Shows the plot for the samples tested for the nDCG. Data from Alfresco was superior and showed significant difference against data collected from SeedDMS and FSS. [D] Shows the plot for the samples tested for the average precision. Data from Alfresco was superior and showed significant difference against data collected from FSS. The data was sorted by means to better show how the samples differ.

Figure 4.6 shows the box plot for each metric. The box plot defines the quartiles from readings at each system. The first quartile is the one coming from the lower whisker, the line outside the box and ending at the box edge. The second quartile is the segment from the lower edge of the box to the red line defining the median. The third quartile ranges from the median line to the upper edge of the box. The fourth quartile group extends from the upper edge of the box to the end of the upper whisker. A boxplot can have no whiskers if no data points exist beyond the upper/lower quartile (second and third group). Some data points shown as red + signs are known as outliers. An outlier is an observation that is numerically distant from the rest of the data.

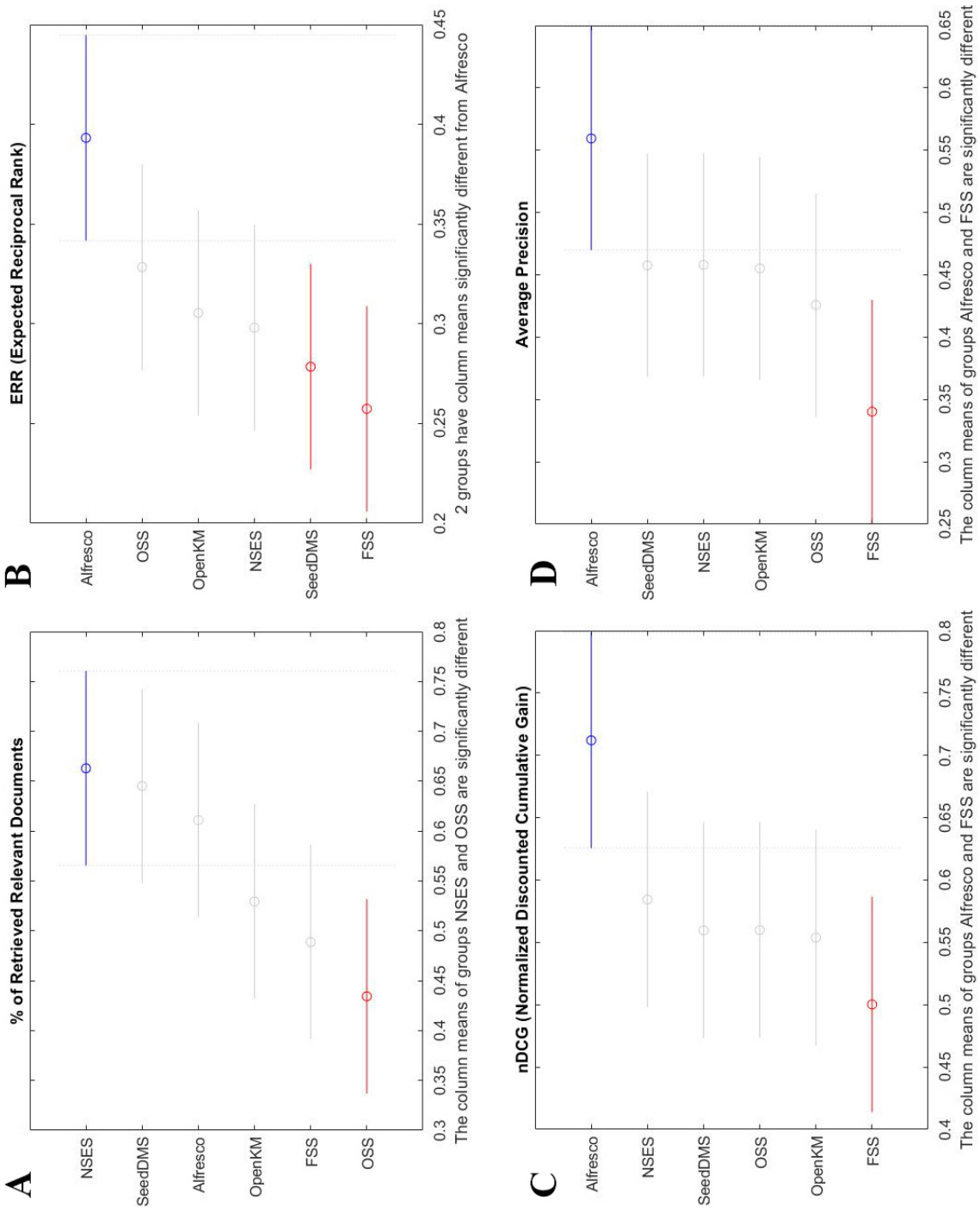


Figure 4.5: TukeyHSD plots for the evaluation metrics from the tested systems.

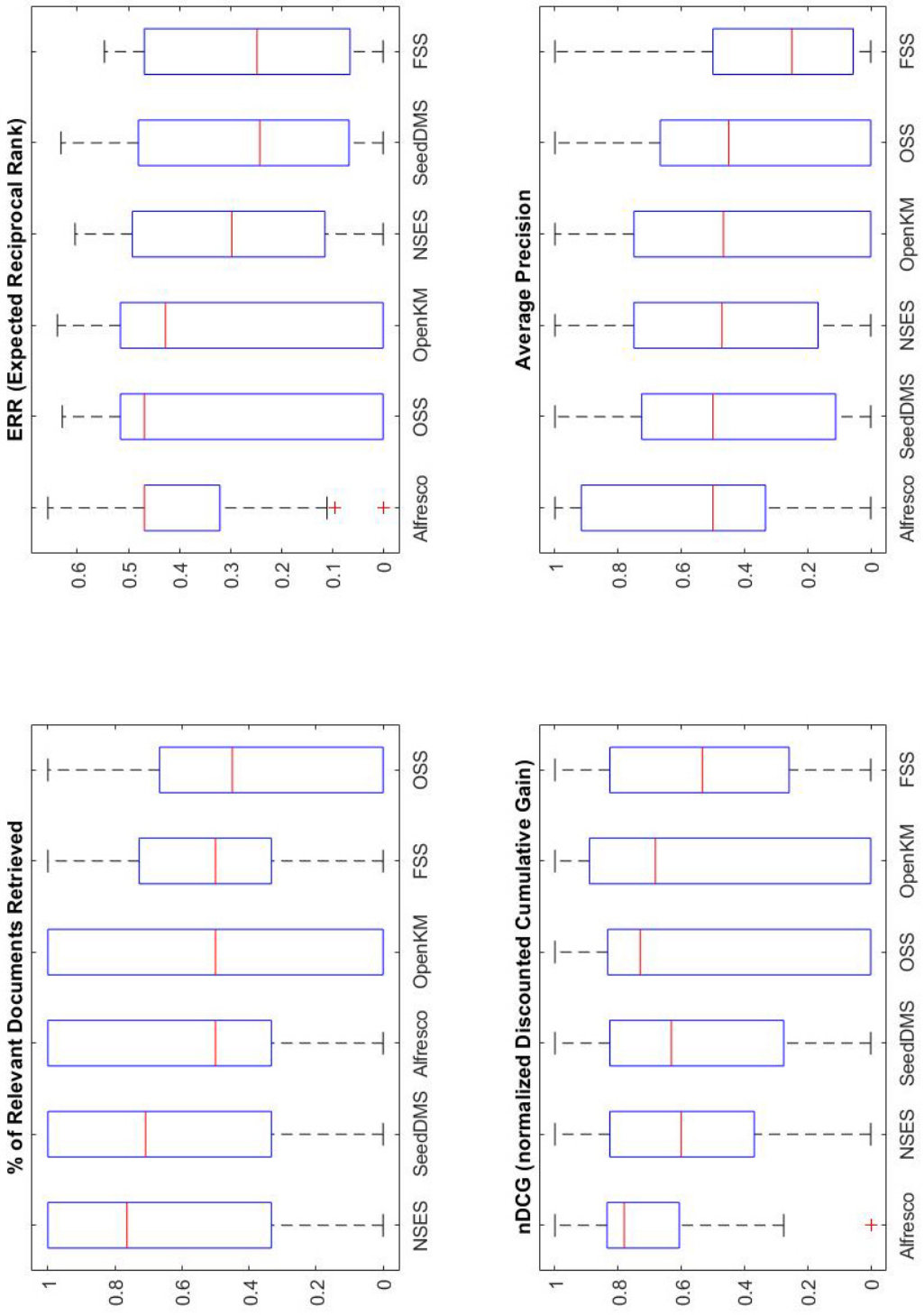


Figure 4.6: Box plots showing the evaluation metrics from the tested systems.

5 Implementation and Validation

When implementing a DMS in an enterprise for the first time, there are several steps that are recommended to follow to increase the probability of making the project succeed. But, to follow the proper implementation methodology, a quality standard measure is probably a better option. For that, the ISO 9001:2008 Quality Management Systems requirements procedure was followed in some form. The following list is the steps recommended to fulfil the requirement for that quality control standard [Hernad & Gaya, 2013].

- Step 1. Definition of document requirements
- Step 2. Evaluation of existing systems
- Step 3. Identification of document management strategies in the organization
- Step 4. Design the Document Management System
- Step 5. Implementation of the Document Management System
- Step 6. Maintenance and continuous improvement of the Document Management System

5.1 Initial Implementation

As from the functional requirement scoring table and the IR search test results, Alfresco CE was characterised as the best open-source DMS solution for DAP usage. To prevent an excessive and sudden change in the company's approach in document management, the solution was implemented in their quality control department at first. This was done to

evaluate the DMS from the point of view of a group of users with similar yet diverse tasks. If the implementation was labelled successful, more departments would be recommended to adopt the solution for their document management.

We implemented the DMS in a Virtual Machine embedded in the local servers at DAP. The VM was set to have the following specifications:

- Memory: 16GB of RAM
- Processor: Intel Xeon Quad-Core @2.6GHz
- OS: Ubuntu Server 18.04 LTS
- HDD: Expandable, with an initial 200GB of storage memory

The VM was assigned a static IP address that would be accessible to the users in the local network. To access the DMS from outside the local network, users are granted a VPN access to which the IP of the VM would be accessible. This is done to serve the needs of employees in other production sites or employees on the move to conferences or business meetings. The firewall in the VM was implemented such as the root login can't be done remotely and the to accept TCP connections to the ports running the Alfresco DMS. Since Alfresco CE is open-sourced, many modifications are applicable. The most notable modifications implemented was the change of the logo inside the website to match that of DAP, and also the usage of editable user permissions to create shared folders and the monitoring capabilities for heads of departments and the administrators. The OCR plugin was applied such that all uploads to a given folder are ran into the OCR processing and successful ones are transferred to a shared folder with the encoded text embedded. This is done so that to know if any documents had an error in their OCR, if so, the processed document won't be forwarded to the shared folder. The user then would be recommended to re-scan the document as possible error is due to low quality scan.

5.2 User Accounts Synchronization

The DMS started with the **User Roles and Privileges** mentioned in section 2.6.1 of this dissertation. The user accounts were created and handed over to the employees at the the quality control department. All those users have been added to a user group to ensure the atomicity of the files circling their department. The user accounts were created by syncing the active directory (ad) user group inside DAP using the LDAP (Lightweight Directory Access Protocol). The syncing process allows the domain admin to decide the user group who can use the DMS. Since machine operators are not required to use the DMS, there is no need to have an account. The syncing also allows the automatic creation of an account when a new user is added to a group.

To configure the LDAP settings for alfresco, the following settings were added to the *alfresco-global.properties* files [Alfresco-Services, 2017].

```
authentication.chain=alfinst:alfrescoNtlm,ldap1:ldap ad
ldap.authentication.allowGuestLogin=false
ldap.authentication.userNameFormat=%s@domain.com
ldap.authentication.java.naming.provider.url=
ldap://domaincontroller.domain.com:389
ldap.authentication.defaultAdministratorUserNames=DomainAdmin
ldap.synchronization.java.naming.security.principal=
admin@domain.com
ldap.synchronization.java.naming.security.credentials=*****
ldap.synchronization.groupSearchBase=ou=Security
Groups,ou=Alfresco,dc=domain,dc=com
ldap.synchronization.userSearchBase=ou=UserAccounts,ou=Alfresco,
dc=domain,dc=com
```

This instance regarding the ldap-ad is given the name ldap1. This is done by editing the *authentication.chain* property. *ldap.authentication.allowGuestLogin* controls the access of guest (unauthorized users).

In addition, *ldap.authentication.userNameFormat* is a template that defines how user IDs are expanded into Active Directory User Principal Names (UPNs) containing a placeholder %s. This stands for the unexpanded user ID entry. The domain name is added to to match that used in the AD.
ldap.authentication.java.naming.provider.url is the identifier used to point to the location of the LDAP URL. This include the host name, the domain name, and the port number (port 389 is the default port got LDAP applications). To give administrator privileges to an imported user, the user name is added to the identifier *ldap.authentication.defaultAdministratorUserNames*. *ldap.synchronization.java.naming.security.principal* is the username of the priveleged user inside the LDAP service to allow the alfresco synchronizer to view the users and groups to be imported. The entry credentials for that users is in the identifier *ldap.synchronization.java.naming.security.credentials*. *ldap.synchronization.groupSearchBase* this identifier is set to be the Distinguished Name (DN) of the Organizational Unit (OU) below which security groups can be found. Lastly, *ldap.synchronization.userSearchBase* is set for the DN of the OU below which user accounts can be found [Alfresco-Services, 2017].

Additional settings are tweaked internally to only create the user on their first sign in. The main page and the a folder containing all files and directories are created after the first log in. This is done to limit the number of users to only those who would be willing to use the DMS.

5.3 Backup Planning

For backup plans, there are 2 options to consider. The first is to export content in the mode saved in the database from where Alfresco is deployed. The second option to consider is a

simple file export to an external/internal filesystem.

5.3.1 Database Backup

That is, backing up the database and the files in the form of year-month-day-hour-minute. The database is also backed up and thus the system can be migrated from one Alfresco instance to another if necessary. This has the advantage of being reliable, fast, and automatable. It is also more viable in case of Alfresco version update where the entire database with the file system as stored in the old database is required for migration.

To perform the database backup, the alfresco instance should be stopped before all. The user can easily perform a database dump using the PostgreSQL script *pg_dump* found in *{ALFRESCO_HOME}/postgresql/bin*.

```
sudo ./pg_dump alfresco --user alfresco >
$ALFRESCO_HOME/alf_data/db-backup.sql
```

When this script is ran, the database entry credentials are requested. The database backup is exported in an SQL file containing all the tables and the directories at the time of the backup. The user is then requested to head to the *{ALFRESCO_HOME}/alf_data* directory where the files are stored. The folder *content.deleted* can be deleted if the user wishes to do so. The entire directory is compressed and stored away.

To restore the backup using this methodology, the SQL file for the database and the relevant filesystem are required. As with backing up, for restoring, the alfresco instance needs to be turned off by stopping the *alfresco.sh* script. The old database created by the installation package is dropped and recreated again. This is done to ensure that no inconsistencies happen in the *Foreign Keys* linking tables together. The following commands are applied in the postgres shell ran from the postgresql folder in the installation directory.

```
su - postgres
psql
DROP DATABASE alfresco;
CREATE DATABASE alfresco WITH OWNER alfresco;
psql -U alfresco -d alfresco
ALTER USER alfresco WITH PASSWORD '{pw in alfresco-global.properties}';
```

The script for restoring the backed up database, *pg_restore*, is ran using the following command.

```
sudo ./pg_restore -d alfresco alfresco_postgres.tar
```

Where alfresco_postgres.tar is the compressed file containing the backed up database SQL configuration output from the *pg_dump* operation. The script will then restore the database

to the state it was in at the moment of backup. The backed up compressed alf_data directory is then inflated in place of the new one to restore the file and folders to the alfresco instance. The alfresco service can now be turned back on and the system is restored.

5.3.2 File Export Backup

Another method for backing up is to export all/some files stored inside the alfresco instance to an external directory inside the filesystem of the deploying machine. This is a preferable solution in case of migration to another DMS other than alfresco where the administrator wants to keep the directories as stored in the Alfresco. It is also a viable solution if the files in the back up are required to have the same file names and relevant file metadata. The method described in 5.2.1 stores the document as stored in the database, hence stored with the file name being the file version and stored inside directories pointing to the time of uploading the document. The database system can make sense of such method to restore the file names and metadata inside Alfresco as these are stored in other tables inside the postgres database. A regular user however can not make sense of such taxonomy, for that reason, exporting the files can be considered viable solution for backing up the system.

For this backup method, another plugin is required for installation, the *Alfresco Bulk Export Tool* (`github.com/Alfresco/alfresco-bulk-export`). The installation procedure is similar to that of the AutoCAD plugin (the *amp* file was applied using the *apply_amps.sh* script), but no additional settings are needed in the *alfresco-global.properties* file. The tool works by specifying a folder inside the DMS where the files needed for backup are placed alongside a folder to export to inside the hosting machine. The tool is accessible using the the following link format.

```
http://{host}:{port}/alfresco/service/extensions/
bulkexport/export?nodeRef={noderef}&base={base}&
ignoreExported={ignoreExported?}&exportVersions=true
&revisionHead=false&useNodeCache=true&
```

{host}: Is the host name/IP address used to access the Alfresco service.

{post}: The port number from which the Alfresco service is running.

{nodeRef}: The node reference required for exportation. This can be retrieved by clicking on *View Details* on the folder/file to export. This will forward you to the workspace at which the folder/file is present. The *nodeRef* follows the this format *workspace://SpacesStore/UUID*.

{base}: The target directory to which the folder/file is exported. An example would be */home/user/target_directory*. It is recommended that the folder is set to be a public folder to allow the dedicated alfresco user to write with it avoiding permission related errors.

{ignoreExported}: An optional parameter stating if the export tool should ignore files/directories already exported into the target base directory. Entries should be either *true* or *false* with *false* being default option.

{exportVersion}: A parameter specifying if the tool should export all version of a given document inside the exported directory.

{revisionHead}: A parameter specifying that the latest version of a given document will be exported with head *latest* assigned to it. The true option can only be set if exportVersion is set to true. If revisionHead is set to false, then default numbering is used as head for the exported documents.

{useNodeCache}: if true then a list of nodes to export is cached to the export area for future repeated use. The default value is false.

After accessing the link, the user is promoted to enter the entry credentials. The user is then forwarded to a page showing the source and target directories to allow the user to check if the data provided is correct. The page would also show any errors if any to make the user aware of any problems before initializing the export procedure. If the data provided match the intended process, the user should refresh the page and the export starts. After the export process is over, the user can access the files through the file system. The tool would leave a *xml* file containing metadata such as the file owner, date of upload, version number, and other information regarding the exported file. For safe backup practices, the exported folder should be compressed and stored away alongside a naming template stating the date and time of the export process.

To restore the backup into the DMS, the contents are simply re-uploaded back into Alfresco using the Bulk Import Tool plugin that is preadded into the Alfresco service by default. This tool can be accessed through this link `http://{host}:{port}/alfresco/service/bulkfsimport`. After the credential entry, the user is promoted to specify the directory to import inside the hosting machine and the directory where the documents are placed inside the DMS.

Backup Implementation

The method chosen for implementation was the file export backup. To apply the backup formally, certain procedures are followed. A main directory was created inside the hosting machine that was set to host the files from the DMS. A *crontab* task was set to export the file system into the given directory at a given time everyday. Since the export tool is only accessible via a link, a bash script containing a *curl* command was put into the crontab task to follow. The curl command also passed user entry credentials to allow access to the tool. The backup was initially only done for the shared folders since they were the ones modified often and are more vulnerable for data loss due to human error. After that, it was decided that all files are to be backed up with the **{ignoreExported}** and the **{exportVersion}** options in the export tool set to true.

5.4 GUI Customization

After implementation, the visual design of the DMS was changed for purpose of customization and personalizing. This was done match the design identity of DAP set by their logo and other color schemes followed in their publication/products/IT services. The color scheme chosen was blue/drak grey. To create a custome theme to fit the proposed scheme, a free theme creation tool (Alfresco ThemeBuilder) was used. This tool is accessible at `https://flex-solution.com/theme/`. The HEX color codes were entered to fit the design in mind and the amp file is generated accordingly. The theme was applied using the regular amp installation method with *-force* parameter and the Alfresco service was restarted.

The login page was also modified to match the color scheme. To do so, HEX color code modifications were applied to the *login.css* file found in *{ALFRESCO_HOME}/tomcat/webapps/share/components/guest* directory. Other css files can also be found in the directory level above the suggested one (components). More experimentation can be done in case that more customization is required later on.

Figure 5.1: Screenshot of the replication jobs page at the admin tools.

Figure 5.2: Screenshot of the replication jobs page at the admin tools.

Figure 5.3: Screenshot of the login page.

E-Signatures

Other than the activity logs shown in the user/worksite dashboards, the system stakeholders also wanted an implementation of an e-signature PDF files. This was wanted to add a way for department managers to know if a user has fully read new terms/regulations or any other important document published internally. The e-signature was advised to be put by users only in case that they have fully read a document or to show some sort of a vote. The e-signature is not a bind to any legal agreement by EU law and is only used to show that a document is read.

To apply such a system, yet another plugin was chosen. There were several options compatible to the needs set by the stakeholders, but the one that fit the most was *Eisen Vault E-Signature* (`https://github.com/sumitt/eisenvault-esign`). The *repo* and the *share* plugin were built using *mvn clean package* command. The target amp files were placed in the *amps* and the *amps_share* directories respectively. The applied amps allowed the user to add an electronic signature in the form of an image (accessible at the user dashboard). A PDF document can be signed by any user with read/edit permissions. The e-signature adds a new version with a new page to the document with the signature, username, date, and time at which the signature is applied.

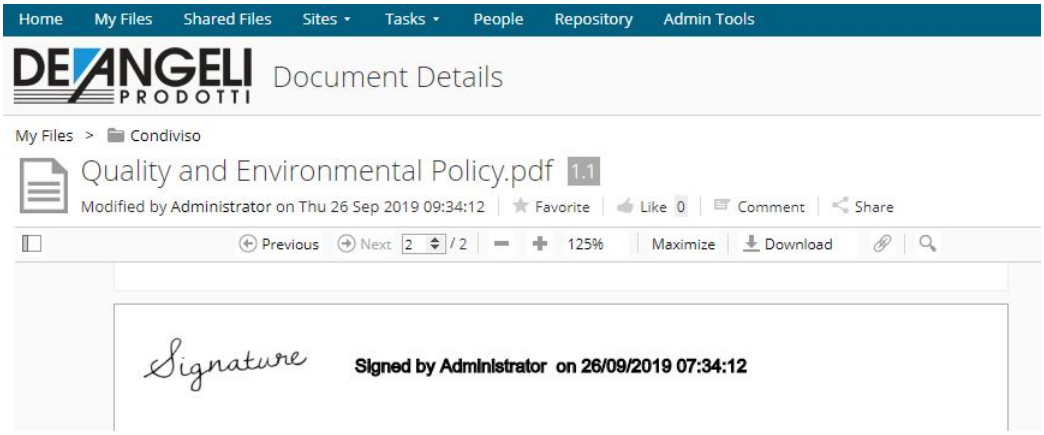


Figure 5.4: Screenshot showing the applied e-signature.

6 Conclusion

The object of this thesis was to find a DMS (Document Management System) compatible with the functional requirements set by DAP (De Angeli Prodotti). A list of DMS webservers was built based on readily available opensource materials. Some of the opensource DMSs in the list were deployed and discarded, others, were discarded without deployment due to lack of the necessary of functional properties. At the end of the research phase, a list of 5 contenders was collected. These five were evaluated based on the list of functional requirements.

The same five were tested for their IR (Information Retrieval) capabilities. For that, an extensive research was conducted to determine the best evaluation metrics for our application as an enterprise. Several metrics were acquired from each DMS from a a target document search test. The test produced document search results out of a 5000 document database dump from DAP's quality department. The collected metrics were the target document rank, MRR (Mean Reciprocal Rank), average precision, DCG (Discounted Cumulative Gain), and the ERR (Expected Reciprocal Rank). The DMS IR results were compared against each other and against results from 2 control groups, OpenSearchServer, and a File System Search. The presence of significant difference in the produced results was investigated. ANOVA (Analysis of Variance) testing was conducted to distinguish if the IR results were significantly different; a significance level of 5% was used in this application. Alfresco Community Edition Content Management System was the best performer in most of the tests conducted in the functionality analysis and in the IR testing. The conducted tests showed an increase of 47.3%, 64.4%, 42.3%, 52.8%, 25.0% in the mean reciprocal rank (MRR), mean average precision, average normalized discount cumulative gain (DCG), average expected reciprocal rank (ERR), and the average percentage of retrieved documents respectively.

The implemented DMS by itself isn't capable to perform all the required functions; for that several plugins were installed to ensure that it satisfies the requirements. The installed

plugins were, an OCR plugin, AutoCAD previewer, bulk export tool, e-signature, and a GUI customization. The user groups in DAP were exported into the running Alfresco instant using LDAP (Lightweight Directory Access Protocol). This enabled employees to use their regular credentials to access the DMS. SMTP (Simple Mail Transfer Protocol) was also implemented to allow the DMS to send notifications to users when changes occur in worksites, document libraries, or workflows that they are engaged in. A user guide was also created for users for ease of integration into the new system. A different guide was also created for system admins to ensure system maintenance and ease of troubleshooting.

A systematic backup plan was also implemented. Using Crontab, the bulk export tool systematically copied all document uploads taking place inside the DMS. The directory pointer for file copying is set to copy all worksite files, user files, and shared files all while maintaining directories as created by users. This method is preferred over traditional database backups as it also keeps the file names and relevant templates with author names and creation dates. Bulk import tool was also ran with a set schedule to ensure that the AD (Active Directory) inside the shared file system at DAP is also available to DMS users.

Future works might include a HTTPS (Hypertext Transfer Protocol Secure) implementation. Currently Alfresco as applied in DAP is ran in their local network. In the future, they might consider releasing the application on the internet to allow their users to access the DMS from outside the company's premises. Additional security measures should also be taken in such case. New workflow models are to be introduced in the future. These can be designed using Activiti, an open-source workflow engine written in Java that can execute business processes. This would allow DAP to customize models in an ad-hoc manner for process that suits their needs. These models can be ingested into Alfresco via the admin panel. DAP might also implement a PMS (Project Management System) in the near future. One possible addition is to find a way to synchronize the users' calendars and workflows from Alfresco to the ones created in the PMS.

A
Installation Methods

For this section, I will only consider the installation method of the 5 DMSs that are put into the information retrieval testing. Most installation steps were collected from the official repositories for the opensource software alongside some troubleshooting for missing dependencies and unreported steps. The installation procedure was ran using virtual machines (VMs) implemented on Oracle's VirtualBox. From those Vms, the webservers were deployed and tested by users in the enterprise. A master Linux Ubuntu 16.04 distribution with the latest updates and upgrades was cloned every time an installation is carried out.

A.1 SeedDMS

SeedDMS is a free document management system with an easy to use web based user interface. The system is written in PHP and the database is implemented in MySQL. It has been around for many years and thus the long term development has made it a mature, powerful and enterprise ready platform for sharing and storing documents. The project homepage can be found at `www.seeddms.org`. SeedDMS uses the Zend framework as its main search engine. This enables the users to perform fulltext searching to comply with the functional requirement.

The installation starts by installing MySQL server (5.7) and setting a root password and a **seeddms** user and password. The initial database is created and the user privileges are granted

to the DMS user (seeddms with password seeddms).

```
> CREATE USER 'seeddms'@'localhost' IDENTIFIED BY 'seeddms';
> GRANT ALL ON seeddms.* TO 'seeddms'@'localhost';
> FLUSH PRIVILEGES;
```

The webserver was ran on Apache 2.4 server with the *mod_rewrite* module being enabled. The PHP release used was the stable 7.0 release alongside the Apache 2.0 extension library and the php-common, php-mbstring, php-mysql, and the php-gd packages. The PEAR framework was used to obtained the reusable php components such as the Log, Mail, and WebDAV server. The latest release of SeedDMS was obtained from the **SourceForge** website and the package was inflated into the /var/www/html default directory for Apache 2 webservers. SeedDMS requires the activation of its install tool, to do so, an empty file named **ENABLE_INSTALL_TOOL** is created in the **/conf** directory in the inflated root folder.

To index and display input documents, SeedDMS requires additional software whose main objective is to catch the documents in plain text mode in the command window. Several free software can do so, but each type of document (MIME Type) has a different software requirement. The installed Linux software packages include: poppler-utils, imagemagick, id3, catdoc, docx2txt, a2ps and antiword. Other requirements include several python pip packages such as: html2txt, csvkit, and xlsx2csv. Additionally, the Zend framework was also installed from its main Linux Ubuntu repository.

Once the installation is initialized through the main PHP install file, the directories for the pear packages and the database credentials are specified. Default administrator logins are then used to access the DMS. In the system settings, it is important to set the Zend framework as the default search engine as several PHP files are directly linked to that option being turned on, otherwise most functionalities of the system won't work. The MIME types are set along with the software/package needed for the indexing. The converter uses the softwares to convert source document (% s) to displayable text. File extensions, MIME types, and the relevant converters are shown in table A.1.

File Extension	MIME Type	Converter
.*doc	application/msword	catdoc %s
.*docx	application/vnd.openxmlformats-officedocument.wordprocessingml.document	docx2txt %s
.*html	text/html	html2text %s
.*pdf	application/pdf	pdftotext -q -nopgbrk %s - | sed -e 's/ [a-zA-Z0-9.]{1} / /g' -e 's/[0-9.]//g'
.*ppt	application/vnd.ms-powerpoint	catppt %s
.*pptx	application/vnd.openxmlformats-officedocument.presentationml.presentation	unzip -qc "%s" ppt/slides/slide*.xml | grep -oP '(?<=a:t).*?(?=/a:t)'
.*txt	text/plain	cat %s
.*xls	application/vnd.ms-excel	xls2csv %s
.*xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	xlsx2csv %s

Table A.1: MIME Types of file extensions and the converters required to display stored texts for indexing purposes.

The documents are then imported into the system using its built-in importer and then the indexing process is initiated. User groups, accounts, and privileges are created and the system is put into testing phase.

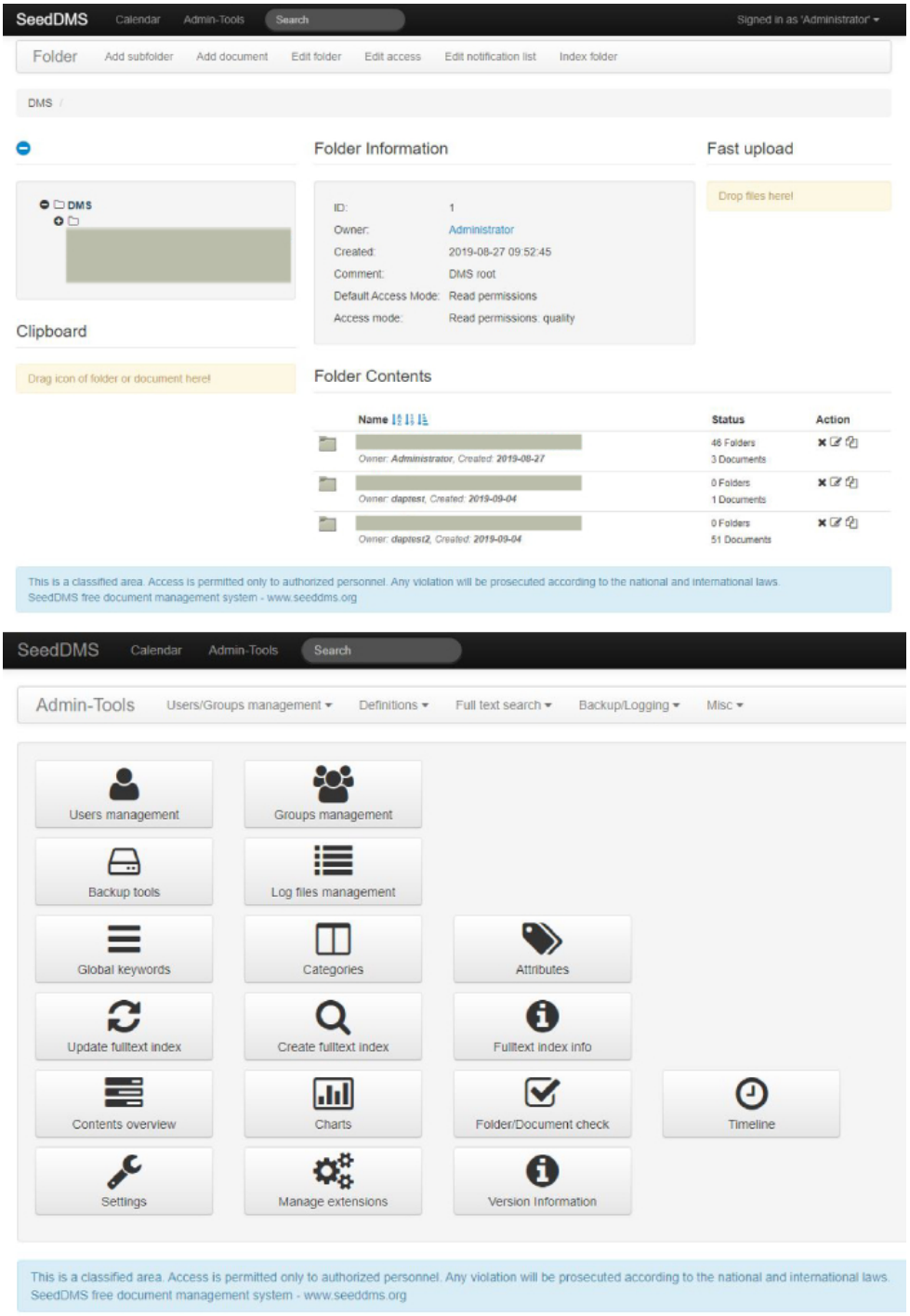


Figure A.1: Screenshots from the deployed SeedDMS webserver

A.2 Mayan-EDMS

Mayan-EDMS is an open-source web-based DMS coded in Python using the Django framework. All functionalities are readily available in its free and main public version. It has an active online community of volunteers including coders, testers, donors, and third-party service and support providers. According to its developer, Roberto Rosario, the largest repository collected in Mayan-EDMS was for the Puerto Rico's main permit agency (20,000+ documents). It is designed to be easy to migrate to different physical computers and this is clearly visible from its easy installation methods and the presence of proper documentation. Mayan-EDMS uses its own database search framework to perform full text search and full text indexing. In the case of our installation, that was using PostgreSQL full text search.

The installation is done by pulling a pre-configured Docker container with all software dependencies and Python packages installed. This container is available under the name mayanedms/mayanedms at the docker-hub website. The container is then linked to another container where the database is installed. In this case, PostgreSQL (v9.6) was used and the entry credentials are set in the initial run commands by which the dockers where deployed. The following are the sets of commands used to pull and deploy the containers using docker (these were collected from the official installation procedures).

```
$ docker pull mayanedms/mayanedms:latest
$ docker pull postgres:9.6

$ docker run -d \
  --name mayan-edms-postgres \
  --restart=always \
  -p 5432:5432 \
  -e POSTGRES_USER=mayan \
  -e POSTGRES_DB=mayan \
  -e POSTGRES_PASSWORD=mayanuserpass \
  -v /docker-volumes/mayan-edms/postgres: \
  /var/lib/postgresql/data \
  -d postgres:9.6
```

```
$ docker run -d \
  --name mayan-edms \
  --restart=always \
  -p 80:8000 \
  -e MAYAN_DATABASE_ENGINE= \
  django.db.backends.postgresql \
  -e MAYAN_DATABASE_HOST=172.17.0.1 \
  -e MAYAN_DATABASE_NAME=mayan \
  -e MAYAN_DATABASE_PASSWORD=mayanuserpass \
  -e MAYAN_DATABASE_USER=mayan \
  -e MAYAN_DATABASE_CONN_MAX_AGE=0 \
  -v /docker-volumes/mayan-edms/media:/var/lib/mayan \
  mayanedms/mayanedms:latest
```

The *docker run* command for the postgres container is set to operate on the default 5432 port on both the container and the host PC. The initial database is created using the -e as it sets the environment variables for the containers. The -v option binds a directory to mount its contents to the container. The -d option runs the container in background and prints the container ID for the user. The default IP for docker is 172.17.0.1 which is used by the Mayan-EDMS container to bind the DMS to the database. An external directory is also mounted to allow for bulk data import and both containers are ran. The *–restart=always* option guarantees that the containers are deployed when the docker is on (it is set to be always on when the hosting PC is on too). More information is available in the documentatin for the EDMS https://docs.mayan-edms.com/index.html.

Other installation options include creating a custom Python virtual environment and manually installing the required packages and software. This is a great option for those with prior knowledge with databases and Python, however, Docker containers are also dependable and flexible when it comes to resource management and communication between containers.

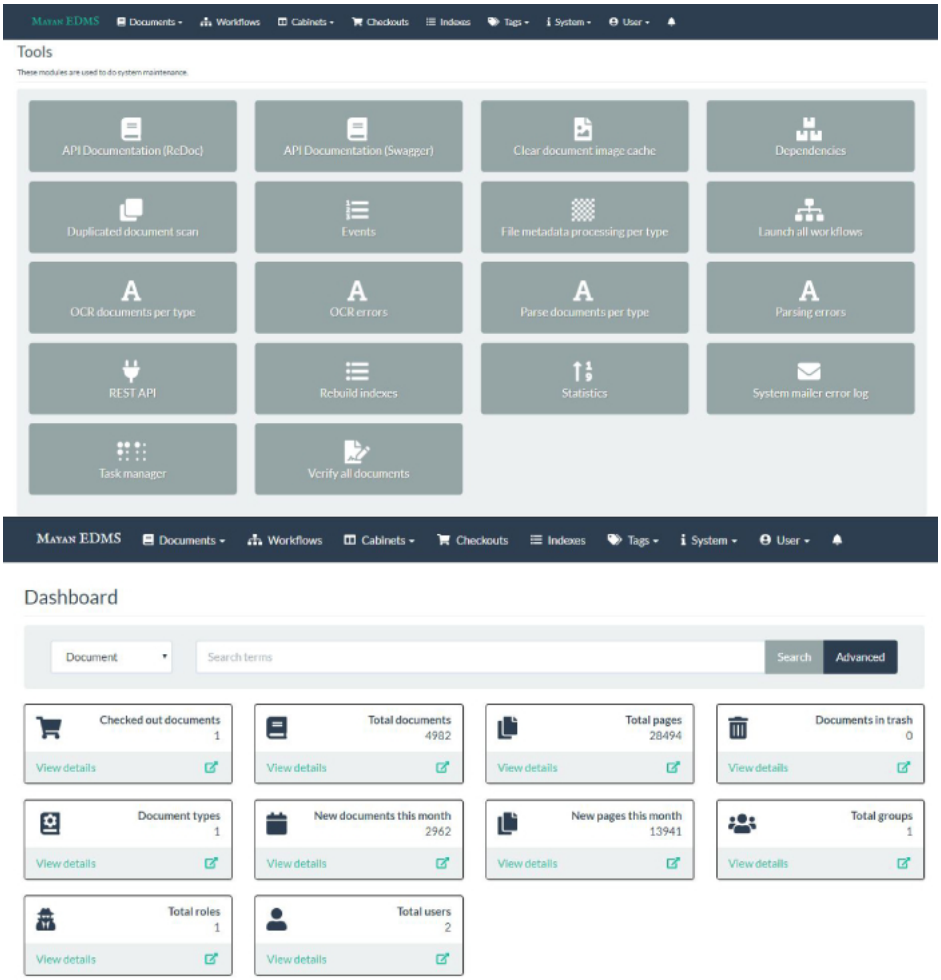


Figure A.2: Screenshots from the deployed Mayan-EDMS webserver

A.3 OpenKM (Community Edition)

OpenKM is a DMS written in Java that provides 2 versions, the Enterprise version with proprietary license and the Community version which is open source with GNU GPL license. OpenKM uses the Lucene library to build their own search framework to allow for full text search and information retrieval. The DMS provides a web interface for managing nonspe-

cific files but the indexing procedure is only available for documents with certain extensions, that is, documents that can be edited with usual document editing software.

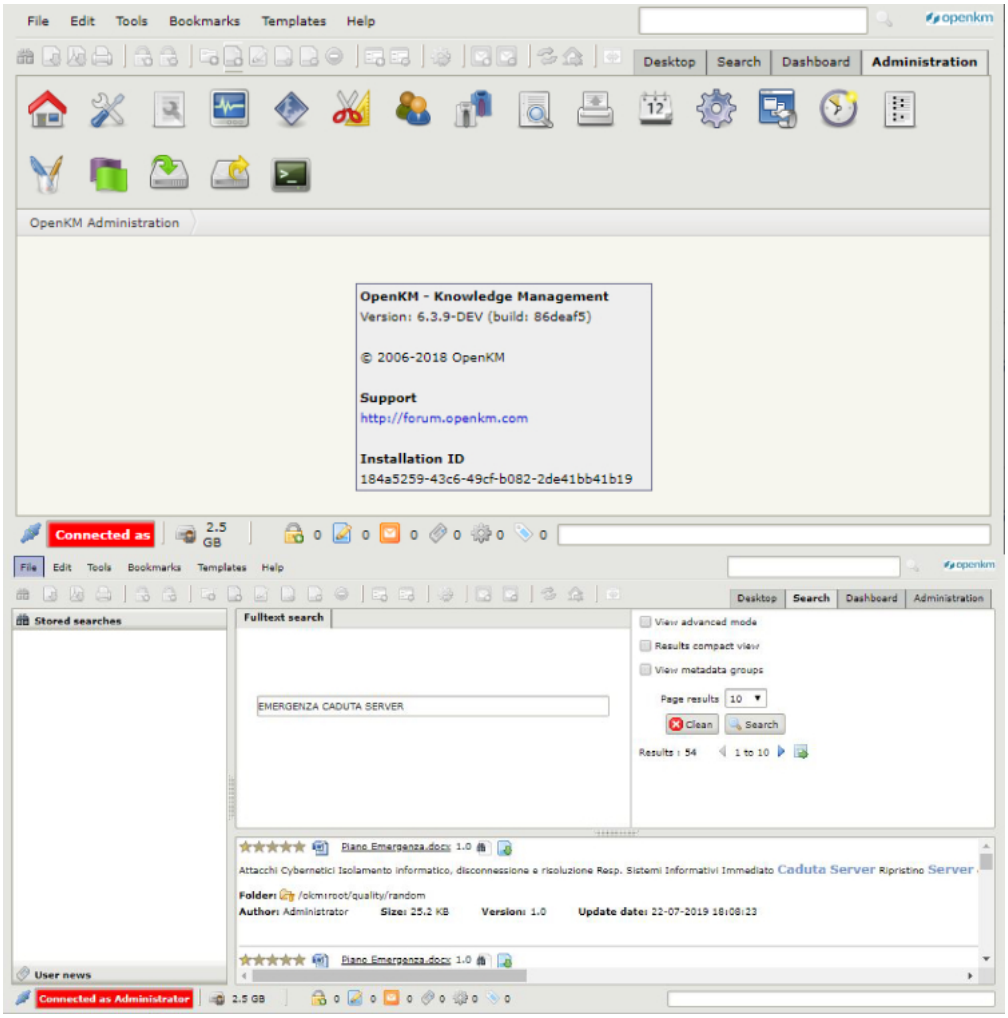


Figure A.3: Screenshots from the deployed OpenKM webserver

There are several methods by which OpenKM can be installed. The best of which for production purposes is the full installation, similar to how SeedDMS was installed. This method starts by creating a user for OpenKM installation and installing the latest stable OpenJDK Runtime Environment (1.8.0_212). By default, this environment would be set to as the default Java, otherwise, it should be done manually. MySQL server was selected for creating the database. A database user is created and the is granted the privileges for the OpenKM tables (database is created). The Tomcat installation bundle is downloaded from the main website. At the time of writing, the tested community edition version was 6.3.2. Tomcat should be configured as a service, by creating and running a configuration script at the */etc/init.d*

directory.

Third-party software are installed to allow for proper functionalities such as document preview, indexing, and OCR. LibreOffice, Tesseract, Clamav, Imagemagick, and ghostscript are therefore installed. The Tomcat service is then started and the webserver is deployed and tested after importing the document repository.

Other installation methods include using a pre-built installer written in java that does all for the user, however to customize the installation, it might get a bit tedious to do so this way. Also, another viable way is to download a pre-configured VM ovf file and deploy the webserver from there. This is more applicable for testing before going into production stage. It can also be used in production stage if the enterprise in use is a small one.

For more information regarding the app alongside with links for all other requirements, please refer to `https://docs.openkm.com/kcenter/view/okm-6.3-com/`

A.4 Alfresco (Community Edition)

Alfresco is a collection of information management software including a content management system that can be used as a DMS due to its extensive functionalities. The software is written in Java and uses Apache Solr as its search framework. For this application, the open source community edition of Alfresco Content Services was used. This version has an installer for both UNIX and Windows based systems. The installer was straight forward to use, the user only had to make the downloaded file to be executable using the *chmod* command. The installer leads through all the steps with reasonable flexibility in deciding the mount points and database credentials.

Tomcat (Apache) server was used to host the Java app. The installer creates a directory where all the dependencies are installed alongside the hosted webserver. The installer creates a database in PostgreSQL and the automatically sets the entry credentials in the *alfresco-global.properties* file. This file can be found in the *{ALFRESCO_HOME}/tomcat/shared/classes* directory. This file is home for all other configurations, including external modules and addons needed for other functionalities. No further input from the user is required for the main installation [Alfresco-Services, 2017].

Further plugins where added to enhance the functionalities of the DMS to match the requirements of the hosting institution. Namely, an OCR plugin, a plugin allowing the preview of AutoCAD files directly from the DMS (browser), and a login manager plugin to enable fast transition between user's points of view. These were installed using the module directories.

The OCR plugin is called *Simple OCR action for Alfresco* and is documented in https://github.com/keensoft/alfresco-simple-ocr. The installation procedure starts by installing the ImageMagick package and tesseract-ocr for the languages wanted in the application. This is followed by installing the *OCRmyPDF* package for the OS running the webserver. The source code for the OCR action is downloaded and built using the maven builder command (*mvn clean package*). The output *Repo JAR* is added to the *{ALFRESCO_HOME}/modules/platform* directory while the *Share JAR* is placed in the *{ALFRESCO_HOME}/modules/share* directory. The *alfresco-global.properties* file is then edited to have the configuration needed to run the module. When using OCRmyPDF, the following settings are the recommended addition.

```
ocr.command=/usr/local/bin/ocrmypdf
ocr.output.verbose=true
ocr.output.file.prefix.command=
ocr.extra.commands= verbose 1 force ocr l ita+eng+fra
ocr.server.os=linux
```

The *User Account Switcher* plugin (https://github.com/Aimprosoft/user-account-switcher) was also added to the DMS. The same principle for building the JAR file was used here as well. However, the Repo JAR was added to the *{ALFRESCO_HOME}/tomcat/webapps/alfresco/WEB-INF/lib* directory while the Share JAR was added to the *{ALFRESCO_HOME}/tomcat/webapps/share/WEB-INF/lib* directory.

The *AutoCAD-Viewer* plugin https://github.com/EisenVault/Alfresco-Autocad-Viewer was an *amp* file added to the *{ALFRESCO_HOME}/amps* directory. The amp file was applied to *WAR* file containing the webserver using the *apply_amps.sh* script found in the main *bin* directory of the installation. A trial version of *QCAD* was also installed in the hosting machine as it is a dependency of the addon. The *alfresco-global.properties* was edited and the following settings were added.

```
dwg2pdf.root=/home/{user}/opt/qcad {version} trial linux x86_64
content.transformer.dwg2pdf.priority=50
content.transformer.dwg2pdf.extensions.dwg.pdf.supported=true
content.transformer.dwg2pdf.extensions.dwg.pdf.priority=50
content.transformer.dxf2pdf.priority=50
content.transformer.dxf2pdf.extensions.dxf.pdf.supported=true
content.transformer.dxf2pdf.extensions.dxf.pdf.priority=50
```

More information on Alfresco Content Service as well as manuals, documentations, and download links are accessible on their website at https://docs.alfresco.com

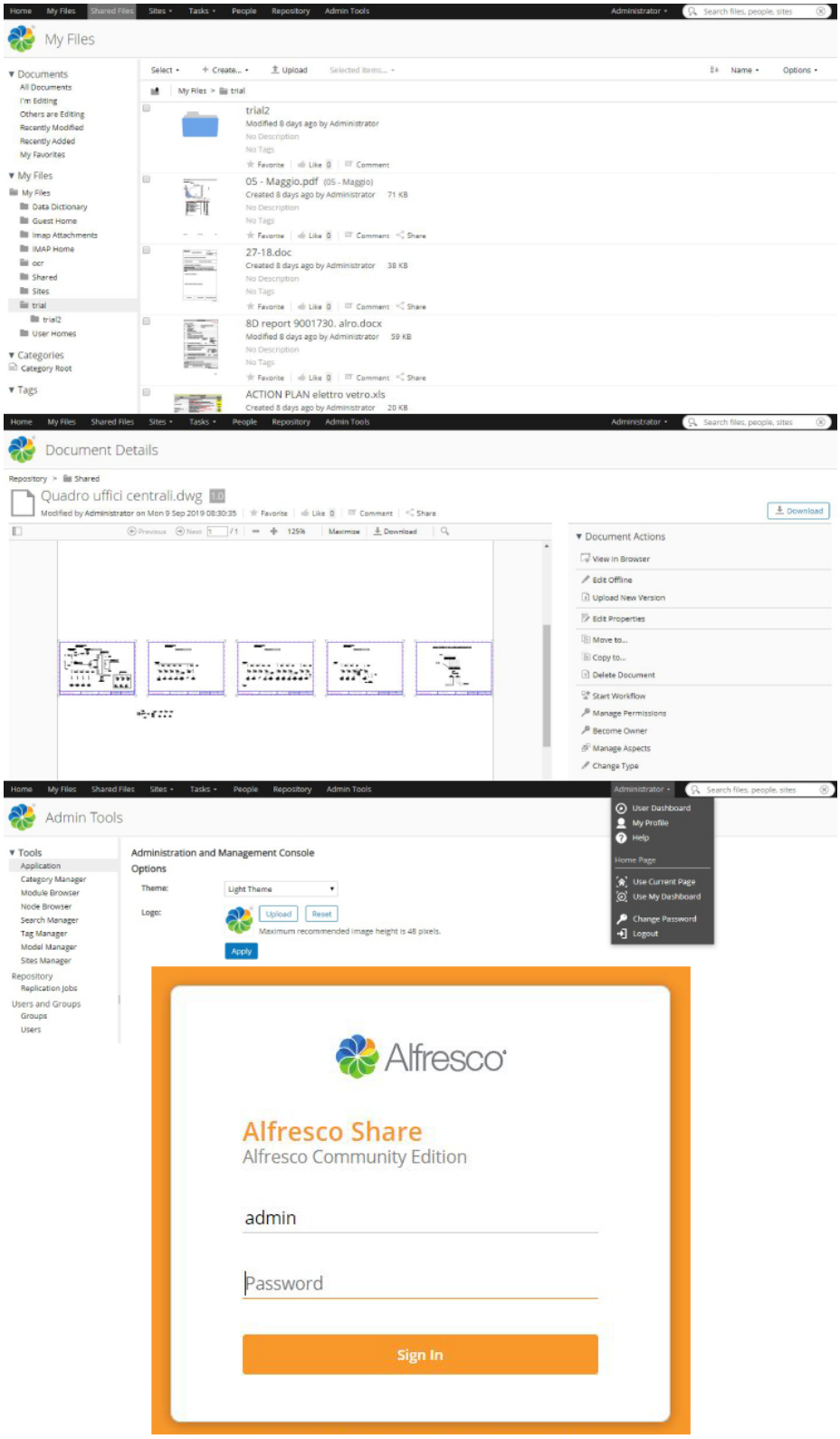


Figure A.4: Screenshots from the deployed Alfresco CE webserver

A.5 NextCloud with ElasticSearch

NextCloud is an open source self hosting cloud service that is considered to be one of the most favourable Content Management Systems available. It allows the user the capability of creating and using file hosting services. This is due to the extensive variety of plugins available to it. The software is written in PHP and JavaScript and is freely available with all of its stable versions. The latest version is NextCloud 15, however, for our functionalities, we used NextCloud 13, a more stable version with the ElasticSearch (ES) plugin to allow for document indexing and full text search.

The stable Nextcloud 13 package is downloaded from the main github repository. Apache2 is installed and is given access in the host machine's firewall. For this application, PHP version 7.2 was used alongside the libapache2-mod-php php-common php-gmp php-curl php-intl php-mbstring php-xmlrpc php-mysql php-gd php-xml php-cli php-zip modules. The hosting database is initiated on an MySQL server and the control privileges are granted to a NextCloud user. The Composer tool was then installed to allow for dependency management in PHP. Using the standard *sudo composer install* command, installation procedure is started.

A *nextcloud.conf* configuration file for Apache is created in the */etc/apache2/sites-available* directory. This is done to allow the hosting of the website. Default configuration data is kept untouched and the configuration file is filled with the following.

```
Alias /nextcloud "/var/www/nextcloud/"
<Directory /var/www/nextcloud/>
  Options +FollowSymlinks
  AllowOverride All
 <IfModule mod_dav.c>
  Dav off
 </IfModule>
 SetEnv HOME /var/www/nextcloud
 SetEnv HTTP_HOME /var/www/nextcloud
</Directory>
```

Lastly, the following Apache modules are enabled for proper functionalities of the NextCloud service.

```
sudo a2ensite nextcloud.conf
sudo a2enmod rewrite
```

```
sudo a2enmod headers
sudo a2enmod env
sudo a2enmod dir
sudo a2enmod mime
```

ES needs to be installed separately as a plugin. The installation starts by installing the default Java environment, Jre-8. The user must add the repositories for downloading the ES version compatible with the Linux build they are running followed by the installation command. The ES settings must by configured to perform the searching procedures locally and thus the *network.host* address in the */etc/elasticsearch/elasticsearch.yml* file should be set to the loop back address 127.0.0.1. The default port used by ES is 9200. and a test command "curl -XGET '127.0.0.1:9200/?pretty'" can be run to find the test document with the tagline "You Know, for Search".

Once the ES installation is verified, the ingest-attachment plugin is installed to allow the search of input documents in the to be deployed NextCloud webserver.

```
$ sudo /usr/share/elasticsearch/bin/elasticsearch-plugin \
  install ingest-attachment
```

As in OpenKM, the OCR service is performed by the third party software Tesseract. The user is given the option to download OCR packages for a wide variety of languages, for our application, the English and Italian language packages were installed.

The indexing procedure is started once the documents are imported into the system by the users. This process can be automated, but for testing the procedure, it can also be done manually by running the PHP code for ES.

```
$ sudo -u www-data php /var/www/html/nextcloud/occ \
fulltextsearch:index
```

For any further information, the documentation website can be accessed at https://docs.nextcloud.com/.

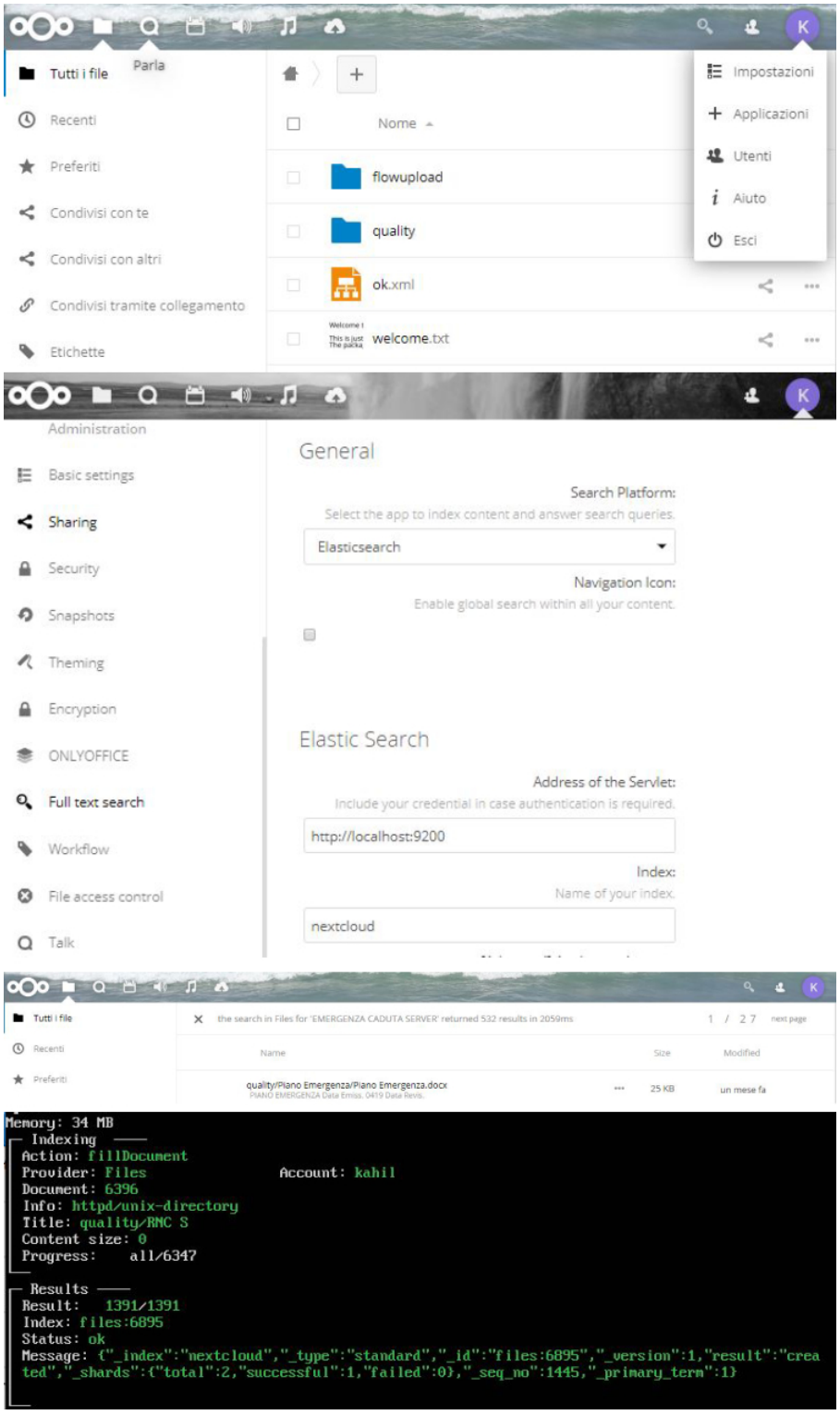


Figure A.5: Screenshots from the deployed NextCloud webserver with the ElasticSearch plugin

A.6 OpenSearchServer

OpenSearchServer (OSS) application is written in Java. Java 7 or higher is required to run the app. Installing OSS was very straight forward as the developers have already created a debian package that can be downloaded and executed. From their website http://www.opensearchserver.com/documentation/, the proper documentation can be extracted. The installation is started by depackaging the *deb* file using the *dpkg -i* command. OSS is configured as a service and can be controlled likewise (*service opensearchserver start*). Port 9090 is used to host the service. After importation of the documents, indexing can be initiated and the search service can be used afterwards.

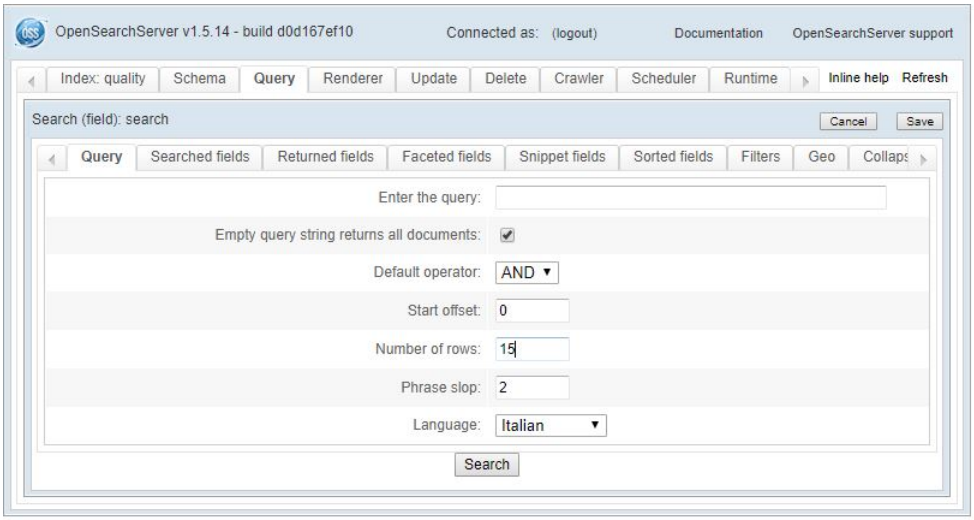


Figure A.6: A screenshot from the deployed OpenSearchServer webserver.

Bibliography

[Ahmad et al., 2017] Ahmad, H., Bazlamit, I., & Ayoush, M. (2017). Investigation of document management systems in small size construction companies in jordan. volume 182.

[Alfresco-Services, 2017] Alfresco-Services (2017). Alfresco content services 5.2.6.

[Alshibly et al., 2016] Alshibly, H., Chiong, R., & Bao, Y. (2016). Investigating the critical success factors for implementing electronic document management systems in governments: Evidence from jordan. *Information Systems Management*, accepted.

[Apache-Software-Foundation, 2017] Apache-Software-Foundation (2017). Apache solr reference guide.

[Bondarenko et al., 2018] Bondarenko, A., Hagen, M., Völske, M., Stein, B., Panchenko, A., & Biemann, C. (2018). Webis at trec 2018: Common core track. In *TREC*.

[Chapelle et al., 2009] Chapelle, O., Metlzer, D., Zhang, Y., & Grinspan, P. (2009). : (pp. 621–630).

[Choi et al., 2018] Choi, W., Jo, S.-H., & Lee, K.-S. (2018). Cbnu at trec 2018 incident streams track. In *TREC*.

[Cooper, 1970] Cooper, W. S. (1970). On deriving design equations for information retrieval systems. *Journal of the American Society for Information Science*, 21(6), 385–395.

[Dietz et al., 2017] Dietz, L., Verma, M., Radlinski, F., & Craswell, N. (2017). Trec complex answer retrieval overview. In *TREC*.

[Dizon et al., 2017] Dizon, P. J. C., Jacob, M. J. Y., Ko, J. B., Reyes, J. J. D., Domingo, M. J., & Rodriguez, E. V. (2017). University of santo tomas faculty of medicine document management system (mdms). In *2017 Manila International Conference on "Trends in Engineering and Technology"(MTET-17)*.

[ElShobaki, 2017] ElShobaki, A. (2017). Impact of Senior Management Support in the Success of electronic Document Management Systems. *International Journal of Engineering and Information Systems (IJEAIS)*, 1(4), 47 – 63.

[Ferro & Peters, 2019] Ferro, N. & Peters, C. (2019). *Information Retrieval Evaluation in a Changing World Lessons Learned from 20 Years of CLEF: Lessons Learned from 20 Years of CLEF*.

[Gormley & Tong, 2015] Gormley, C. & Tong, Z. (2015). *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc., 1st edition.

[Harman, 2011] Harman, D. (2011). *Information Retrieval Evaluation*, volume 3.

[Hernad & Gaya, 2013] Hernad, J. M. C. & Gaya, C. G. (2013). Methodology for implementing document management systems to support iso 9001:2008 quality management systems. *Procedia Engineering*, 63, 29 – 35. The Manufacturing Engineering Society International Conference, MESIC 2013.

[ISO 15489-1:2016, 2016] ISO 15489-1:2016 (2016). *Information and documentation — Records management — Part 1: Concepts and principles*. Standard, International Organization for Standardization, Geneva, CH.

[ISO 23950:1998, 1998] ISO 23950:1998 (1998). *Information and documentation — Information retrieval (Z39.50) — Application service definition and protocol specification*. Standard, International Organization for Standardization, Geneva, CH.

[ISO 9001:2015, 2015] ISO 9001:2015 (2015). *Quality management systems — Requirements*. Standard, International Organization for Standardization, Geneva, CH.

[Jansen & Rieh, 2010] Jansen, B. J. & Rieh, S. Y. (2010). The seventeen theoretical constructs of information searching and information retrieval. *Journal of the Association for Information Science Technology*, 61(8), 1517–1534.

[Järvelin & Kekäläinen, 2002] Järvelin, K. & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4), 422–446.

[Joseph et al., 2013] Joseph, P., Debowski, S., & Goldschmidt, P. (2013). Search behaviour in electronic document and records management systems: an exploratory investigation and model. *INFORMATION RESEARCH-AN INTERNATIONAL ELECTRONIC JOURNAL*, 18(1), 1–38.

[Keen, 1967] Keen, M. (1967). *Evaluation Parameters*.

[Kruschwitz & Hull, 2017] Kruschwitz, U. & Hull, C. (2017). Searching the enterprise. *Foundations and Trends® in Information Retrieval*, 11(1), 1–142.

[Lin et al., 2017] Lin, J., Mohammed, S., Sequiera, R., Tan, L., Ghelani, N., Abualsaud, M., McCreadie, R., Milajevs, D., & Voorhees, E. M. (2017). Overview of the trec 2017 real-time summarization track. In *TREC*.

[López-García et al., 2017] López-García, P., Oleynik, M., Kasác, Z., & Schulz, S. (2017). Trec 2017 precision medicine-medical university of graz. In *TREC*.

[Manning et al., 2008] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

[Misutka & Galambos, 2008] Misutka, J. & Galambos, L. (2008). Extending full text search engine for mathematical content.

[Regazzi, 1980] Regazzi, J. (1980). Evaluating indexing systems: a review after cranfield. *The Indexer.*, 12.

[Salton, 1971] Salton, G. (1971). The smart system. *Retrieval Results and Future Plans*.

[Shahi, 2015] Shahi, D. (2015). *Apache Solr: A Practical Approach to Enterprise Search*. Berkely, CA, USA: Apress, 1st edition.

[Snaidero, 2018] Snaidero, B. (2018). Sql server full text indexes.

[Sprague, 1995] Sprague, R. H. (1995). Electronic document management: Challenges and opportunities for information systems managers. *MIS Quarterly*, 19(1), 29–49.

[Ugale et al., 2017] Ugale, M. K., Patil, S. J., & Musande, V. B. (2017). Document management system: A notion towards paperless office. In *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)* (pp. 217–224).

[Zend, 2016] Zend (2016). Manual - documentation.

[Zhang & Huang, 2019] Zhang, Y. & Huang, M. (2019). Overview of the ntcir-14 short text generation subtask: Emotion generation challenge. In *Proceedings of the 14th NTCIR Conference*.

Acknowledgments

Give credit when credit is due,

This dissertation project wouldn't have been possible without the help and support of my supervisor, Professor Nicola Ferro, and the good people at De Angeli Prodotti. I would like to thank Dr. Ezio Fantinato, Dr. Michele Pellegrini, and Dr. Nicola Lazzaro for the help they provided throughout this project.