An Information Extraction Model for Recommending the Most Applied Case

Author: Thashen Padayachy

Student Number: 211228729

Supervisor: Prof Brenda Scholtz

Masters Dissertation

Submitted in fulfilment of the requirements for the degree of Master of Science in Computer Science and Information Systems at the Nelson Mandela University

April 2019

Acknowledgements

My academic career at the Nelson Mandela University has been a very interesting and fulfilling experience. Only few know that the original plan was for me to stop studying after obtaining my undergraduate degree. This plan became even more set-in stone when finances became dire. One always hears about good people out there but not many get to meet these people or have them play an important role in their lives. Prof Brenda Scholtz is without a doubt one of the good people!

Ma'm, you were a necessary disruption of the plan as you made it possible for me to be where I am today. Working with you for the last five years has been an incredible learning experience. You constantly refined my abilities as a researcher and supported me since the beginning. I am forever grateful for all that you have done for me and my parents have the utmost respect and appreciation for all that you have provided me. We are truly thankful that you came into our lives!

To my parents, thank you for the constant support throughout my Masters degree. It is through the values you instilled in me from a young age that helped me remain focused and give of my best.

Thank you to Ashlynne and all my friends who supported me throughout the last 2 years of my Masters.

I would like to thank the National Research Foundation and LexisNexis for providing me with the funding to complete my Masters.

Abstract

The amount of information produced by different domains is constantly increasing. One domain that particularly produces large amounts of information is the legal domain, where information is mainly used for research purposes. However, too much time is spent by legal researchers on searching for useful information. Information is found by using special search engines or by consulting hard copies of legal literature.

The main research question that this study addressed is "*What techniques can be incorporated into a model that recommends the most applied case for a field of law?*". The Design Science Research (DSR) methodology was used to address the research objectives. The model developed is the theoretical contribution produced from following the DSR methodology.

A case study organisation, called LexisNexis, was to help investigate the real-world problem. The initial investigation into the real-world problem revealed that too much time is spent on searching for the Most Applied Case (MAC) and no formal or automated processes were used. An analysis of an informal process followed by legal researchers enabled the identification of different concepts that could be combined to create a prescriptive model to recommend the MAC.

A critical analysis of the literature was conducted to obtain a better understanding of the legal domain and the techniques that can be applied to assist with problems faced in this domain, related to information retrieval and extraction. This resulted in the creation of an IE Model based only on theory. Questionnaires were sent to experts to obtain a further understanding of the legal domain, highlight problems faced, and identify which attributes of a legal case can be used to help recommend the MAC. During the Design and Development activity of the DSR methodology, a prescriptive MAC Model for recommending the MAC was created based on findings from the literature review and questionnaires. The MAC Model consists of processes concerning:

- Information retrieval (IR);
- Information extraction (IE);
- Information storage; and
- Query-independent ranking.

Analysis of IR and IE helped to identify problems experienced when processing text. Furthermore, appropriate techniques and algorithms were identified that can process legal documents and extract specific facts. The extracted facts were then further processed to allow for storage and processing by query-independent ranking algorithms.

The processes incorporated into the model were then used to create a proof-of-concept prototype called the IE Prototype. The IE Prototype implements two processes called the IE process and the Database process. The IE process analyses different sections of a legal case to extract specific facts. The Database process then ensures that the extracted facts are stored in a document database for future querying purposes.

The IE Prototype was evaluated using the technical risk and efficacy strategy from the Framework for Evaluation of Design Science. Both formative and summative evaluations were conducted. Formative evaluations were conducted to identify functional issues of the prototype whilst summative evaluations made use of real-world legal cases to test the prototype. Multiple experiments were conducted on legal cases, known as source cases, that resulted in facts from the source cases being extracted. For the purpose of the experiments, the term "source case" was used to distinguish between a legal case in its entirety and a legal case's list of cases referred to. Two types of NoSQL databases were investigated for implementation namely, a graph database and a document database.

Setting up the graph database required little time. However, development issues prevented the graph database from being successfully implemented in the proof-of-concept prototype. A document database was successfully implemented as an alternative for the proof-of-concept prototype.

Analysis of the source cases used to evaluate the IE Prototype revealed that 96% of the source cases were categorised as being partially extracted. The results also revealed that the IE Prototype was capable of processing large amounts of source cases at a given time.

Declaration

I, Thashen Padayachy, hereby declare the dissertation, ***An Information Extraction Model for Recommending the Most Applied Case***, for Magister Scientae in Computer Science and Information Systems is my own independent work. All sources used or quoted have been indicated and acknowledged by means of complete references. This dissertation has not been previously submitted for assessment to any other university or completion of any other qualification.

Thashen Padayachy

List of Acronyms

AI	Artificial Intelligence
ALL SA	All South African
ALL SA	All South African
BSON	Binary JSON
CO	Co-Reference Resolution
CRT	Case Referred To
DFA	Deterministic Finite Automaton
DSR	Design Science Research
EE	Event Extraction
FEDS	The Framework for Evaluation in Design Science Research
GATE	General Architecture for Text Engineering
HITS	Hyper-Link Induced Topic Search
IDE	Integrated Development Environment
IE	Information Extraction
IR	Information Retrieval
IRP	Irrelevant Pages
IT	Information Technology
JSON	Javascript Option Notation
MAC	Most Applied Case
NER	Named Entity Recognition
NFA	Non-Deterministic Finite Automaton
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NoSQL	Not Only SQL
PDF	Portable Document Format
POS	Part-of-speech
R	Relevant
RE	Relation Extraction
Ref-To-Case	Referred to Case
RO	Research Objective
ROI	Return on Investment
RQ	Research Question
Skmeans	Spherical kmeans
TF	Term Frequency
VR	Very Relevant
WR	Weakly Relevant
XML	Extensible Markup Language

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

Over the years the value and dependency of information has become important resulting in information explosion (Ifijeh, 2010). Information is present in various forms of media and consists of data, facts, and ideas. The types of media that contain information include printed documents and documents in electronic format. Information explosion refers to a major increase in the supply of information to users (White, 2009). Katz (2002) states that the Internet has contributed greatly to information explosion. The amount of information generated is predicted to increase from 4.4 zettabytes to 44 zettabytes by 2020 (Khaso, 2016). Although there is an abundance of information available due to information explosion, retrieving useful information is not always easy. Factors that affect the quality of information retrieved are retrieval models, web search, and user modelling (University of Massachusetts, 2002). Multiple retrieval models have been created to cater for tasks such as describing a document's content and structure. However, more comprehensive retrieval models are required to incorporate the evolving information needs of users and to use less computation. Search engines provide accurate results to users' queries, but users are generally not looking for only a single page. To improve web searching, aspects such as web structure, crawling and indexing must be investigated.

To return valuable information to users, support for Information Retrieval (IR) needs to be provided (Roshdi & Roohparvar, 2015). IR facilitates various facets of data such as representation and consists of many intermediate stages and processes. Further processing of information returned by IR is possible by means of Information Extraction (IE). IE is the process of extracting facts from sources of text that can be unstructured, semi-structured, or structured (Jiang, 2012). Three processes, namely extracting, integrating, and translating facts to output are performed by IE that use a particular task and IE technique. The task and technique chosen depends on the user's goal and the source of text to be used.

A technique of IE is the use of web scrapers to automatically search for and extract specific information from a website (Vargiu & Urru, 2012). Web scrapers can be created using libraries, frameworks, or desktop-based applications. In-depth analysis of information extracted from IE can be obtained by applying Natural Language Processing (NLP). NLP is used to analyse natural language and perform tasks based on the analysis (Chowdhury, 2003). NLP is made up of different phases and tasks. Phases include morphological and syntactic analysis whilst tasks include Part-of-Speech tagging (POS) and chunking. Another technique for IE is the application of regular expressions. Regular expressions are patterns applied to manipulate text and simplify text processing (Goyvaerts & Levithan, 2009). In addition to information being processed, information must also be stored for retrieval, querying, and to avoid reprocessing of processed information. Standard relational databases can be used to store information, but a more efficient method would be to use a NoSQL database such as a graph or document database. Graph databases use graphs to store information in nodes and allow for relationships to be created between nodes using edges. Advantages of graph databases include performance and flexibility (Robinson, Webber, & Eifrem, 2015). Document databases allow for data to be stored in the form of documents. Document databases also support embedding of data into documents and require no schema (MongoDB, 2018c). Once information has been stored, additional processing can occur such as query-independent ranking.

A domain that is particularly affected by information explosion is the legal domain. Any new case that goes to court increases the body of knowledge that legal practitioners use (Marr, 2016). This

knowledge is commonly used for precedents. Marr (2016) further states that the legal domain's data is mainly used for research and stored in massive databases. Access to the data is only possible through a search engine-like system called LegalCitator.

A combination of IR, IE, information storage, and query-independent ranking can aid legal researchers who are involved in court cases. Various factors must be considered when advocating a court case or deciding on a sentence (LAW.gov, 2016). Due to the time sensitivity of each case it is important for lawyers to access the Most Applied Case (MAC) so that they can access relevant information quickly to strengthen their argument and improve their chances of winning a legal dispute. The MAC refers to a case that is the most useful and commonly used case for a field of law. The act of using a previous case to strengthen or win an argument is known as a precedent (Black, Nolan, Nolan-Haley, Hicks, & Brandi, 1990). Different studies ranging from artificial intelligence (AI), IR, and rule-based systems have been conducted within the legal domain. However, no studies have been conducted that aid in recommending the MAC for a field of law. Legal citations can aid in locating legal cases, and more specifically, the MAC. A legal citation refers to legal authorities or precedents within a legal dispute to help strengthen a case (Black et al., 1990). Legal citations eliminate the need to write out long references by using abbreviations.

The real-world problem of this study is that legal organisations struggle to obtain accurate and useful information related to the MAC for a field of law. LexisNexis is one such organisation that provides legal and risk services to companies and government agencies globally (LexisNexis, 2017a). Experts from LexisNexis revealed that they require techniques to recommend the MAC for a field of law. These techniques can reduce the amount of time spent on searching for important cases. LexisNexis' existing system, LegalCitator, stores vast amounts of data relating to Case Law and Legislation from various African countries and makes use of elementary searching techniques provided by the Elasticsearch search engine. LegalCitator highlights the importance of a case by means of a signal and provides a summary of the case's judgement. A case can only receive one of six signals. A case's judgement includes aspects such as judgement details, subject index, and judgment history. LegalCitator has no built in AI and does not use any IE techniques.

1.2 Problem Statement

Legal practitioners require fast and efficient access to information regarding precedents. This access can assist lawyers to strengthen their case as courts base decisions on principles established in prior cases (Black et al., 1990). Precedents are referenced by means of legal citation. Returning relevant information can be challenging as information must be accurately processed (Ikonomakis, Kotsiantis, & Tampakas, 2005).

Legal practitioners at LexisNexis and users of LexisNexis' LegalCitator product currently spend a large amount of time searching for cases to use as precedents in their legal disputes. Experts at LexisNexis reported that they require their LegalCitator system to cater for functions that recommend the MAC for a field of law. The experts further revealed that reducing search time will allow users to focus on other aspects of a legal dispute and improve the value of the LegalCitator system. The research problem within the legal domain is to determine how legal case documents can be processed using various techniques to suggest the MAC (Young, 2010). Appendix A visualises the real-world research problem at LexisNexis whilst the answered questionnaires given to LexisNexis can be seen in Appendix B.

1.3 Aim and Scope of Research

The aim of this study is to design a prescriptive model of techniques and algorithms that recommend the MAC for a field of law to a legal researcher. A model in Design Science Research is used to depict

a problem within its solution space. More specifically, a prescriptive model is used to provide descriptions of possible future solutions and aid in constructing artefacts.

Due to the large scope in the field of law, the study will focus on the accurate retrieval and extraction of text found in legal case documents pertaining to all fields of law in the All South African (ALL SA) legal journals for the period 1996 to 2018 from the South African division of LexisNexis. The study will not focus on any visualisation techniques. The implementation and testing of results from the various approaches will be limited to the data received from LexisNexis.

1.4 Relevance and Envisaged Contribution

This study will make a theoretical and practical contribution once completed. The theoretical contribution to the body of knowledge will be the combination of techniques to recommend the MAC. The envisaged practical contribution will be the IE model for recommending the MAC. The model will consist of four processes, namely IR, IE, information storage, and query-independent ranking. Once completed, the outcome of the study will be a final proof-of-concept artefact which is a prototype that processes legal cases to extract the facts required for recommending the MAC for a field of law.

1.5 Research Questions

The main research question of this study is:

> ***RQ**: What techniques can be incorporated into a model that recommends the Most Applied Case (MAC) for a field of law?*
>
> ***RQ-Context**: What text processing techniques can be used to process legal cases at LexisNexis?*

In this context research question (RQ-Context) will explore the legal domain and literature related to the research problem.

1.6 Research Objectives

The main research objective (**RO_M**) of this study is:

To develop an information extraction model to recommend the Most Applied Case for a field of law.

The following preliminary subsidiary research objectives for this study are:

> **RO1**: *Identify the problems experienced when processing text as identified by literature and within a real-world context.*
> **RO2**: *Identify the attributes of a court case that can be used to aid in recommending the MAC.*
> **RO3:** *Determine what techniques and algorithms can be used to recommend the MAC.*
> **RO4**: *Identify the criteria that can be used to evaluate the proposed model.*
>
> In this context the term 'processing text' refers to all tasks required to ensure that bodies of text are in the best form to be used in IR models and by IE techniques.

1.7 Research Methodology and Layout of Dissertation

The selected research methodology for this research is the Design Science Research (DSR) methodology (Hevner, March, Park, & Ram, 2004). More detail on DSR and evaluations is provided in Chapter 2 and Chapter 5. The research strategies that will be used in the DSR context in this study are:

- A literature review;
- A case study; and
- The Framework for Evaluation in DSR (FEDS).

To address the problem in this study the DSR methodology, proposed by Hevner (2007) and Peffers, Tuunanen, Rothenberger, and Chatterjee (2007) is used to facilitate the research process. The DSR methodology is used to create an artefact in the form of a model that uses various constructs. To further understand the problem domain, additional questionnaires were sent to experts and a literature review was conducted. The additional questionnaires were used to obtain LexisNexis experts' opinions to derive a set of requirements and research objectives to solve a problem. Requirements are then further derived by means of the literature review and analysis of existing systems. The literature review covered topics such as IR, IE, web scraping, NLP, regular expressions, graph and document databases, and aspects of the legal domain. The set of requirements derived allow for creation of a model that recommends the MAC. The model will be a prescriptive model that provides a solution to recommending the MAC and aids in constructing the system to recommend the MAC (Johannesson & Perjons, 2012). In addition to reducing users' search time, the questionnaires revealed that the solution to the proposed research problem would add value to LexisNexis' product.

Peffers et al. (2007) identify six activities that must be completed when following the DSR methodology, namely:

- A1: Problem identification and motivation;
- A2: Definition of the objectives for a solution;
- A3: Design and development;
- A4: Demonstration;
- A5: Evaluation; and
- A6: Communication.

The first activity, **Problem identification and motivation**, involves identifying a problem that needs to be solved and solutions to the problem. It can be helpful to describe the problem in detail to illustrate how the solution will address the problem's complexity. The second activity, **Define the objectives for a solution**, requires a researcher to determine what a solution will encompass and highlight aspects of the solution that will be possible and feasible. The first and second activities are performed during the relevance cycle whilst the second activity is also performed during the rigor cycle. The third activity, **Design and development**, sees the creation of an artefact that solves the identified problem from the first activity. The artefact can be a construct, model, or method. The fourth activity, **Demonstration**, requires the artefact to be used to illustrate how the artefact solves the identified problem. Demonstration can be conducted in either an experiment, proof of concept, simulation, or case study. The fifth activity, **Evaluation**, involves determining how well the artefact solves the identified problem. Appropriate metrics must be used when evaluating an artefact. The sixth activity, **Communication**, reports on the identified problem's severity and on the usefulness of the artefact. Activities three to six are all performed during the design cycle. Figure 1-1 illustrates the DSR methodology activities.

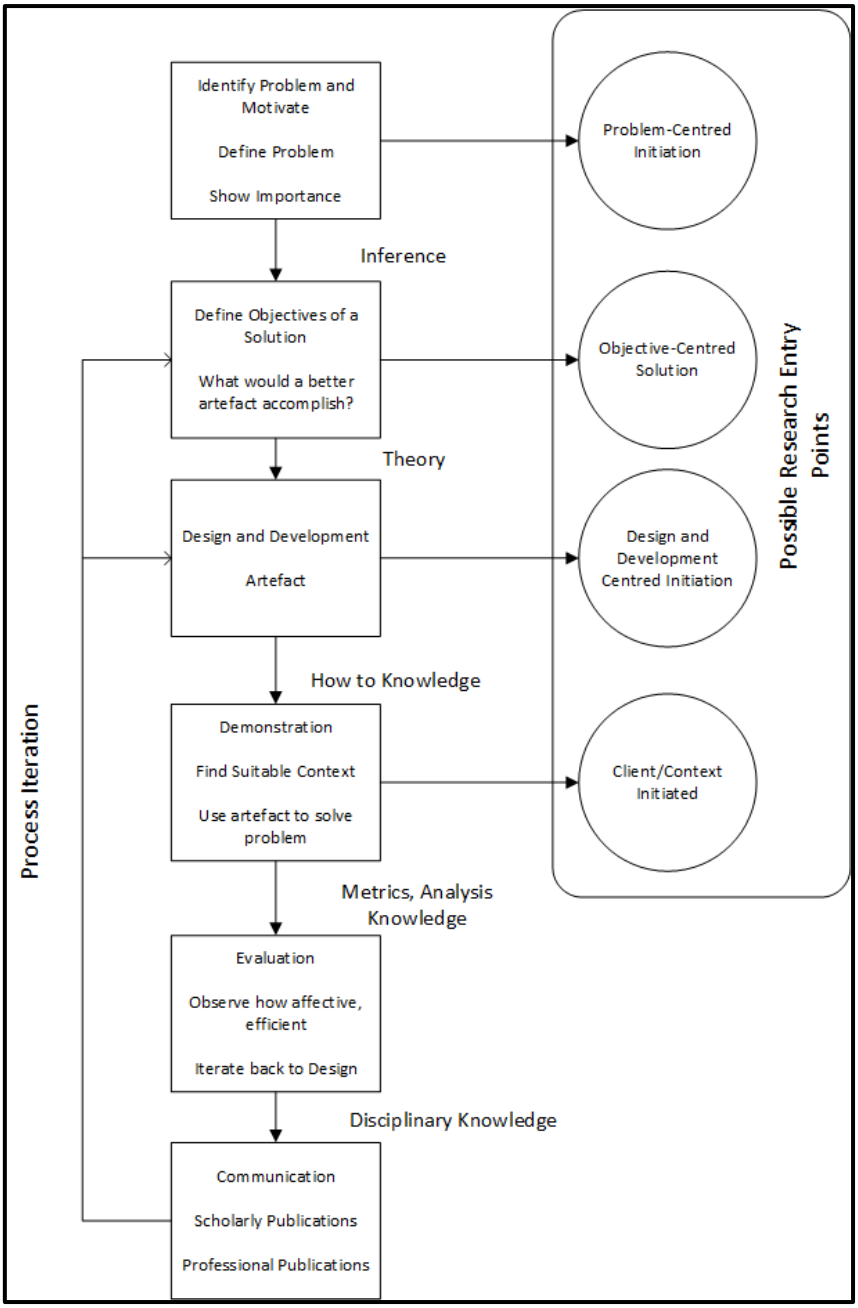


Figure 1-1: DSR Methodology Activities (Peffers et. al,,2007)

The chapters of this research were structured based on the DSR activities. Chapter 1 reported on the first and second activities of the DSR methodology namely, Problem Identification and Motivation, and Definition of Objectives for a Solution. Chapter 1 has introduced the research topic and the reason for conducting the study. The problem statement and research objectives have been stated. The research's scope and constraints, and research methodologies to be used have been identified. High level objectives for a solution and problems in processing text were introduced.

Chapter 2 will continue to report on the second activity of the DSR methodology. Chapter 2 illustrates how the DSR methodology will be used to address the research objectives as well as ethical considerations. Chapter 3 continues to report on the first activity of the DSR methodology. During Chapter 3, a literature review on the techniques that can be applied to the research topic and an investigation of the legal domain will be presented. Chapter 3 will produce an expanded list of

problems in processing text, and provide techniques and algorithms for processing text in the legal domain.

Chapter 4 will continue to report on the second DSR activity and report on findings from the legal domain. Chapter 4 will also present the solution objectives and requirements for a model to be used within the legal domain and in doing so, report on the third DSR activity namely, Design and Development. The solution will be derived from criteria in the literature review. The requirements will be derived from the literature review, findings from the questionnaires, and findings from the analysis of extant systems.

Chapter 5 will continue to report on the third DSR activity and also report on the fourth activity of the DSR methodology namely, Demonstration. Chapter 5 will also present an evaluation plan for the proposed model that will consist of evaluation strategies and methods. The development and evaluation of the prototypes will also be presented in Chapter 5.

Chapter 6 will report on the fifth DSR activity. During Chapter 6, the findings from the evaluations of the prototype will be presented. The findings will then be interpreted to determine the overall success of the prescriptive model. Chapter 7 will report on the final DSR activity namely, Communication in which the conclusion from the research will be presented. Table 1-1 provides a summary of the layout of this dissertation.

Chapter	DSR Activity	Deliverables
1	A1: Problem Identification and Motivation A2: Definition of Objectives for Solution (high level)	High Level Objectives for a Solution Problems in Processing Text -High Level (RO1)
2	-	Research Design Ethical Clearance
3	A1: Problem Identification and Motivation	Expanded List of Problems in Processing Text (RO1 - theoretical) Techniques and Algorithms for Processing Text in the Legal Domain (RO3)
4	A2: Definition of Objectives for Solution A3: Design and Development	Expanded List of Problems in Processing Text (RO1 - practical) Solution Objectives (RO3) Solution Requirements (RO2) Proposed Solution
5	A4: Demonstration A5: Evaluation	Evaluation Plan Developed and Evaluated of Prototypes Solution: 2 Artefacts: MAC Model and IE Prototype (RO_M)
6	A5: Evaluation	Evaluated Prototype (RO4) Findings
7	A6: Communication	Theoretical and Practical Contributions (RO_M)

Table 1-1: Dissertation Layout

Chapter 2: Research Design

2.1 Introduction

The previous chapter provided an overview of the research that will be presented in this dissertation. The aim of this chapter is to report on the research methodology that is applied throughout this research. The research methodology that is used in this research is the DSR methodology (Section 2.2). The DSR methodology follows an iterative three-cycle process that is used to create an artefact (Section 2.3). The DSR methodology will be applied along with other research methods and result in deliverables throughout the research (Section 2.4). To conduct the research, various ethical considerations must be considered by the researcher (Section 2.5).

2.2 Motivation for DSR in This Study

Design science aims to improve the world by creating artefacts that help people meet demands, overcome problems, and take hold of new opportunities (Johannesson & Perjons, 2012). With regards to Information Technology (IT), artefacts can be constructs, models, methods, and instantiations (March & Storey, 2008). Constructs enable the communication and description of problems, solutions, constraints, and objectives for an artefact. Models make use of constructs to represent a problem within its solution space. Methods can be algorithms or guidelines that search the solution space and enable instantiations that are computer-based systems implemented in an organisation.

Johannesson and Perjons (2012) identify a relationship between artefacts, people, practices, and problems. Practices are a set of activities that are performed regularly and are seen to be meaningfully related to each other by the people engaging in them. The relationship states that when people engage in practices, they may encounter practical problems that prevent them from completing their practices. To combat any problems encountered, people make use of artefacts that directly address the problems. The DSR methodology can be applied to a range of domains within IT to solve practical problems. Examples showing the diverse application of the DSR methodology can be seen in creating a mobile health application (Myers & Venable, 2014), an information system for law enforcement (Kaza, Hu, & Chen, 2011), and measuring the value and impact of Enterprise Architecture by stakeholders within an organisation (Meyer, Helfert, Donnellan, & Kenneally, 2012).

DSR will be used as the research methodology for this study. DSR is ideal for IT research as it is proactive instead of reactive like typical behavioural science research (de Villiers, 2005). The goal of DSR is to create innovative artefacts that address practical problems (Hevner et. al, 2004). As such, DSR will be used to create an artefact in the form of a prescriptive model to solve the problem of recommending the MAC for a field of law to legal researchers. Following DSR will allow for a theoretical and practical contribution from the study. The model consisting of the techniques and algorithms to recommend the MAC will form the theoretical contribution while the implementation of a proof-of-concept of the prescriptive model will form the practical contribution.

2.3 The Three Cycle View of DSR and Guidelines.

This section will present the three-cycle view of DSR and a set of guidelines for DSR. A discussion on artefacts will also be presented.

2.3.1 Three Cycle View of DSR

The DSR methodology consists of an iterative three-cycle process that results in the output of an artefact (Hevner, 2007). Figure 2-1 illustrates the three cycles mapped to their specific domain. The three cycles are:

- The relevance cycle;
- The rigor cycle; and
- The design cycle.

The **relevance cycle** connects design science with the environment of the application domain. An application domain is made up of people, organisational systems, and technical systems. During the relevance cycle, requirements and acceptance criteria for the artefact are determined. The cycle also encompasses field testing of the artefact once it has been completed to determine whether additional iterations are required.

The **rigor cycle** allows the project to set a firm basis based on previous work and existing artefacts. It is important to identify and analyse previous sources of work to clearly detect opportunities or problems. Analysing previous sources of work will also ensure that artefacts created are contributions to the body of knowledge and not based on the application of well-known processes. All existing theories and techniques identified in the rigor cycle are passed through to the design cycle.

The **design cycle** consists of iteratively building and evaluating artefacts until the artefacts are accepted within its application domain. The design cycle is dependent on both the relevance and rigor cycles because the relevance cycle identifies the requirements whilst the rigor cycle provides theories and techniques related to design and evaluation. The design cycle is the core cycle of the DSR methodology as it is within this cycle that the artefacts are created and evaluated.

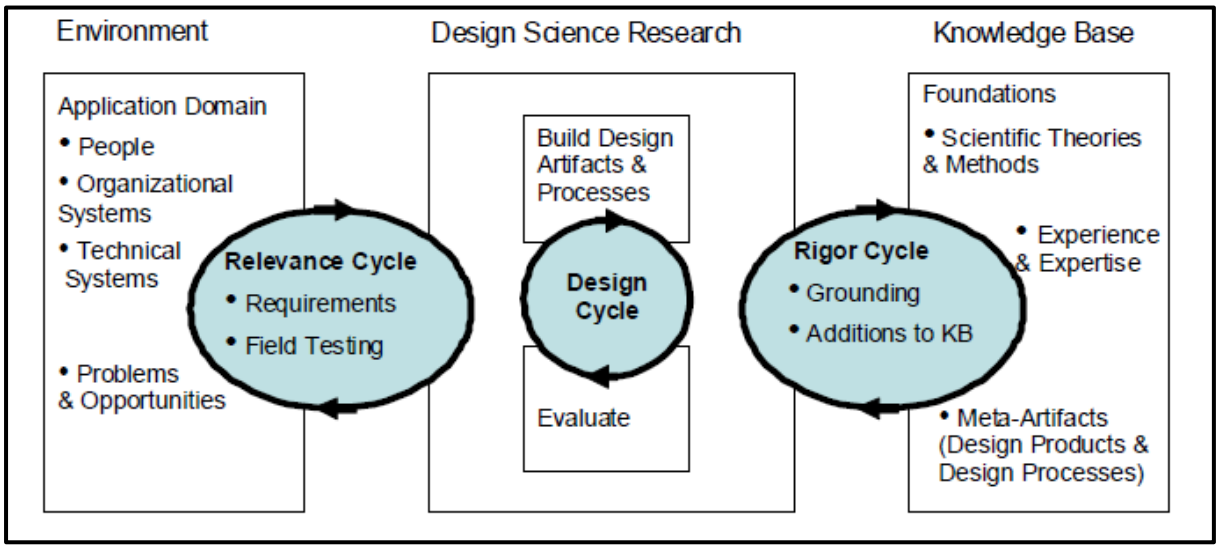


Figure 2-1: Three Cycles of DSR (Hevner,2007)

2.3.2 DSR Guidelines

Various factors can influence the design process of an artefact. The factors that were identified by Hevner et al. (2004) are:

- Volatile requirements and constraints due to poorly defined environmental contexts;
- Complex interactions amongst various entities within the problem space; and
- Constant flexibility to alter designs and processes.

Hevner et al. (2004) provides six guidelines on how DSR can be conducted within an IT domain. The guidelines are:

- Guideline 1: Problem Relevance;
- Guideline 2: Research Rigor;
- Guideline 3: Design as a Search Process;
- Guideline 4: Design as an Artefact;
- Guideline 5: Design Evaluation; and
- Guideline 6: Research Contributions.

The first guideline, **Problem Relevance**, requires an artefact to be created that addresses a business problem. All opportunities presented should be taken to further the development of the artefact. Opportunities can be interactions with experts, other organisations, and information technology systems. Two criteria are used to determine relevance, namely representational fidelity and implementation. In terms of representational fidelity, the artefact must accurately represent the business and technology within the problem domain. Furthermore, experts must be able to implement the artefact within the problem domain.

The second guideline, **Research Rigor**, requires that an artefact undergo an iterative process of development and evaluation. The use of the DSR methodology ensures the iterative development of the artefact. The third guideline, **Design as a Search Process**, states that the research should be iterative to ensure that an effective artefact that solves a problem is found. To ensure that an effective artefact is created, three factors must be considered namely means, ends, and laws. Means refers to the actions and resources available to construct an artefact whilst ends represent the goals and constraints of an artefact. Laws are forces within the problem domain that experts have no control over.

The fourth guideline, **Design as an Artefact**, states that the result of design science must be an artefact that is either a model, method, or instantiation. Furthermore, the resultant artefact should be represented and presented in a manner that allows for evaluation and comparison with similar artefacts. The fifth guideline, **Design Evaluation**, states that the quality and efficacy of a design artefact must be continuously evaluated. Continuous evaluation ensures that the artefact meets all requirements and constraints within its problem domain. Requirements and constraints are decided on by the experts in the domain. Evaluation of the artefact will also determine its level of quality by how well the artefact can be evaluated based on a set of criteria. Criteria used for evaluation can be functionality, completeness, accuracy, performance, and reliability. The sixth guideline, **Research Contributions**, requires the DSR to provide clear contributions. Contributions can be the design artefact, theoretical foundations, or evaluation methodologies. Furthermore, the artefact must be identifiable and validated as a new contribution to research. Theoretical foundations refer to a new artefact that improves existing theoretical foundations whilst evaluation methodologies refer to the creation and use of new methods and criteria to evaluate artefacts.

The guidelines discussed above are summarised in Table 2-1.

Guideline	Description
Guideline 1: Problem Relevance	Research must create artefacts that will address relevant organisational problems
Guideline 2: Research Rigor	Methods must be applied to construction and evaluation of the artefact being designed
Guideline 3: Design as a Search Process	Designing an artefact must be an iterative process. All options should be used until a final, accepted artefact is achieved
Guideline 4: Design as an Artefact	Research must produce a design in the form of an artefact
Guideline 5: Design Evaluation	The quality of an artefact must be well demonstrated through evaluation method
Guideline 6: Research Contributions	The result of DSR must provide a clear contribution to the body of knowledge relating to artefact's design, construction, and evaluation

Table 2-1: DSR Guidelines

2.3.3 Research Artefact Types

An artefact is an object created by humans to address a practical problem (Johannesson & Perjons, 2012). All artefacts consist of a construction, are part of an environment, and perform a function. An artefact's construction refers to how an artefact's components and inner workings relate and interact with each other. All artefacts identified by Johannesson and Perjons (2012) in Section 2.2 operate within a specific environment under certain conditions to achieve a specific goal.

Constructs provide definitional knowledge as they can be terms, notations, definitions, and concepts that are used to formulate problems and solutions. Examples of constructs are classes in object-orientated programming, methods in Java, and functional dependency in relational databases (Johannesson & Perjons, 2012). It is important to have correct constructs as they allow for the construction of models (Gregor & Hevner, 2013).

Models are used to depict or represent objects unlike constructs that provide definitional knowledge. Three types of models exist, namely (Johannesson & Perjons, 2012):

- Descriptive models;
- Prescriptive models; and
- Predictive models.

Descriptive models are used to represent existing situations and help explain the nature of the situations. Additionally, descriptive models can describe possible solutions to practical problems. Prescriptive models are used to provide descriptions of possible future solutions and aid in constructing artefacts. Predictive models are used to forecast behaviour of systems and objects. As such, models can express descriptive, prescriptive, or predictive knowledge.

Methods help express prescriptive knowledge by providing guidelines and processes to solve practical problems and achieve goals. Methods can be in the form of algorithms, or can be informal such as best practices or rules of thumb.

Instantiations are working systems that can be used within a domain. A working system consists of an instantiated artefact such as a model of a blueprint or architecture. In terms of a working system, constructs cannot be instantiations as they would result in a small outcome that cannot be regarded

as a working system. Furthermore, methods cannot be instantiations as they are used to help create a working system. A working system does not instantiate a method.

The artefact produced from this research will be a prescriptive model. The prescriptive model will provide a possible solution in recommending the MAC for a field of law to legal researchers.

2.4 Case Study and Application of DSR

This section will introduce LexisNexis as the case study that will be used for this research. A report on how the DSR methodology can be applied will then be provided.

2.4.1 Context and Case Study

A case study refers to an empirical inquiry that investigates a phenomenon within a real-life context (Yin, 2014). The South African division of LexisNexis will be used as a case study for this research. LexisNexis is a legal organisation that provides legal advisory services and products. Amongst their products is the LegalCitator that provides an analysis of legal cases. Part of LexisNexis' advisory services include performing research to find the MAC for a field of law. This is a tedious process that LexisNexis wants to automate and incorporate into their LegalCitator product. Developing a prescriptive model to recommend the MAC is the aim of this research. Questionnaires will be sent out to experts at LexisNexis to understand the problems and challenges faced within the legal domain. Extant systems will also be investigated to determine shortcomings and requirements for the prescriptive model. Part of recommending the MAC requires the analysis of legal citations that are found within legal cases.

Legal citation is a language of abbreviations used to save space that is usually consumed by unnecessarily long references (Martin, 2013). Legal citation allows legal practitioners to refer to legal authorities with precision and generality, therefore allowing readers to easily follow the references. References that are correctly written in legal citation allow a reader to effortlessly identify a document to which a legal practitioner is referring and provide the reader with enough information to find the referenced document. Legal citations are often labelled with a particular action that depends on what decision was made regarding the legal case that is being cited. Some of the labels that LexisNexis use are (LexisNexis, 2017b):

- Applied;
- Distinguished; and
- Followed.

Distinguished is defined as follows:

"The court in the subsequent case holds that the legal principles articulated by the primary case (usually otherwise persuasive or binding authority) do not apply because of some difference between the two cases in fact or law."

Followed is defined as follows:
"The annotation is similar to applied but is used in circumstances where the facts in the primary case resemble reasonably closely the facts in the subsequent consideration case."

Applied is defined as follows:
"A principle of law articulated in the primary case is applied to a new set of facts by the court in the subsequent case."

Various information management approaches such as summarisation and classification have been used within the legal domain (Galgani & Hoffmann, 2010). Summarisation reduces the length and detail of a document without discarding the document's main points (Gupta & Lehal, 2009). Summarisation has been used to classify sentences in a legal report to determine if sentences should be part of an extractive summary or not (Hachey & Grover, 2005). Classification is the process of applying a model or classifier to a set of data to predict what class labels the data fits into (Han, Jiawei, Kamber, Micheline, Pei, 2012). In the context of legal citations, Galgani and Hoffmann (2010) used an incremental approach based on a Ripple Down Rule methodology to classify legal citations. The authors created their own corpus of legal citations based on legal reports obtained from the Australasian Legal Information Institute. Classification was made possible through a series of rules that processed a legal citation. The rules identified aspects of a case such as:

- General data -Judge's names;
- General data –Parts such as 'plaintiff' and 'defendant'; and
- General data – Division of the court.

Once the rules had processed the legal citation, a class label was added to the case. The labels added were those used by LexisNexis (2017b).

Analysing legal cases implies that the text has to be processed. Various problems can be experienced when processing text. It is important to understand the format in which the file containing the text is stored as it can affect the text processing. Formats such as Portable Document Format (PDF) are known to bring about inconsistencies with formatting which results in inefficient processing of text. Once there is an understanding of the format in which a legal case is saved, the text must then be pre-processed to remove unnecessary words. Pre-processing can be achieved by performing IE (Section 3.4) and NLP tasks (Section 3.5.1). In the context of legal cases, not only must general case data be extracted but the cases referred to must also be extracted. To extract a case referred to, which is written in legal citation form, the referred to case's information, must be broken up into smaller pieces.

2.4.2 Application of DSR to this Study

The DSR methodology will be used in this study to create the final and accepted artefacts. This study will therefore use the three-cycle view of the DSR as presented by Hevner (2007) along with the DSR activities presented by Peffers et al. (2007). The mapping of the cycles and activities is depicted in Figure 2-2. In the figure, the DSR activities are numbered and start with the prefix of 'A'.

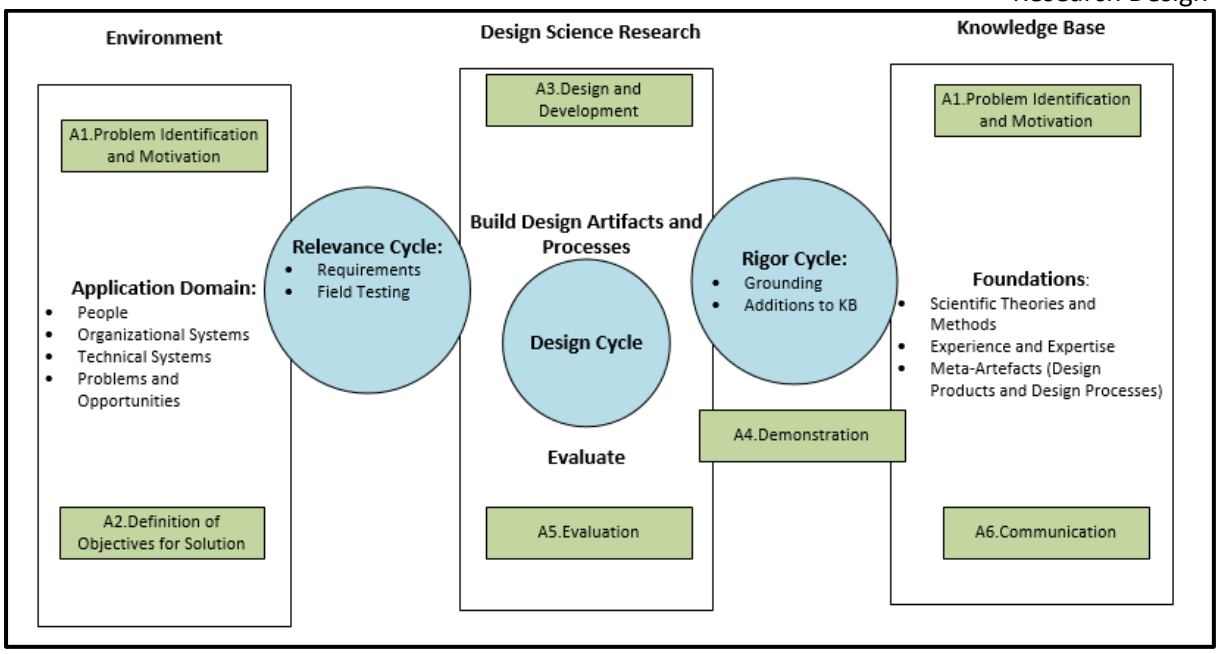


Figure 2-2: Mapping of DSR Activities and Cycles (Author's own work)

During the **relevance cycle**, requirements for the research will be identified. The first two DSR activities will be completed. Experts within LexisNexis will be consulted to determine what problem they require a solution to. An analysis of LexisNexis' existing systems will be conducted to identify any shortcomings. This will in turn help develop research questions and research objectives. An appropriate sample of experts from LexisNexis will be selected based on their expertise within the domain of the research problem. A series of questionnaires will be drafted for these experts to complete.

During the **rigor cycle**, various theories and techniques will be analysed to form a foundation to base the artefact on. The first DSR activity, Problem identification and Motivation, will be further expanded to clarify the objectives derived from the second DSR activity, Definition of Objectives, in the relevance cycle. Various research methods will be reviewed within literature to select the most appropriate method for this research. Throughout the rigor cycle all findings will be communicated to the relevant experts.

During the **design cycle**, DSR activities Design and Development, Demonstration, and Evaluation will be completed. The proposed solution will be created in the form of two artefacts namely, the model and the prototype of the model. Questionnaires will be sent to the experts from LexisNexis to obtain an understanding of the formal and informal processes followed when working with legal cases. The artefact will be designed, developed, demonstrated and evaluated continuously until a final artefact is accepted. Figure 2-2 maps the chapters of this research to the DSR activities, DSR guidelines, ROs, research methods, and deliverables.

Figure 2-3 expands on Table 1-1 by including the research methods that will be used throughout this research. A literature review and extant systems analysis will be conducted to get an understanding of the techniques and algorithms that can be applied to problems faced in the legal domain. LexisNexis will be used as a case study and experts from LexisNexis will be given questionnaires to answer. Prototyping, experiments, and evaluation methods will be used to evaluate the prototype created.

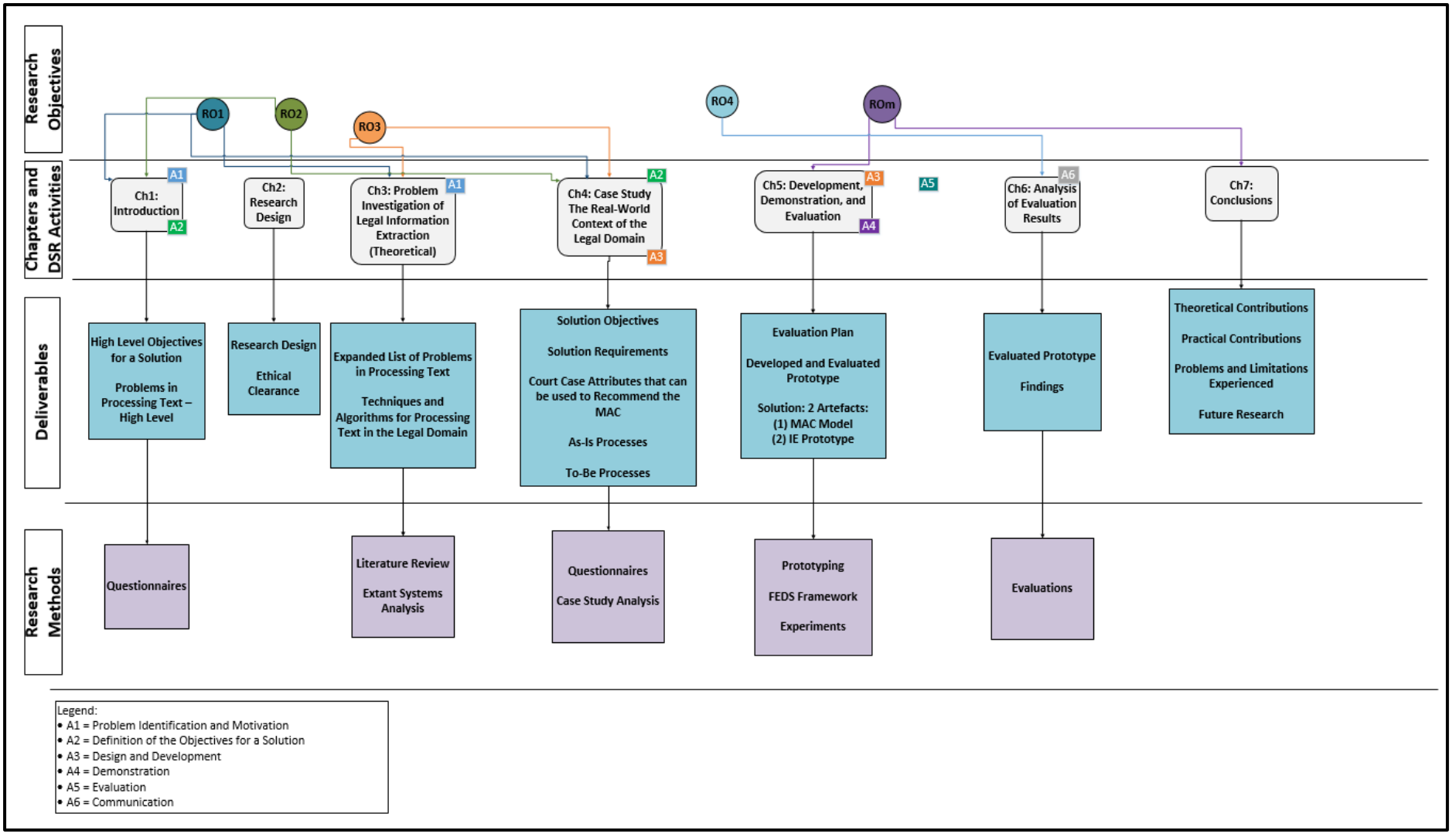


Figure 2-3: Chapter Layout

2.5 Ethical Considerations

Ethics refers to moral principles that govern or influence conduct (Soanes & Stevenson, 2004). DSR has become an important research methodology in the field of Information Systems (IS) as it seeks to improve various areas. DSR often requires the use of participants throughout a research study which has resulted in a set of principles for researchers to adhere to. Areas that DSR seeks to improve are (Myers & Venable, 2014):

- Effectiveness and efficiency within an organisation;
- People's health;
- Education; and
- Quality of life.

Mason (1986) states that IS researchers must take the responsibility to ensure that any information system developed is used for the correct and ethical reasons. Furthermore, Mason (1986) proposes four ethical questions that researchers must consider. The four questions relate to privacy, accuracy, property, and access. For this research accuracy, property, and access will be considered. Accuracy refers to gathering error-free information which will be ensured by obtaining published versions of legal cases. Property refers to who owns the intellectual property of the artefact and access refers to who will be authorised to access the information on the artefact. LexisNexis are the owners of the data provided for this research while the Nelson Mandela University has the right to access and distribute the findings from this research.

Many universities and research institutions require researchers to obtain permission from an ethics board to conduct research that involves people or animals. In conducting this research, the DSR methodology will be used. As such, the DSR methodology requires the researcher to interact with various experts and obtain information from the experts throughout the study. Therefore, REC-H approval was obtained from the Nelson Mandela University. The ethical clearance number for this research is H17-SCI-CSS-009 (Appendix C).

2.6 Summary

This chapter investigated the research and design methodology that will be used in this research, namely the DSR methodology. The DSR methodology consists of three iterative cycles that must be completed (Hevner, 2007). These cycles are the relevance, rigour, and design cycles. Additionally, there are also six activities that must be completed when following the DSR methodology (Peffers et al., 2007). These activities guide the researcher in starting and completing the research. This research will make use of questionnaires, a literature review, and a case study and to identify problems faced within literature and the legal domain, as well as completing the ROs identified in Section 1.6. Ethical clearance was also obtained from the Nelson Mandela University.

The next chapter will apply the first activity of the DSR methodology, namely Problem Identification and Motivation. The chapter will focus on addressing RO1 and RO3.

Chapter 3: Problem Investigation of Legal Information Extraction (Theoretical)

3.1 Introduction

The previous chapter investigated the research and design methodology that will be applied throughout this research. This chapter reports on the first activity of the DSR methodology, namely Problem Identification (Figure 3-1) and will address the following research objectives (Section 1.6):

- **RO1**: *Identify the problems experienced when processing text as identified by literature and within a real-world context.*
- **RO3:** *Determine what techniques and algorithms can be used to recommend the MAC.*

Different techniques for IE can be applied to process information, specifically text. IR must take place before any processing can occur (Section 3.2) and several IR models have been proposed (Section 3.3). IR is possible through general IE techniques (Section 3.4). However, additional techniques are available. NLP is one such technique that can be used to process raw text (Section 3.5). Sources of online information can also be processed (Section 3.6). Another technique to process information is regular expressions (Section 3.7). Storing processed information is possible by means of NoSQL databases (Section 3.8). After information has been stored, it can then be ranked (Section 3.9) Several frameworks and methodologies in the legal domain have been proposed (Section 3.10). Lastly, based on the techniques investigated, a generic IE model is presented (Section 3.11).

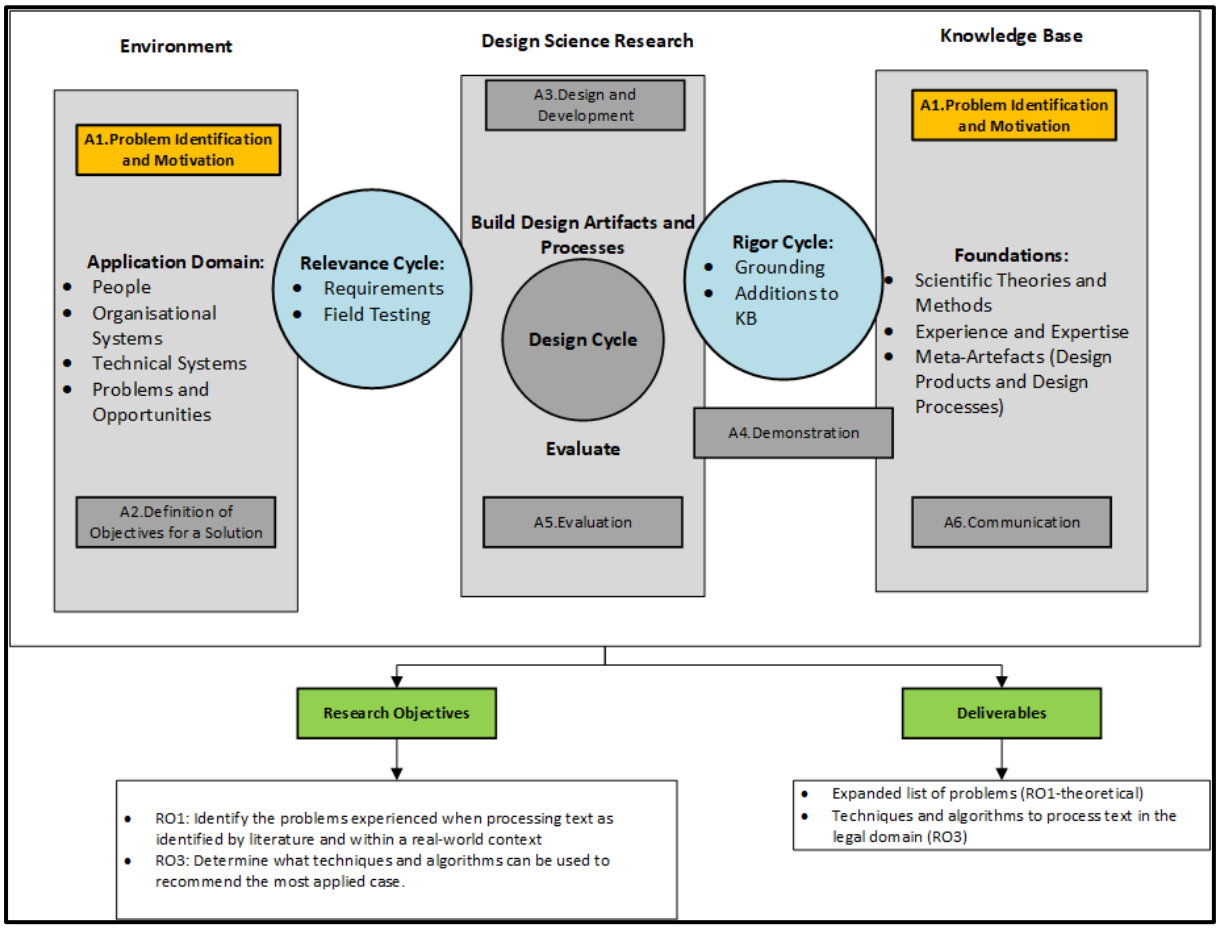


Figure 3-1: Chapter 3 DSR Activities

3.2 Information Retrieval Processes

IR is a necessary technique for processing text as information first needs to be retrieved before any extraction of text can occur. To return valuable results to a user, various processes and intermediate stages must be conducted on a set of data. IR is a process that deals with the representation, storage, and searching of a collection of data in response to a request from a user (Roshdi & Roohparvar, 2015). IR's main goal is to return relevant information based on a user's request. Additionally, IE can be applied to further process information from IR to extract facts from bodies of text. Figure 3-2 illustrates the high-level process that should be followed when processing information.

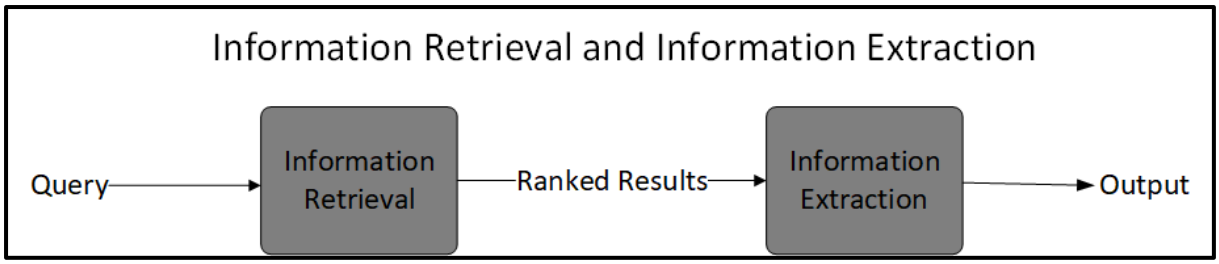


Figure 3-2: Link between IR and IE (Author's own work)

Relevance of the information returned by IR can be determined by applying two measurements known as precision and recall. Precision deals with the percentage of retrieved documents that are relevant to the user's query whilst recall refers to the percentage of documents that are relevant to a query and are retrieved. To help ensure relevance, all IR systems must support three basic processes, namely (Roshdi & Roohparvar, 2015):

- Representation of a document's content;
- Representation of the user's information need; and
- A comparison of the two above mentioned representations (query and document).

The IR processes are completed by following five intermediate stages, namely (Roshdi & Roohparvar, 2015):

1. Indexing;
2. Searching;
3. Matching;
4. Query-dependent ranking;
5. Filtering.

Indexing is done during the process of **representing a document's content**. Indexing is the process of creating a logical view of documents in a collection by means of keywords or terms (Ceri et al., 2013). When representing a document's content, indexing occurs offline. The result is an indexed representation of the document. Commonly used indexing techniques for IR are the signature file method and inversion indices (Roshdi & Roohparvar, 2015). The signature file method makes use of hashing and superimposed coding. The result of the signature file method is a document containing sequentially stored signatures that allow for faster searching. Inversion indices work with the keywords of a document. These keywords are inverted to allow for faster retrievals.

Searching begins during the process of **representing a user's information need,** when the user creates a query. In the second stage, the query is parsed through a search algorithm to search for documents. Pre-processing tasks such as tokenisation and stop-word removal must be applied onto a document's text before searching can occur (Gurusamy & Kannan, 2014). These tasks aid in reducing the size of a document's body of text by eliminating unnecessary and confusing words. A smaller body of text

allows for meaningful keywords to be identified and aid in returning relevant documents to a user. The absence of these two tasks can result in poor performing IR models. Commonly used searching algorithms for IR are the linear search, brute force search, and binary search. Linear search is a simple search algorithm that finds a keyword in a list by traversing every keyword contained in the list. Brute force searching enumerates all potential keywords for a solution and determines if each keyword satisfies the problem. Binary searching finds a keyword based on its position in a list. A keyword is matched with the middle element of the list and if a match is found, the match is returned. If no match is found, then processing continues to the left or right of the list depending on the value of the middle element.

The third process is the **comparison of the two representations,** which is done by **matching** the two representations to obtain retrieved documents. Once the comparison is completed, the outcome is a set of **ranked** retrieved documents in response to the user's query. This form of ranking is known as query-dependent ranking. The user then provides feedback if different information is needed by altering the query to **filter** the results. Figure 3-3 represents the processes and intermediate stages described by Roshdi and Roohparvar (2015) along with accompanying techniques, algorithms, IR models (Section 3.3), and quality measurements. Once an IR system supports the three processes, the system can then be tailored to a specific area.

IR systems are applied in various areas such as digital libraries, search engines, and media search engines (Roshdi & Roohparvar, 2015). Digital libraries consist of vast amounts of digital documents that are only accessible via computer. Contents of a digital library can be stored locally or remotely. Search engines are a common form of an IR system as various IR techniques are applied on large scale text documents. Another application for IR is with media searches where techniques are used to retrieve various forms of media such as images. In addition to the areas mentioned by Roshdi and Roohparvar (2015), IR can also be applied to the maintenance and evolution of a software project (Binkley & Lawrie, 2009). Each IR system used in these areas are based on a specific IR model.

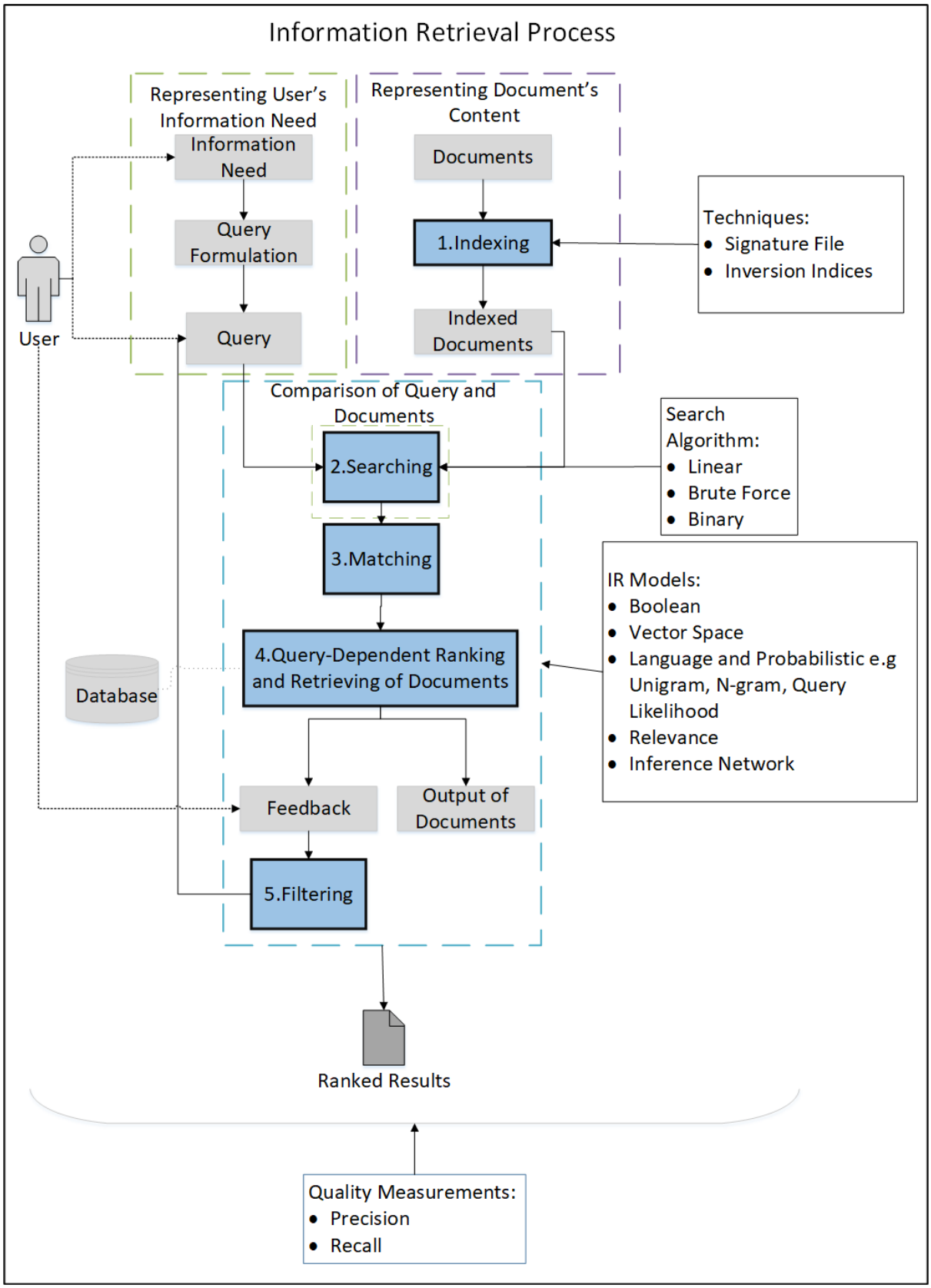


Figure 3-3: IR processes and techniques (Author's own work)

3.3 Types of Information Retrieval Models

IR models address the second to fifth intermediate stages of IR. An IR model controls how a document and query are represented as well as how to define the relevance of a document to a user's query (Liu, 2011). All documents and queries are treated as a set of distinct terms carrying a specific weight (Equation 3.3-1). A term is a word whose semantics helps keep track of a document's main themes.

Equation 3.3-1 represents a collection of documents and is interpreted as follows:

$$V = \{t1, t2, \dots, t_{|v|}\}$$

Equation 3.3-1: Representing a collection of documents

Given a collection of documents, D, let V be the set of distinctive terms in the collection, where t_i is a term. V is known as the vocabulary of the collection with $|V|$ representing the collection's size (number of terms in V). A weight $W_{ij} > 0$ is associated with each term t_i of a document $d_j \in D$.

Five IR models were proposed by Liu (2011) namely:

- A Boolean model;
- A Vector Space model;
- A Language and Probabilistic model;
- A Relevance model; and
- An Inference Network model.

3.3.1 The Boolean Model

The Boolean model is one of the first and simplest IR models developed and makes use of exact matching using Boolean algebra when matching documents to a user's query (Liu, 2011). In terms of document representation, the Boolean model represents documents and queries as sets of terms, where a term is considered as being either present or absent in a document. A user's query terms are combined using the following Boolean operators (Molloy Librarian, 2017):

- AND – all terms stated in the query must be found within the results;
- OR – one of the terms stated in the query must be found within the results; and
- NOT – the term following NOT in the query will be excluded from the results.

With regards to document retrieval, the Boolean model returns every document that results in the user's query being true. Therefore, document retrieval is binary in the sense that a document is either relevant or irrelevant. The binary nature of the Boolean model is disadvantageous as it leads to poor results returned to the user. Furthermore, the Boolean model is unable to rank and return a list of documents (Roshdi & Roohparvar, 2015). The reasons the Boolean model cannot rank documents is due to its binary nature and it assumes that all documents in a retrieved set are equivalent in terms of relevance. Therefore, the effectiveness of the Boolean model depends entirely on the user. A user who is well experienced can create complex queries to retrieve data. The only advantages of the Boolean model is that its results are predictable, easy to explain to users, and the operands of a query can be any feature from a document (Croft, Metzler, & Strohman, 2015).

3.3.2 The Vector Space Model

The Vector Space model is the most commonly used IR model (Al-Anzi & AbuZeina, 2018). Documents are represented as weighted vectors, where each component's weight is calculated based on a variation of term frequency (TF) or term frequency-IDF scheme (Liu, 2011). The weights of terms within this model can be any number unlike the Boolean model where weights are in {0, 1}. In the TF scheme, a term's weight is based on the amount of times that the term occurs in a document. A

disadvantage of the TF scheme is that it does not cater for a term appearing in multiple documents of a collection. The TF-IDF scheme has many variations, however, the most basic variation is the following (Equation 3.3-2):

Let N be the total number of documents in the system or the collection and dfi be the number of documents in which term ti appears at least once.

Let fij be the raw frequency count of term ti in document dj. Then, the normalised term frequency (denoted by tfij) of ti in dj is given by:

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots f_{|v|j}\}}$$

Equation 3.3-2 The normalised term frequency

Where the maximum is computed over all terms that appear in document d_j.

Queries are represented the same way as a document in a collection. The term weight of each term in the query can be calculated the same way as a normal document or a different method can be used. Unlike the Boolean model, the vector space model is not binary. Documents are ranked based on their degree of relevance to a user's query. Relevance can be determined by calculating the similarity of a query to each document within the collection. Many measures to calculate similarity have been proposed but a popular measure is the cosine similarity. The cosine similarity computes the cosine of an angle between a query vector and a document vector. Once similarity has been calculated, ranking is performed using the similarity values. The top ranked documents are more relevant to the user's query. An alternative method to calculate relevance is to use the Okapi method which calculates a relevant score for each document associated with a query.

3.3.3 Language and Probabilistic Models

Language models are based on probability and are founded from statistical theories (Liu, 2011). Language models represent text in various language technologies such as speech recognition, machine translation, and handwriting recognition. Examples of language and probabilistic models are Unigram and N-gram Models, a Query Likelihood Model, and a Relevance Model.

3.3.3.1 Unigram and N-gram Models

An example of a simple language model is a unigram that has a probability distribution over all the words in a language. Therefore, a probability of occurrence is created for every word in a language (Croft et al., 2015). The following example is provided by Croft et al. (2015) for a unigram language model:

If the documents in a collection contained only five words, then a possible language model for the collection can be (0.2;0.1;0.35;0.25;0.1). Where each number represents the probability of a word occurring. It must be noted that previous words do not influence the prediction of the next word.

Croft et al. (2015) further state that if a document is treated as a sequence of words then the probabilities in the language model predict what word will occur next in a sequence. With applications like speech recognition, n-gram language models are used to predict words. N-gram models differ from unigram models as n-gram models predict words based on the previous n-1 words. Common n-gram models are bigrams and trigrams. Bigrams base prediction on two words, being the previous word and current word whilst trigrams base prediction on the previous two words with the current word. In terms of search applications, language models are used to represent topical content of each document. In the context of search applications, a topic refers to a probability distribution over a

collection of words. Furthermore, the topic of a query, by an information seeker, can be represented as a language model. This results in three possibilities for retrieval based on language models, namely:

- A possibility based on the probability of generating query text from a document language model;
- A possibility based on generating the document text from a query language model; and
- A possibility based on comparing the language models that represent queries and document topics.

3.3.3.2 Query Likelihood Model

Query Likelihood Models generate query text from a document language model. The query likelihood retrieval model ranks documents based on the probability that query text can be generated by the document language model (Croft et al., 2015). As such, this is a topical relevance model because the probability of a query being generated is the measure of how likely a document is about the same topic as the query. To rank the documents based on a query, one must calculate P(D|Q), that is, the probability of document D given query Q. Equation 3.3-3 depicts how Bayes' Rule can be used to calculate P(D|Q).

$$p(D|Q) = P(Q|D)P(D)$$

Equation 3.3-3: Bayes' Rule

P(D) refers to the prior probability of the document and is assumed to be uniform. Therefore, P(D) does not affect the ranking. P(Q|D) refers to the likelihood of the query given a document. A unigram language model can be used to calculate P(Q|D), using Equation 3.3-4, where q_i represents a query word and n refers to the amount of words in the query word.

$$P(Q|D) = \prod_{i=1}^{n} P(qi|D)$$

Equation 3.3-4: Calculating P(Q|D) for a Unigram Model

To calculate P(q_i |D) an estimate for the language model probabilities is needed. To do this Equation 3.3-5 is used:

$$P(qi|D) = \frac{fqi, D}{|D|}$$

Equation 3.3-5: Calculating Maximum Likelihood Estimate

Where $f_{qi,}$ D represents the amount of times that word qi occurs in document D, and |D| represents the amount of words in document D. Equation 3.3-5 is known as a maximum likelihood estimate, meaning that it makes the observed value of $f_{qi,}$ D most likely. The disadvantage of this estimate is that if any query words are missing from the document then the score returned for P(Q|D) will be zero. To avoid this issue, smoothing can be applied. Smoothing also overcomes data sparsity. Smoothing lowers the probability estimates for words that are seen in a document and assigns the 'leftover' probability to the estimates for words that are not seen in a document. Estimates for unseen words are based on the frequency of occurrence of words in the entire document collection. Therefore, if P(q_i |C) represents the probability for query word I in the collection language model for document collection C, then the estimate used for unseen words in the document is $\alpha DP(qi|C)$ where αD is a constant coefficient controlling the probability assigned to unseen words. To ensure that the probabilities add up to one, the probability estimate for a seen word in a document is shown in Equation 3.3-6.

$$P(qi|D) = (1 - \alpha D)P(qi|D) + \alpha DP(qi|C)$$

Equation 3.3-6: Probability Estimate for a Seen Word

Various estimates occur as a result of different values for αD. However, for simplicity it is best to set αD to a constant value, λ. The collection language model probability estimate used for word qi is $\frac{Cqi}{|C|}$ where Cqi represents the amount of times a query word is found in a collection of documents and |C| is the amount of words in the collection. This changes Equation 3.3-6 to Equation 3.3-7 :

$$P(qi|D) = (1 - \lambda)\frac{fqi, D}{|D|} + \lambda\frac{cqi}{|C|}$$

Equation 3.3-7: Probability Estimate for a Seen Word with a Constant Value

This form of smoothing is called the Jelinek-Mercer method. Substituting the estimate results in Equation 3.3-8:

$$P(Q|D) = \prod_{i=1}^{n}((1 - \lambda)\frac{fqi, D}{|D|} + \lambda\frac{cqi}{|C|})$$

Equation 3.3-8

However, the multiplication of many small numbers can lead to accuracy problems. Therefore, logarithms are used to avoid accuracy problems. The resultant equation will then be Equation 3.3-9:

$$logP(Q|D) = \sum_{i=1}^{n} log\,((1 - \lambda)\frac{fqi, D}{|D|} + \lambda\frac{cqi}{|C|}$$

Equation 3.3-9: Application of Logarithms

3.3.4 Relevance Model

The Query Likelihood Model is limited when it comes to modelling information needs and queries. Furthermore, it is difficult to incorporate information into the ranking algorithm with respect to relevant documents or that multiple queries can be used to describe an information need. These issues can be overcome by extending the model into what is known as a Relevance Model.

A Relevance Model represents the topics covered by relevant documents. Queries are viewed as small samples of text that are generated from the relevance model. Relevant documents are larger samples of text generated by the same model. Examples of relevant documents for a query must be given to estimate probabilities in a relevance model and use this model to predict the relevance of new documents. Predicting the relevance of documents is known as a Document Likelihood Model where P(D|R) is calculated. A Document Likelihood Model is used in conjunction with a Relevance Model. Whilst the document likelihood model incorporates term frequency, it is still difficult to calculate P(D|R) and compare it across different documents. The reason for this is that documents contain a variable number of words compared to a query. Considering two documents Da and Db, containing five and 500 words each. The large difference in word count results in the comparison of P(Da|R) and P(Db|R) for ranking to be difficult than comparing P(Q|Da) and P(Q|Db). An additional issue is obtaining examples of relevant documents. However, an alternative to this is to estimate a relevance model from a query and compare this language model directly with the model from a document. Documents would then be ranked by the similarity of the document model to the relevance model. Therefore, a document with a model similar to the relevance model is likely to be on the same topic.

To compare any two language models, a measure called Kullback-Leibler divergence can be applied. Kullback-Leibler divergence is defined as follows (Equation 3.3-10):

Given the true probability distribution, P, and another distribution Q that is an approximation to P, the Kullback-Leibler divergence is represented as:

$$KL(P||Q) = \sum_{x} P(x) log \frac{P(x)}{Q(x)}$$

Equation 3.3-10: Kullback-Leibler Divergence

Since the Kullback-Leibler divergence is always positive and larger for distributions that are further apart, the negative Kullback-Leibler divergence should be used as the basis for the ranking function. Furthermore, the correct distribution must be chosen as the true distribution. Once all of this has been taken into consideration the Kullback-Leibler divergence can be expressed in Equation 3.3-11:

$$\sum_{w \in V} P(w|R) \log P(w|D) - \sum_{w \in V} P(w|R) \log P(w|R)$$

Equation 3.3-11

A simple maximum likelihood estimate for P(w|D) is given based on the frequency in the query text, fw,Q, and the amount of words in the query, |Q|. Therefore, the score for a document will be Equation 3.3-12:

$$\sum_{w \in V} \frac{fw, Q}{|Q|} \log P(w|D)$$

Equation 3.3-12: Calculating a Document's Score

Summation occurs for all words in the vocabulary. Words that are not in the query do not contribute to the score and have a zero maximum likelihood estimate.

3.3.5 Inference Network Model

An Inference Network model is made up of a directed, acyclic graph containing nodes. The nodes represent events with possible outcomes whilst the arcs of the network represent probabilistic dependencies between the events (Croft et al., 2015). In the context of IR, nodes represent the observation of a document or document features. The events in an inference network model are binary, indicating that true and false are the only outcomes. An inference network model typically consists of nodes that represent the following:

- A document, D;
- Document features, r_n;
- Probabilities associated with features, θ;
- Parameters, μ;
- Queries, q; and
- Information need, I.

Figure 3-4 illustrates how an inference network model works. “D” represents a document in the form of a webpage that is observed by a user. Every document in a collection has one document node for representation. In the figure, features from a webpage’s title, body, and headings are combined in relation to different parameters. These features each have probabilities assigned to them based on the language models used. The query nodes then combine the features extracted from the

representation nodes to create more complex document features. The network as a whole ultimately computes P(I|D,μ), that is the probability of an information need met given a document and specific parameters. The I node is a combination of all information extracted from query nodes in the form of a probability or belief score. The score is used to rank documents.

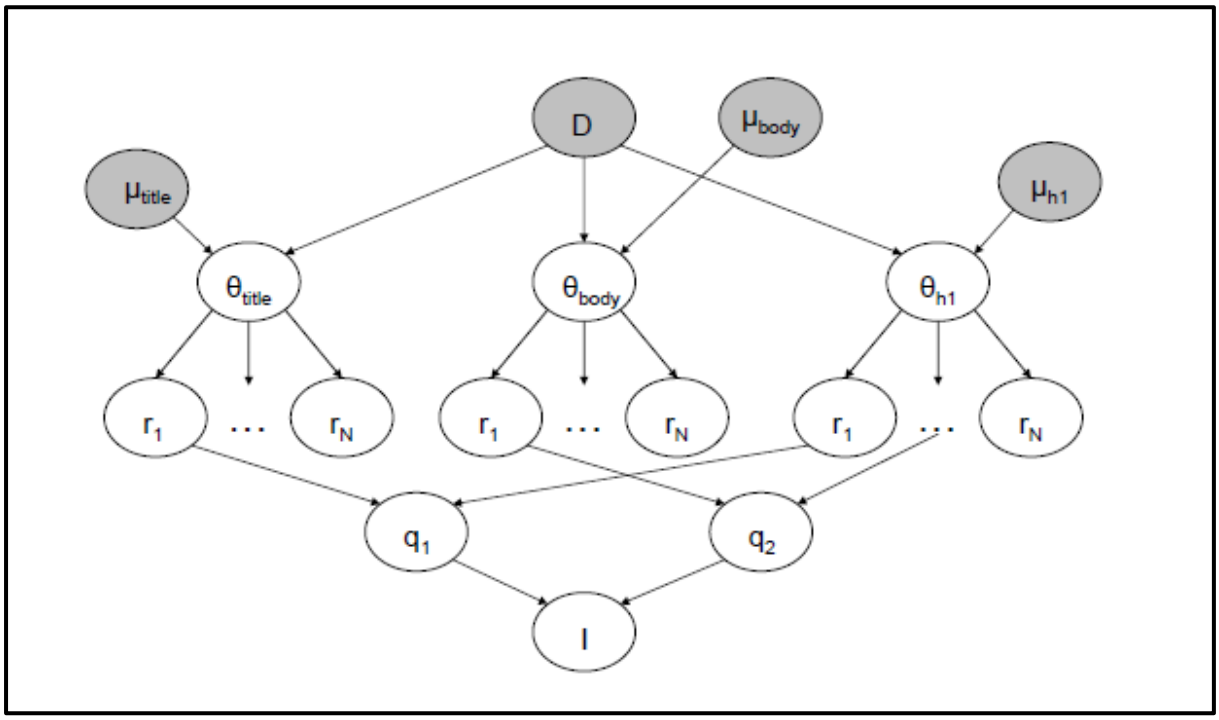


Figure 3-4: Inference Network Model (Croft et al.,2015)

3.3.6 Comparison of Information Retrieval Models

Table 3-1 and Table 3-2 provides a comparison of the IR models that were investigated. The characteristics for IR models were derived from the literature. Based on the criteria from Table 3-1 and Table 3-2, the Vector Space model will be the most suitable model to use for processing documents, since it is the most commonly used IR model, it is also not binary and ranks documents based on its relevance to a user's query. The Boolean and Inference models will not return accurate results due to their binary nature. Language and Probabilistic models would require examples of relevant documents to be shown to the model and could require additional calculations to avoid any shortcomings.

Criteria	**IR Models**			
	Boolean	**Vector Space**	**Language and Probabilistic**	**Inference Network**
How documents are ranked	Unable to rank documents due to binary nature	Ranked based on degree of relevance to user's query	Ranked based on probability	Ranked based on a probability or belief score
How queries are represented	As set of terms	As weighted terms	As a language model	As a language model

Table 3-1: Comparison of IR Models Part 1

Criteria	IR Models			
	Boolean	**Vector Space**	**Language and Probabilistic**	**Inference Network**
Document Retrieval	Binary	Not binary	Uses a Relevance Model to predict the relevance of documents	Binary
Advantages	First and simplest IR model Uses Boolean Algebra for exact matching	Most commonly used IR model Documents are represented as weighted terms	Based on probability and founded from statistical theories	

Table 3-2: Comparison of IR Models Part 2

3.4 Information Extraction Processes and Techniques[1]

IE is a process that derives structured information from unstructured or semi-structured text (Jiang, 2012). IE differs from IR since IR returns a ranked subset of data that is relevant to a user's query whilst IE extracts facts about entities and relationships (Piskorski & Yangarber, 2013). IR can be used with IE to aid in tasks related to pre-filtering large sets of data. Performing IE requires three processes to be followed that can use different IE techniques to form an IE system. Abdelmagid, Ahmed, and Himmat (2015) state that in addition to processing unstructured and semi-structured text, structured text can also be processed with IE. Unstructured text contains a variety of text related to news or stories and thus makes extraction difficult. Semi-structured text is presented and formatted in a specific manner for a domain whilst structured text is highly formatted, structured, and organised. Applications for IE are seen in many fields such as biomedical research, finances, intelligence agencies, and search engines (Jiang, 2012). Various processes must be followed to extract information from text.

IE consists of three processes as depicted in Figure 3-5, namely (Abdelmagid et al., 2015):

1. Extracting facts;
2. Integrating facts; and
3. Translating the facts to output.

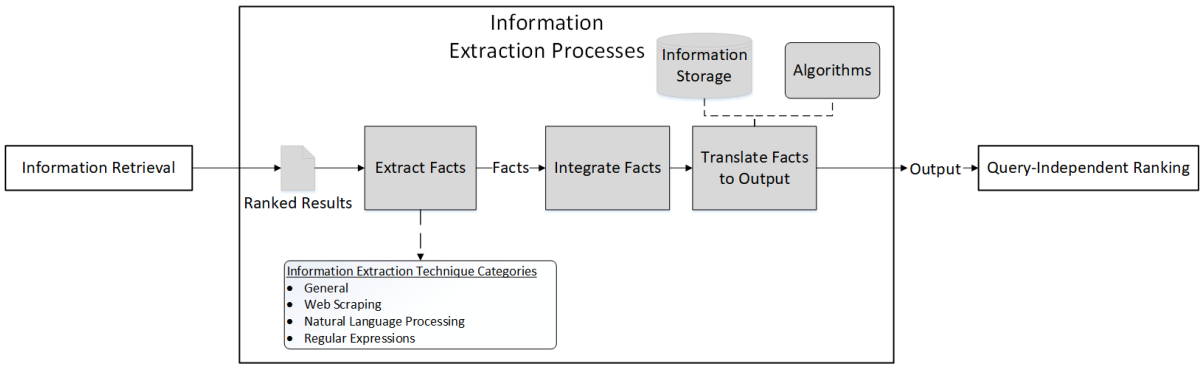


Figure 3-5: IE Process (Author's own work)

[1] The literature discussed in this section was obtained from research that was published as full double-blind peer-reviewed conference paper at the International Conference on Computing, Electronics & Communications Engineering 2018 in August 2018. Padayachy, T, Scholtz, B and Wesson, J. An Information Extraction Model Using a Graph Database To Recommend the Most Applied Case. ICCECE'18 Essex, United Kingdom. (Appendix K)

Extracting facts from a document requires the text to be analysed and extracted. After the facts have been extracted, the facts are integrated to create a larger set of facts or infer new facts. A common issue encountered when determining the meaning of extracted facts is ambiguity (Sumathy & Chidambaram, 2013). This is because often in the English language words or phrases can have multiple meanings. In addition to ambiguity, inconsistencies in text can result from special formats, abbreviations, and acronyms (Gurusamy & Kannan, 2014). Facts can then be put through algorithms to produce output. Various IE techniques can be applied to complete these processes.

Four categories of IE techniques are investigated. A summary of the categories is provided in Table 3-3. This section will report on the general technique's category.

Information Extraction Techniques			
Category	**Technique**	**Author**	**Section**
General Techniques	Named Entity Recognition	Piskorski and Yangarber (2013)	Section 3.4
		Abdelmagid et al. (2015)	
	Co-reference Resolution	Piskorski and Yangarber (2013)	
		Iida, Inui, & Matsumoto (2006)	
	Relation Extraction	Piskorski and Yangarber (2013)	
	Event Extraction	Piskorski and Yangarber (2013)	
Natural Language Processing	Morphological and Lexical Analysis	Piskorski and Yangarber (2013)	Section 3.5
		Chopra, Prashar, and Chandresh (2013)	
	Syntactic Analysis	Chopra et al. (2013)	
	Semantic Analysis	Chopra et al. (2013)	
	Discourse Integration	Chopra et al. (2013)	
	Pragmatic Analysis	Chopra et al. (2013)	
Web Scraping		Vargiu and Urru (2012)	Section 3.6
		Glez-Peña, Lourenço, López-Fernández, Reboiro-Jato, and Fdez-Riverola (2013)	
Regular Expression	Deterministic Finite Automaton	Goyvaerts and Levithan (2009)	Section 3.7
	Non-Deterministic Finite Automaton	Rabin and Scott (1959) Prasse, Sawade, Landwehr, and Scheffer (2015) Hopcroft, Motwani, and Ullman (2006)	

Table 3-3: IE Techniques

Four general IE techniques can be applied to extract facts from text, namely (Piskorski & Yangarber, 2013):

- Named Entity Recognition (NER);
- Co-reference resolution (CO);
- Relation extraction (RE); and
- Event extraction (EE).

NER is a basic technique of IE and processes extracted information from unstructured and structured text (Abdelmagid et al., 2015). When applied, all expressions related to an entity are identified.

Furthermore, NER can involve extracting descriptive information from text about an entity and completing a template based on the extracted information. NER is divided into two tasks, namely the identification and classification of predefined entities. Piskorski and Yangarber (2013) state that predefined entities can be organisations, persons, temporal expressions, and numerical expressions.

The CO technique requires identification of multiple mentions of the same entity. At the time of research not much information could be found on CO. An entity's mention can be (Piskorski & Yangarber, 2013):

- Named;
- Pronominal;
- Nominal; and
- Implicit.

A named mention refers to an entity by name such as *"General Electric"* whilst a pronominal mention refers to an entity by use of a pronoun such as *"John bought food. But he forgot to buy drinks"*. The pronoun is the word *"he"*. A nominal mention refers to an entity by a nominal phrase such as *"Microsoft revealed its earnings. The company also unveiled future plans"*. In the aforementioned example, *"The company"* is the definite noun phrase that refers to *"Microsoft"*. Implicit mention uses zero-anaphora to refer to an entity. Zero-anaphora is a gap in a sentence that has an anaphoric function and is often used to refer to an expression that provides necessary information to understand the sentence (Iida et al., 2006). An example of an implicit mention that uses zero-anaphora is seen in *"There are two roads to eternity, a straight and narrow, and a broad and crooked."* In this example, the gaps of the sentence are "a straight and narrow" and "a broad and crooked".

RE involves detecting and classifying predefined relationships between entities identified in a body of text. Piskorski and Yangarber (2013) provide the following examples of RE:

- *EmployeeOf (Steve Jobs, Apple); and*
- *LocatedIn (Smith, New York).*

The first example, EmployeeOf, involves detecting the relationship between the entities of a person and organisation. The person entity is *"Steve Jobs"* while the organisation entity is *"Apple"*. This example extracts the entities from the text "*Steve Jobs works for Apple"*.

The second example, LocatedIn, involves detecting the relationship between the entities of a person and location. The person entity is *"Smith"* while the location entity is *"New York".* This example extracts the entities from the text "*Mr. Smith gave a talk at the conference in New York"*.

EE involves identifying events in text and deriving a detailed and structured set of information about the events (Piskorski & Yangarber, 2013). During EE, multiple entities and relationships are extracted. As such, EE is said to be the hardest of the four IE tasks as information answering, "who did what to whom, when, where, through what methods?" must be extracted.

NER can be applied with meta-data analysis and tokenisation to this research. All expressions related to a legal case can be identified, tokenised, and extracted. Depending on the approach used to obtain the legal cases, an IE technique called web scraping could be used to obtain the required facts from legal cases.

3.5 Natural Language Processing

This section will investigate the IE technique of NLP. Included in this investigation are the phases and techniques that should be followed when performing NLP. Additionally, the tools available to perform NLP will be compared.

3.5.1 Natural Language Processing Phases and Techniques

NLP explores how computers can be used to process and understand natural language text to perform useful tasks (Chowdhury, 2003). NLP can be used to analyse text that has been extracted from sources such as documents or websites and produce meaning for the text (Singh, 2018). In the context of recommending the MAC, NLP can be applied to text that has been extracted from legal cases. NLP is divided into two categories, namely language processing and language generation. Language processing refers to the analysis of language to produce meaningful representations whilst language generation refers to producing language from a representation (Liddy, 2001). NLP can be applied to various activities such as speech understanding, IE, and knowledge acquisition (Chowdhary, 2012). In the context of IE, NLP can be applied during the Extract Facts process.

There are five phases of NLP that contain various techniques (Chopra et al., 2013), namely:

1. Morphological and lexical analysis;
2. Syntactic analysis;
3. Semantic analysis;
4. Discourse integration; and
5. Pragmatic analysis.

Morphological analysis involves in-depth analysing, identifying and describing the structure of words. **Lexical analysis** requires bodies of text to be divided into paragraphs, words, and sentences. This is known as tokenisation which segments words into separate units called tokens and classifies these units based on their type (Piskorski & Yangarber, 2013). Morphological analysis can be used to extract morphological information from tokens such as a token's base form and part of speech (Piskorski & Yangarber, 2013). **Syntactic analysis,** also known as syntactic parsing, involves analysing the words in a sentence to determine the grammatical structure of the sentence. **Semantic analysis** determines the exact meaning of a section of text based on a given context. **Discourse integration** implies that the meaning of a sentence is determined by the previous sentence and it invokes the meaning of successive sentences. **Pragmatic analysis** derives the purposeful use of language in a situation. The main purpose of pragmatic analysis is to differentiate between what is said and what is actually meant. To fulfil each phase, a set of tasks must be completed. An additional technique identified by Piskorski and Yangarber (2013) that can be used with morphological and lexical analysis is called meta-data analysis. Meta-data analysis involves analysing and extracting titles, body, structure of the body, and important dates from text.

Common NLP techniques are (Collobert et al., 2011):

- Part-of-speech tagging (POS);
- Chunking;
- NER; and
- Semantic role labelling.

POS tagging labels each word in a set of text with a unique tag to indicate the word's syntactic role. Words are labelled based on English POS such as nouns, verbs, and adjectives (Collobert et al., 2011). POS tagging is a simplified form of morphological analysis as words are only tagged, not analysed to find internal structure (Indurkya & Damerau, 2010). **Chunking**, also known as shallow parsing, labels

segments of a sentence with syntactic constituents such as nouns or verb phrases (Collobert et al., 2011). In the context of NLP, **NER** involves labelling elements in a sentence into different categories such as "PERSON" or "LOCATION". **Semantic role** labelling provides a semantic role to a syntactic constituent of a sentence (Collobert et al., 2011). In addition to the NLP techniques mentioned by Collobert et al. (2011), stop-word removal is also another commonly performed NLP technique (Vijayarani et al., 2015) and parsing (Chopra et al., 2013). Stop-word removal involves removing commonly used words that are usually articles, prepositions, or pronouns. Parsing refers to determining the grammatical structure of phrases or sentences. Figure 3-6 maps the phases of NLP to the tasks of NLP.

Morphological and lexical analysis can make use of tokenisation, stop-word removal, and POS. Tokenisation can be used to separate words into tokens after which all unnecessary words can be removed using stop-word removal. Once this has been completed, POS can be applied to identify each word's syntactic role. The result of the morphological and lexical analysis will be analysed and tagged words. Syntactic analysis can then occur in which chunking can be applied to identify the grammatical structure of phrases. The result of syntactic analysis would be sentences that have their structure identified. These sentences can then be passed on for sematic analysis during which the exact meanings of the sentences can be determined. NER and CO discussed in Section 3.4 can be applied for semantic analysis. Alternatively, classification or semantic role labelling can be applied. Classification could use an algorithm such as a Support Vector Machine to determine a sentence's meaning (Collobert et al., 2011). Once the meanings of sentences have been determined, discourse integration and pragmatic analysis can occur. During these two phases, the meanings assigned to the sentences will be further analysed to determine what was said versus what was actually meant.

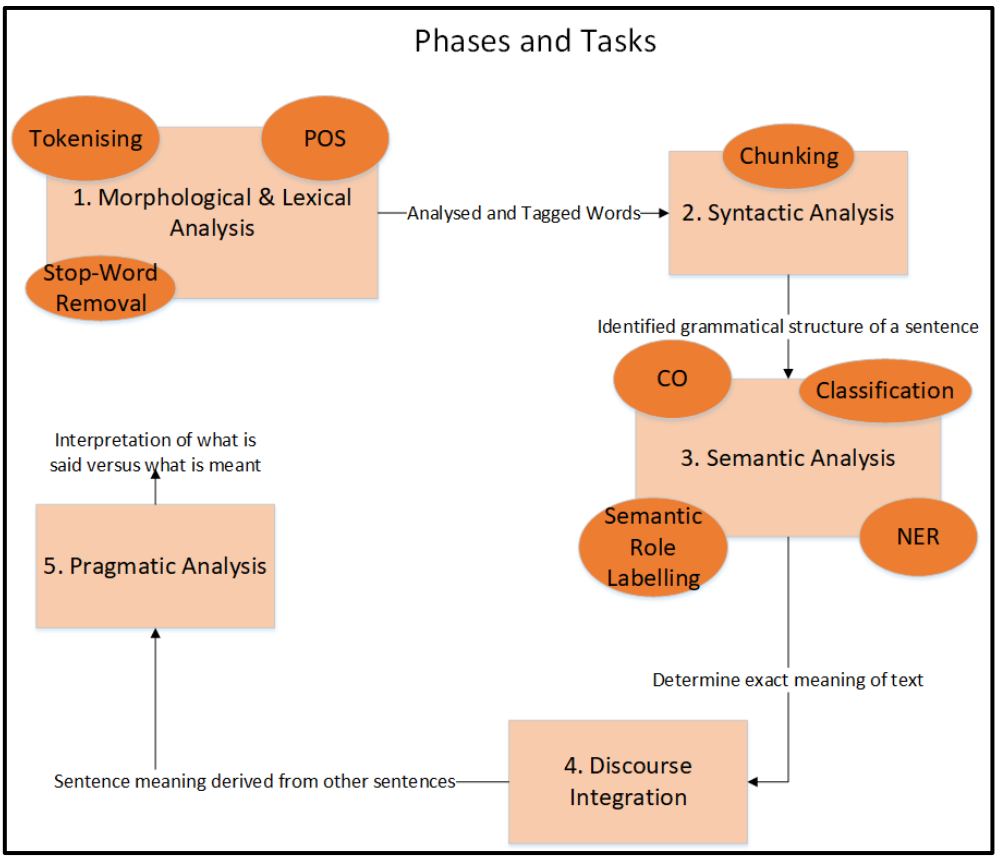


Figure 3-6: Phases of NLP (author's own work)

The phases of NLP can be applied within the legal domain to analyse and obtain meanings from text in legal cases. However, for this research only tokenisation from morphological and lexical analysis would be relevant because words in a legal case will have to be separated and then specific facts would have to be extracted. The process of implementing the NLP tasks in Figure 3-6 can be achieved through an NLP framework or toolkit.

3.5.2 Natural Language Processing Tools

Various frameworks and toolkits are available for implementing NLP. The frameworks and toolkits are available on different platforms and perform different NLP tasks. These frameworks and toolkits are summarised in Table 3-4 and Table 3-5.

	Platform			
	Java	**Python**	**Java**	**Java**
NLP Technique	**Stanford Core NLP**	**NLTK**	**Apache OpenNLP**	**GATE**
POS	✔	✔	✔	✔
Chunking	✔	✔	✔	✔
NER	✔	✔	✔	✔
Tokenisation	✔	✔	✔	✔

Table 3-4: Comparison of NLP Frameworks and Toolkits Part 1

	Platform			
	Java	**Python**	**Java**	**Java**
NLP Technique	**Stanford Core NLP**	**NLTK**	**Apache OpenNLP**	**GATE**
CO	✔	✘	✔	✔
Stop word removal	✔	✔	✔	✔
Sentence splitting	✔	✔	✔	✔
Syntactic Parsing	✔	✔	✔	✔

Table 3-5: Comparison of NLP Frameworks and Toolkits Part 2

Five open-source libraries that can be applied to various phases and tasks of NLP are (Ingersoll, 2015):

- The Stanford Core NLP Suite;
- Natural Language Toolkit (NLTK);
- Apache OpenNLP; and
- General Architecture for Text Engineering (GATE).

The Stanford Core NLP Suite is a Java Virtual Machine-based annotation pipeline framework that provides common NLP functionality (Manning et al., 2014). The framework consists of a raw text source, an annotation object, an execution of various functions, and an annotated text output. Raw text is put into an annotation object after which a series of annotator functions execute to add information to the annotator object. Once all annotator functions have executed, the annotated text can be output in the form of Extensible Markup Language (XML) or other plain forms of text. The eight annotator functions found in the framework are the following:

- Tokenisation;
- Sentence splitting;
- POS;
- Morphological analysis;
- NER;
- Syntactic parsing;
- CO; and
- Other annotators for sentiment and gender.

All annotator functions except the second annotator have been explained in Section 3.5.1. The second annotator, sentence splitting, splits up a sequence of tokens into sentences. Figure 3-7 illustrates the framework of the Stanford Core NLP Suite.

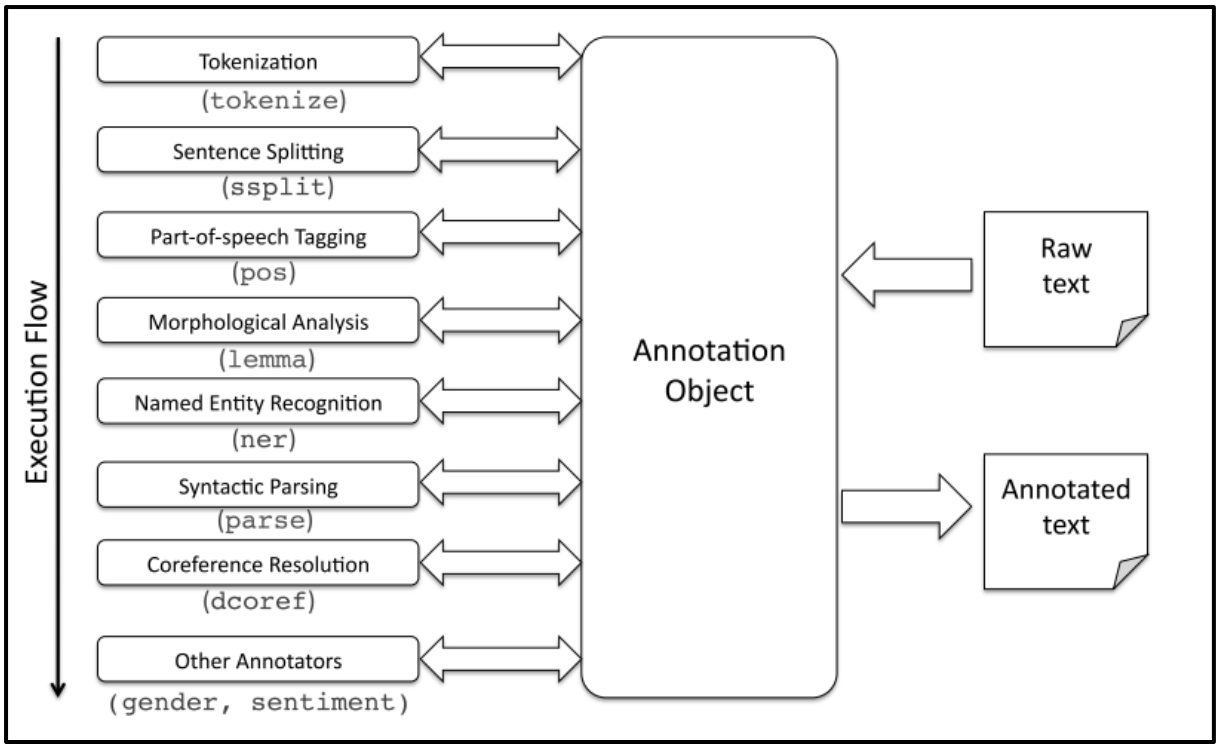


Figure 3-7: Framework of Stanford Core NLP Suite (Manning et al., 2014)

NLTK is a Python-based NLP toolkit (NLTK Project, 2017). The toolkit is designed to meet six criteria, namely (Bird & Loper, 2002):

- Ease of use;
- Consistency;
- Extensibility;
- Documentation;
- Simplicity; and
- Modularity.

The toolkit is aimed at allowing users to build NLP systems without having to spend much time on learning how to use the toolkit. The toolkit uses consistent data structures and interfaces and can easily accommodate new components. In terms of documentation, all aspects regarding the toolkit, its data structures, and its implementation are well documented. With regards to simplicity, all classes defined within the toolkit are created so that users can immediately implement them once users have completed an introductory course. In terms of being modular, the toolkit is designed to keep the interaction of different components to a minimum usage.

The toolkit provides libraries to perform the following functions (NLTK Project, 2017):

- Tokenisation;
- Stemming;
- POS;
- Parsing; and
- Semantic reasoning.

The **Apache OpenNLP** is a machine learning toolkit for NLP (The Apache OpenNLP Development Community, 2011) . The toolkit aims to be a mature toolkit for common NLP tasks and to provide many pre-built models for different languages. The toolkit consists of many components that enable the creation of an NLP pipeline. The following NLP functions are supported by the toolkit:

- Tokenisation;
- Sentence segmentation;
- POS;
- NER;
- Chunking;
- Parsing; and
- CO.

GATE is an open-source framework for creating software components that perform NLP (Cunningham et al., 2017). In addition to NLP tasks, GATE also provides an IE component called ANNIE -a Nearly New Information Extraction System. Built-in components of GATE are the following:

- Language resources;
- Processing resources that are part of the IE component;
- Gazetteers;
- Ontologies;
- Machine learning resources;
- Alignment tools; and
- Parsers and taggers;

The built-in component of interest for this research is GATE's processing resources that are part of ANNIE. The components that make up ANNIE are the following:

- Tokeniser;
- Lemmatiser;
- Gazetteer lookup;
- Sentence splitter;
- Taggers;
- Name matchers; and
- Parsers;

The developmental aspect of this research will use the programming language Python. Therefore, the NLTK toolkit can be used to implement NLP for IE. As seen in Table 3-5, NLTK supports most NLP tasks. Extraction of facts can occur once text has been processed by the NLP tasks. However, it might not be necessary to perform NLP tasks to extract facts. Extraction can occur by applying patterns onto the bodies of processed text. These patterns can be in the form of regular expressions.

3.6 Web Scraping Techniques

This section will investigate the practice of web scraping to extract information from websites. This investigation will include a comparison of the three tools that can be used to perform web scraping, namely libraries, frameworks, and desktop applications.

3.6.1 Web Scraping

Web scraping is the practice of applying techniques to automatically extract information from a website (Vargiu & Urru, 2012). It is therefore an IE technique for websites. There are multiple uses for web scraping such as price comparisons, weather data monitoring, and web data integration. For this

research, web scraping can be used to extract and process facts from legal cases. Web scraping is performed by web scrapers that look for specific information. Web scrapers focus on transforming unstructured data into structured forms that are stored in a data structure such as a database. Furthermore, web scrapers mimic the browsing interaction between web servers and a human by accessing a website and parsing the website's content to find information of interest (Glez-Peña et al., 2013). Three tools can be used to create a web scraper, namely:

- Libraries;
- Frameworks; and
- Desktop-based environments.

Libraries are used to grant access to a website by implementing the client-side of an HTTP protocol (Glez-Peña et al., 2013). The libraries may also provide parsing techniques such as HTML tree building and XPath matching, but it is not uncommon for built-in string functions to be used. Built-in string functions can be tokenisation or regular expressions. Libraries identified by Glez-Peña et al. (2013) will be investigated in Section 3.6.2.

Frameworks are a more integrative method to scrape a website as opposed to using libraries (Glez-Peña et al., 2013). Generally, libraries require integration with additional libraries to create a functioning web scraper. Frameworks on the other hand eliminate the need for additional libraries by providing all the functions required to create a web scraper. Frameworks identified by Glez-Peña et al. (2013) will be investigated in Section 3.6.2.

Desktop-based environments make use of desktop applications for web scraping (Glez-Peña et al., 2013). The desktop applications differ from libraries and frameworks as the applications cater for layman programmers. The programmers are provided with an interface to help create a web scraper. The interface allows programmers to interact with a web page and select elements to be scraped. A disadvantage of using web scraping desktop applications is the limited access to APIs which makes it difficult to embed the web scraper into other programs. Desktop applications will be investigated in Section 3.6.2.

3.6.2 Web Scraping Tools

Various libraries, frameworks and desktop applications for web scraping are identified and summarised in Table 3-6 to Table 3-8. Commonly used libraries are (Glez-Peña et al., 2013):

- Libcurl;
- WWW::Mechanize by programming language Perl; and
- Apache HTTPClient by programming language Java.

Libcurl is an open-source library that supports multiple features of the HTTP protocol. Libcurl also has bindings to multiple programming languages allowing programmers to get full use of the Libcurl library within their programming language of choice. Features of the HTTP protocol that Libcurl supports are:

- SSL certificates;
- HTTP POST;
- HTTP PUT;
- FTP uploading; and
- HTTP authentication.

The WWW::Mechanize library allows programmers to interact with weblinks and forms and requires no additional parsing. Support is provided for HTTPS, cookie management, and HTTP authentication. Java's Apache HTTPClient library emulates features of the HTTP protocol and can be combined with

HTML parsing libraries such as Jsoup. In addition to the libraries identified by Glez-Peña et al. (2013), the following libraries can be used for web scraping:

- Requests;
- Beautiful Soup 4;
- Lxml; and
- Selenium.

Requests is an HTTP library that is used to access web pages. Requests contains built-in functions to make accessing and parsing a website's content easy. The Requests library can also access APIs and post to forms (EliteDataScience, 2017). **Beautiful Soup 4** is a Python based library that pulls data from HTML and XML files. Files are converted into BeautifulSoup objects on which built-in functions can be applied to extract information (Richardson, 2015). **Lxml** is an HTML and XML parsing library that is bound for the libxml2 and libxslt C libraries (LXML, 2017). The two C libraries allow for core tasks to be completed such as parsing, serialising, and transforming (Daly, 2011). **Selenium** is used to scrape websites that are not in favour of being scraped (Marzagão, 2013). As such, Selenium is called a webdriver as it takes control of a user's browser and performs IE.

Frameworks identified by Glez-Peña et al. (2013) are Web-Harvest and jARVEST. **Web-Harvest** is a Java-based web scraping framework that uses XML to describe IE processes. Web-Harvest's various processes are made up of several pipelines that can include procedural instructions such as variable definitions, loops, and primitives to retrieve web content and clean HTML. Web-Harvest uses techniques such as XSLT, XQuery, and Regular Expressions to perform IE (Web-Harvest, 2017). **jARVEST** is a DSL Java-based framework that creates harvesters to scrape a website. Harvesters are made up of transformers that receive streams of strings and output streams of strings. jARVEST contains multiple features such as XPath Expression Matching, CSS Selector Matching, variable definitions, and looping (jARVEST, 2017). In addition to the frameworks identified by Glez-Peña et al. (2013), a framework called Scrapy can also be used for web scraping (Myers & McGuffee, 2015). **Scrapy** is Python-based and comes with an engine, scheduler, downloader, and classes. The engine controls data flow between components, whilst the scheduler receives requests and the downloader fetches webpages. Classes, known as spiders, are customised by users to parse responses and extract items. Scrapy provides an array of features such as extracting data from HTML and XML sources, HTTP support, and support for regular expressions and XPath expressions (Scrapy, 2016).

Desktop applications identified by Glez-Peña et al. (2013) are IrobotSoft, Visual Web Ripper, and Mazenda. **IrobotSoft** is a windows-based application that is scriptable, GUI-based, supports multi-threading and can output to multiple formats. Similarly, **Visual Web Ripper** is also windows-based, uses a GUI, supports multi-threading, and can output to multiple formats. However, Visual Web Ripper is limited in terms of being scriptable. **Mazenda** supports multi-threading, is GUI-based but is not scriptable.

Libraries for Web Scraping			
Library	**Domain Specific Language**	**Language**	**Features**
Libcurl	No	C+ bindings	• HTTP Post; • SSL Certificates; • HTTP PUT; • FTP; and • HTTP Authentication.
WWW::Mechanize	No	Perl	• HTTPS; • HTTP Authentication; • Cookie Management.
Apache HTTPClient	No	Java	• HTTP Protocol
Requests	No	Python	• Built-in functions to process website; and • Can post to forms.
Beautiful Soup 4	No	Python	• Access HTML and XML files; and • Convert files to BeautifulSoup objects.
Lxml	No	Python	• Allows for core processing tasks to occur.
Selenium	No	Python	• Control a browser for IE.

Table 3-6: Libraries for Web Scraping

Frameworks for Web Scraping			
Framework	**Domain Specific Language**	**Language**	**Features**
Web-Harvest	Yes	Java	• XPath; • Regular Expressions; • XLST and • XQuery;
jARVEST	Yes	Java/JRuby	• XPath Expression Matching; • Regular Expressions; • CSS Selection; • Looping; and • Variable definitions.
Scrapy	No	Python	• Regular Expressions; • XPath; • HTTP Authentication; • HTTP Compression; • HTML/XML support; and • Reusable spiders.

Table 3-7: Frameworks for Web Scraping

Desktop Applications for Web Scraping					
Desktop Application	**Platform**	**Output Formats**	**Multi-Threading**	**Scriptable**	**GUI-Based**
IrobotSoft	Windows	• Text; • CSV; • XML; and • DB	Yes	Yes	Yes
Visual Web Ripper	Windows	• CSV; • XML; • DB; and • Excel.	Yes	Limited	Yes
Mazenda	Windows	• CSV; • TSV; • XML; and • Excel.	Yes	No	Yes

Table 3-8: Desktop Applications for Web Scraping

It is evident that there are multiple technologies available to perform web scraping. Depending on the end goal, libraries, frameworks, or desktop applications can be used. Libraries can be used with other libraries to create a fully functional web scraper. However, frameworks are preferable as they provide a more integrated approach to web scraping. If an easier approach to web scraping is required, then desktop applications can be used. The only limitation to using desktop applications is that there could be limited access to APIs. For this research, web scraping using libraries can be applied to extract facts from legal cases located online. The web scraping library can interact with other technologies relating to IR and IE. The facts that are extracted by the library can then be further processed by means of Natural Language Processing to obtain meanings from the facts.

3.7 Regular Expressions

Regular expressions are specific text patterns that are used to search in bodies of text, replace text, segregate text into smaller bodies, and rearrange pieces of text (Goyvaerts & Levithan, 2009). If used correctly, regular expressions can simplify programs and text processing tasks by reducing the amount of code needed for processing. Regular expressions differ from NLP as none of the NLP phases need to be applied when working with bodies of text. This implies that a regular expression can be used directly on an unprocessed body of text. Regular expressions are implemented using finite automatons and can be divided into two categories, namely traditional regular expressions and modern-day regular expressions. Finite automatons are machines that consist of a finite amount of states that are used for memory and computation (Rabin & Scott, 1959). Traditional regular expressions originate from mathematics and computer science theory and reflect a trait called regularity. These traditional regular expressions do not support backtracking and can be implemented by using a deterministic finite automaton (DFA). Conversely, modern-day regular expressions can use backtracking and are implemented using a non-deterministic finite automaton (NFA) (Goyvaerts & Levithan, 2009). Both DFAs and NFAs can be represented using Equation 3.7-1, where both consist of a set of finite states, a set of finite input symbols, a starting state, a final state, and a transition function. The difference between a DFA and NFA is that a DFA returns a single state from its transition function while an NFA can return multiple states. This implies that DFAs can only be in one state at a time while NFAs can be in multiple states at the same time (Hopcroft et al., 2006).

$$A = (Q, E, \delta, q_0, F)$$

Equation 3.7-1: Representation of a DFA and NFA

Where:

- Q represents a finite set of states;
- E represents a finite set of input symbols;
- δ represents a transition function;
- q_0 represents a starting state; and
- F represents a final state.

Regular expressions can consist of characters from an alphabet or apply an operator to a set of argument expressions (Prasse et al., 2015). Multiple operators are available for use such as concatenation, disjunction, and the Kleene star. Additionally, characters can be specific matching symbols, meta characters, or quantifiers. Table 3-9 summarises the common matching symbols, meta characters, and quantifiers that a regular expression can consist of (Vogel, 2016).

Possible Characters of a Regular Expression					
Matching Symbols		**Meta Characters**		**Quantifiers**	
Symbol	**Meaning**	**Meta Character**	**Meaning**	**Quantifier**	**Meaning**
.	Matches any character	\d	Any digit.	*	Occurs zero or more times
^regex	Matches a regular expression at the beginning of a line	\D	A non-digit.	+	Occurs one or more times
regex$	Matches a regular expression at the end of a line	\w	A word character.	?	Occurs zero or one time.
[abc]	Matches ‘a’, ‘b’, or ‘c’	\W	A non-word character	{X}	Occurs X number of times.
[^abc]	Matches any character except for ‘a’, ‘b’, or ‘c’	\b	A word boundary	*?	The ‘?’ makes the regular expression stop at the first match
[a-d1-7]	Matches a range between the letters a to d, and numbers from 1 to 7				
X|Z	Find ‘X’ or ‘Z’				

Table 3-9: Possible Characters for a Regular Expression (Vogel, 2016)

Goyvaerts and Levithan (2009) state that various programming languages are available that support the implementation of regular expressions. The programming languages include Perl, Java, Ruby, and Python. Perl and Ruby support regular expressions as part of their language while Java and Python provide packages or modules to support regular expressions.

Table 3-10 demonstrates how Python's regular expression engine works with the string '**abcbd**' and regular expression '**a[bcd]*b**' (Kuchling, 2018):

Step	Match	Explanation
1	a	The 'a' in the string matches the 'a' in the regular expression
2	abcbd	A match is found using [bcd]* by going to the end of the string
3	Failure	A match is attempted for 'b' but the current position is at the end of the string so there is no match
4	abcb	The regular expression engine backtracks so that one less character is matched, this means the 'b' after '*' is dropped
5	Failure	The regular expression then reattempts to match 'b' but the current position is at the 'd'
6	abc	The regular expression engine backtracks again so that [bcd]* matches 'bc'
6	abcb	The regular expression engine reattempts to match 'b'. This is successful as the current position is on 'b'

Table 3-10: Example of how a Regular Expression Engine Works (Kuchling, 2018)

In the context of this research, regular expressions can be applied to extract facts from a legal case. Legal cases consist of a semi-structured format that can allow for specific regular expression patterns to be created and applied to a legal case.

3.8 Information Storage

This section investigates the Not Only SQL (NoSQL) Database options available for storing processed information. Two types of options will be investigated, namely NoSQL Graph Databases and NoSQL Document Databases. These two types of NoSQL databases are investigated as they cater for multi-structured data types and allow for large amounts of data to be easily inserted and stored (MongoDB, 2018c). NoSQL Databases differ from traditional Relational Databases as they are distributed, non-relational, flexible, and designed for large-scale data storage (Moniruzzaman & Hossain, 2013).

3.8.1 NoSQL Graph Databases

Performing IE results in the output of facts that need to be stored for easy access or use. Many fields such as science, government, and business can be modelled using graphs to understand the datasets produced from these fields (Robinson et al., 2015). The graph space is divided into two parts, namely graph compute engines and graph databases. Graph compute engines analyse large datasets primarily for offline graph analytics. Graph databases are graph-orientated databases that consist of one or many graphs to manage and perform complex queries over data (Pokorný, Valenta, & Kovačič, 2017). A graph consists of a set of vertices and edges. Vertices are called nodes and are connected by edges that define the relationship between nodes. Relationships are a key feature of graph databases as they eliminate the need to deduce connections between entities using foreign keys (Robinson et al., 2015). Graphs can be implemented using two types of structures namely, adjacency matrices or adjacency lists (Kolosovskiy, 2009). An adjacency matrix is a symmetric matrix that reflects adjacencies between the vertices or edges within a graph (Kolosovskiy, 2009). Creating an adjacency matrix is possible by means of a 2D array (Jemini, 2018). An example of a graph implemented as an adjacency matrix is seen in Figure 3-8 and Figure 3-9 (Singh & Sharma, 2012). Graph, G, is represented by four nodes, namely X, Y, Z, and W in Figure 3-8.

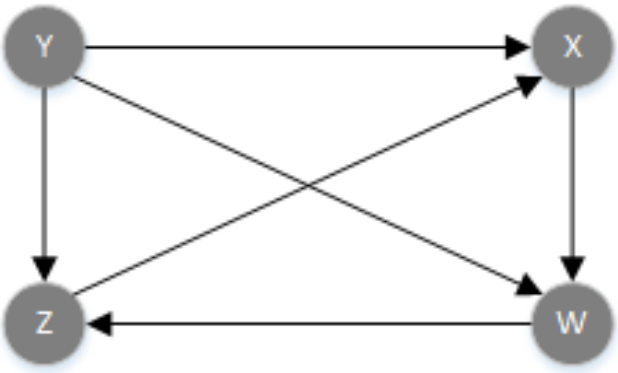


Figure 3-8: Example of a Graph G

The adjacency matrix, A, of graph G is represented by Figure 3-9. Adjacency matrix A represents the nodes from graph G and indicates connections with a 1 and no connections with a 0.

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 3-9: The Adjacency Matrix of Graph G

An adjacency list represents all edges in a graph as a list (Singh & Sharma, 2012). An adjacency list can be implemented by using nodes and linked lists, or a dictionary if it is being implemented in Python (Programiz, n.d.). If a node and linked list are used, then a node represents a vertex and the linked lists represents all the nodes connected to the vertex. If a dictionary is used, then vertices are represented as the keys and the values are represented as set of nodes. An example of a graph implemented as an adjacency matrix is seen in Figure 3-10 and Figure 3-11 (Singh & Sharma, 2012). Graph B in Figure 3-10 contains nodes A, B, C, D, and E.

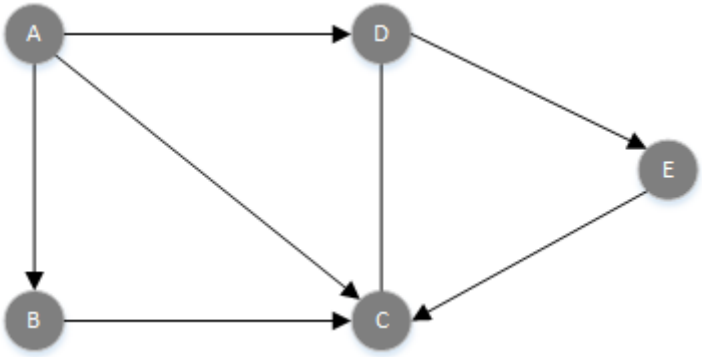


Figure 3-10: Example of a Graph B

Figure 3-11 represents Graph B implemented as an adjacency list. The adjacency list is interpreted as "Node A is connected to nodes B, C, and D".

Node	Adjacency List
A	B, C, D
B	C
C	
D	C, E
E	C

Figure 3-11: The Adjacency List of Graph B

A commonly used graph model is a Labelled Property Graph that consists of nodes and relationships (Robinson et al., 2015). Nodes can have multiple labels and contain properties. Relationships are directed, named, have a start and end node, and can also contain properties. Figure 3-12 illustrates a Labelled Property Graph within a social network context. In a social network context, users can follow each other and view current and previously sent messages. Figure 3-12 consists of three 'User' labelled nodes and three 'Message' labelled nodes. The edges represent the relationship between nodes for example, 'Ruth' follows 'Billy' and the recent message by 'Ruth' is represented by 'Message' node with property 'id: 101'.

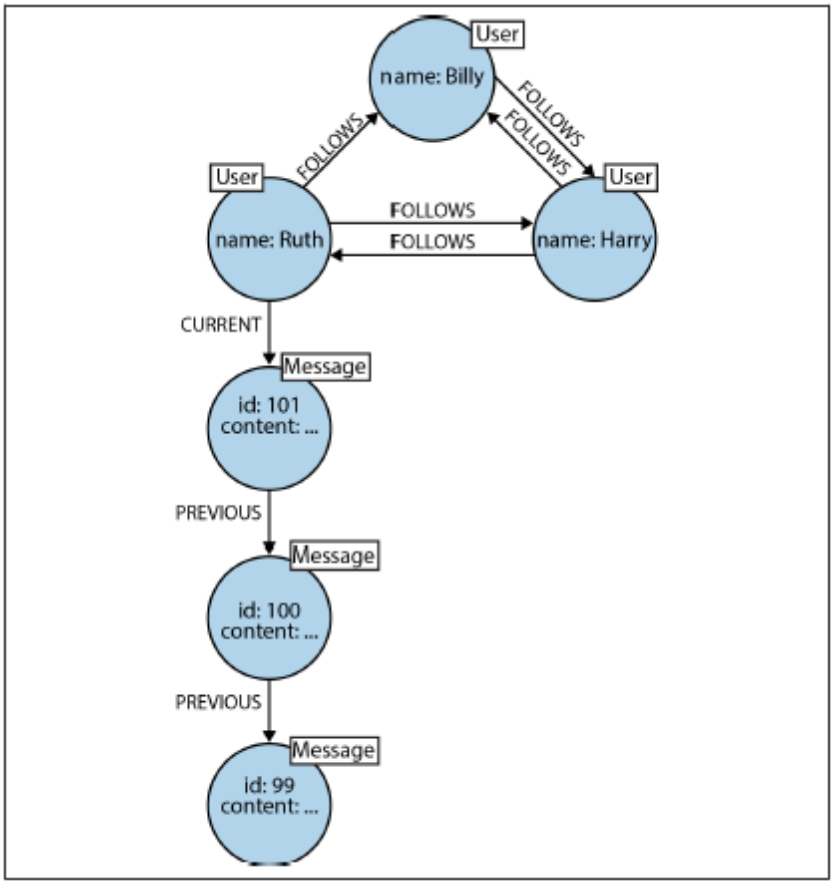


Figure 3-12: A Labelled Property Graph within Social Network Context (Robinson et al., 2015)

Once a graph database has been implemented, methods to create, read, update, and delete data are applied to the graph database. By creating a simple abstraction of nodes and relationships into a graph database one ends up with an accurate model that represents a problem domain. In addition to

simplicity (Robinson et al., 2015), various characteristics make graph databases advantageous to traditional relational databases such as performance, flexibility, and agility (Zhang, 2017).

Robinson et al. (2015) elaborates on the advantages of graph databases listed by Zhang (2017). In terms of **performance**, graph databases bring about an increased performance when working with connected data as opposed to relational databases. Increased performance is seen when querying the database. Relational databases' join-intensive query performance becomes poor as a dataset increases whilst performance in a graph database remains constant. A possible reason for query performance in a graph database remaining constant is because queries are localised to a portion of the graph resulting in execution times that are constant or faster. In terms of **flexibility,** graphs enable new data to be easily added. New data can be in the form of relationships, nodes, labels, or subgraphs and do not interrupt existing query and application functionality. This results in less migration, overhead, and risk. Relational databases on the other hand require more work. In terms of **agility,** graph databases allow for easy development and maintenance of systems. Systems can be evolved in a controlled manner due to the systems being schema-free. In addition to the advantages discussed, graph databases also allow data elements that have complicated relationships to be easily handled. This is possible by using edges to connect elements instead of foreign keys.

A commonly used vendor for implementing a graph database is Neo4j (MongoDB, 2018c). Neo4j makes use of linked lists to implement the graph structure (De Marzi, 2012). Linked lists are used to represent and store nodes, relationships, and properties. Properties use key-values and point to the next property in the linked list. Nodes and relationships reference the first property associated with it. Nodes also represent the first relationship in the relationship linked list. Each relationship references a start and end node.

The advantages mentioned by Robinson et al. (2015) and Zhang (2017) make graph databases suitable for storing information regarding legal cases. A graph database can therefore be used to contain all extracted information obtained from the IE processes illustrated in Figure 3-5 and can be queried to translate facts to output. The use of a graph database would eliminate the need to setup and manage a relational database containing tables. Instead of using tables to represent legal cases, nodes can be used to represent information obtained via IE.

3.8.2 NoSQL Document Databases

A document database is a type of NoSQL database that stores data in the form of documents that can be grouped together to form collections (MongoDB, 2018c). Documents can be viewed as objects that contain typed values such as strings, binary values, or arrays. Unlike relational databases that store data across multiple tables and columns, document databases store data in a single document. This helps eliminate the need for JOIN operators.

Data can be stored in three types of structures namely, XML, Javascript Option Notation (JSON), or Binary JSON (BSON) (Moniruzzaman & Hossain, 2013). JSON objects store data as strings and numbers while BSON objects are an extension of JSON objects that allow representation of additional types such as int, long, floating point, and date (MongoDB, 2018b). The data is stored as key-value pairs where both the keys and values are searchable. In addition, document databases are schema-free.

Document databases do not require any schemas to be predefined before data can be added to the database (Parmar & Roy, 2018). The schema is automatically created as data is added. This lack of a predefined schema provides developers with more flexibility than using relational databases as they do not have to force-fit new types of application data to the database. A key characteristic of a document database is that documents can contain embedded documents and lists containing multiple values. This eliminates the need to join multiple sets of data together as would be the case in a

relational database. Figure 3-13 illustrates an example of how a document is stored as a JSON object that has a list embedded within its document. In the example, the document contains a key called 'PreviousPositions' that contains a list of documents.

```
{
    firstName: "Bob",
    lastName: "Wilson",
    positionTitle: "Manager",
    officeNumber: "2-130",
    officePhone: "555-222-3478",
    hireDate: "1-Feb-2010",
    terminationDate: "12-Aug-2014"
    PreviousPositions: [
        {    \position: "Analyst",
         StartDate:"1-Feb-2010",
        endDate:"10-Mar-2011"
        } {
            position: "Sr. Analyst",
            startDate: "10-Mar-2011"
            endDate:"29-May-2013"
        } ]
}
```

Figure 3-13: Embedding Data into a Document (Parmar & Roy, 2018)

Document databases are ideal for storing and managing big-data sized collections of data regarding text documents, email messages, or XML documents (Moniruzzaman & Hossain, 2013). Additionally, document databases are also good at storing conceptual documents such as a representation of a database entity, as well as semi-structured data that would normally require relational databases to use many nulls for missing values. Commonly used vendors for implementing document databases are CouchDB and MongoDB (Parmar & Roy, 2018). CouchDB stores data as JSON objects while MongoDB stores data as BSON objects. Figure 3-14 illustrates an example of a BSON object stored in MongoDB (MongoDB, 2018a).

```
{
   _id: ObjectId("5099803df3f4948bd2f98391"),
   name: { first: "Alan", last: "Turing" },
   birth: new Date('Jun 23, 1912'),
   death: new Date('Jun 07, 1954'),
   contribs: [ "Turing machine", "Turing test", "Turingery" ],
   views : NumberLong(1250000)
}
```

Figure 3-14: Example of a BSON object stored in MongoDB (MongoDB, 2018a)

Figure 3-14 is different from Figure 3-13 as it supports the additional types discussed above such as date and long. The date type is represented by the keys "birth" and "death" while the long type is represented by the key "views". Table 3-11 provides a comparison of Graph Databases and Document Databases.

Comparison of Graph Database and Document Database		
Characteristic	**Graph Database**	**Document Database**
Underlying structure	Adjacency matrixes or Linked Lists	JSON, BSON, or XML
How data is stored	Nodes connected by edges	Documents containing key-values
Schema	No predefined schema needed	No predefined schema needed
Relationships	Uses edges to create relationships amongst data	No relationships used
Embedding data	Not supported	Supports embedding data
Uses	When there is an interest in the relationship amongst data	Storing and managing big-data sized collections of literal documents

Table 3-11: Comparison of Graph Database and Document Database

It can be seen that while graph and document databases are both NoSql databases and do not require a predefined schema, they are different when it comes to the structures used to represent them, how data is stored, and have different uses. IE can be used to extract and store facts that can later be retrieved for actions such as query-independent ranking.

3.9 Query-Independent Ranking Algorithms

The World Wide Web is rapidly growing and is massive, diverse, and unstructured (Choudhary & Burdak, 2012). As a result, the number of queries submitted by users is also increasing. Therefore, IR and IE systems require efficient methods to process queries to return relevant information to users. Four query-independent ranking algorithms can be used to further process and return relevant results to a user, namely:

- PageRank;
- Weighted PageRank;
- Hyper-link Induced Topic Search; and
- Focused Rank.

3.9.1 PageRank Algorithm

The PageRank algorithm evaluates the importance of a webpage based on a webpage's link structure (Gleich, 2014). The algorithm is both recursive and iterative (Choudhary & Burdak, 2012). PageRank is based on the concept that if a page has important links pointing towards it, then the links of this particular page that point towards other pages will result in the particular page being considered as important. When determining a page's rank, the algorithm considers all back-links. If the addition of all the ranks of back-links are large, then a page is assigned a large rank value. The algorithm uses a damping factor to prevent other pages from having a large influence on a page's rank. The damping factor is a value between zero and one but is usually set to 0.85. Furthermore, a page's rank is divided evenly amongst its outgoing links. Due to the recursive and iterative nature of the PageRank algorithm, computation can take long if there are a large number of pages for the PageRank algorithm to process (Prakash & Kumar, 2015).

The PageRank algorithm is represented by Equation 3.9-1.

$$PR(A) = (1 - d) + (\frac{PR(T1)}{C(T1)} + \cdots + \frac{PR(Tn)}{C(Tn)})$$

Equation 3.9-1: The PageRank Algorithm

The variables of Equation 3.9-1s represent the following:

- PR(A) – the PageRank of page A;
- PR(Ti) – the PageRank of pages Ti which link to page A;
- C(Ti) – the number of outbound links on page Ti; and

- D – the damping factor value between 0 and 1.

3.9.2 Weighted PageRank Algorithm

The Weighted PageRank algorithm is a modification of the original PageRank algorithm. WPR is an iterative algorithm that assigns a ranking based on a page's popularity (Choudhary & Burdak, 2012). Popularity of a page is determined by calculating the weight of a page's outgoing and incoming links. Popular pages are assigned higher rank values. Furthermore, ranks are not divided equally amongst a page's outgoing links unlike the original PageRank algorithm.

The formula for the Weighted PageRank algorithm is shown in Equation 3.9-2.

$$WPR(n) = (1 - d) + d \sum_{m \in B(n)} WPR(m)\, W^{in}(m,n) W^{out}(m,n)$$

Equation 3.9-2: The Weighted PageRank Algorithm

Equation 3.9-3 and Equation 3.9-4 are used to calculate weight values of incoming and outgoing links:

$$W^{in}(m,n) = I_n \sum_{P \in Re(m)} I_p$$

Equation 3.9-3: Calculate weight of incoming links

$$W^{out}(m,n) = O_n \sum_{P \in Re(m)} O_p$$

Equation 3.9-4: Calculate weight of out-going links

Equation 3.9-3 represents the number of incoming links with respect to pages n and p. $R_{e(m)}$ represents all reference pages of page m.

Equation 3.9-4 is computed the same as Equation 3.9-3 to determine the outgoing link's weight.

A limitation of the Weighted PageRank algorithm is that the algorithm only considers the link structure of a page and not the page's content. This limitation can result in irrelevant pages being returned (Kumari, Gupta, & Dixit, 2014). The original PageRank algorithm and Weighted PageRank algorithm can be compared by analysing the resultant webpages of a given query. Based on a user's query, the resultant webpages can be categorised into four categories, namely (Jain, Sharma, Dixit, & Tomar, 2013):

- Very relevant pages (VR);
- Relevant pages (R);
- Weakly relevant pages (WR); and
- Irrelevant pages (IRP).

VR pages contain important information in relation to a user's query whilst R pages are relevant but do not contain important information with regards to a user's query. WR pages can contain a query's keywords but does not have relevant information whilst IRP contain no relevant information and keywords from a query.

Due to PageRank and Weighted PageRank producing a ranked list of pages in a particular sorting order, a relevancy rule can be applied to calculate the relevancy value of all pages produced by the

algorithms. The relevancy rule assigns a value to a page based on the page's category and position in the ranked list. Equation 3.9-5 is used to calculate the relevancy value of a page:

$$k = \sum_{i \in R(p)} (n - i) * Wi$$

Equation 3.9-5: Calculate relevancy of a page

Equation 3.9-5 has the following variables:

- i – the i^{th} page in the list R(p);
- n – represents the first n pages chosen from the list R(p); and
- W_i – represents the weight of the i^{th} page.

3.9.3 Hyper-link Induced Topic Search (HITS) Algorithm

The Hyper-link Induced Topic Search (HITS) is an iterative algorithm that views the world wide web as a directed graph that contains two types of pages, namely hubs and authorities (Jain et al., 2013). Hubs are pages that act as resource lists and authorities are pages that contain important content. A good hub points to many authoritative pages. A good authority page is pointed to by many good hubs that contain pages of the same content. HITS has two steps, namely:

- A sampling step; and
- An iterative step.

The sampling step collects a set of relevant pages for a given query whilst the iterative step uses the output of the sampling step to find hubs and authorities.

Equation 3.9-6 and Equation 3.9-7 are used to calculate the weight of a hub and the weight of an authority:

$$H_p = \sum_{q \in B(p)} Aq$$

Equation 3.9-6: Calculate weight of hub

$$A_p = \sum_{q \in B(p)} Hq$$

Equation 3.9-7: Calculate weight of authority

The HITS algorithm has four constraints, namely (Jain et al., 2013):

- Hubs and authorities;
- Topic drift;
- Automatically generated links; and
- Efficiency.

In terms of hubs and authorities being a constraint, it is not easy to differentiate between hubs and authorities as many pages can serve as both hubs and authorities. With regards to topic drift, some results produced are not relevant to the query due to equivalent weights. In terms of automatically generated links, the HITS algorithm gives equal importance for automatically generated links which may be irrelevant to a user's query. The HITS algorithm is not efficient for real-world application due to the above-mentioned constraints.

3.9.4 Focused Rank Algorithm

The Focused Rank algorithm is based on the PageRank Algorithm and the focused web surfers model. The focused web surfers model states that a PageRank of a node is proportional to the probability of a node being reached by a user randomly going through a graph. Equation 3.9-8 represents how a preferable path is presented to the user (Krapivin & Marchese, 2008).

$$Pi = (1 - d). \sum_{i \neq j}^{j \in D} Pj. s(j|i) + \frac{d}{N}$$

Equation 3.9-8 Determine Preferable Path

Equation 3.9-8 has the following variables:

- Pi – a Page P;
- (1-d) – the damping factor;
- S(j|i) – probability to follow the reference I being at place j. S may be arbitrary.

Variable s(j|i) from Equation 3.9-8 can be calculated as follows:

$$s(j|i) = \frac{C(i)}{\sum_{k \in D} C(k)}$$

Equation 3.9-9 Determining Variables

No further information on the Focused Rank algorithm could be found.

Table 3-12 provides a summary of the ranking algorithms that were investigated. The PageRank algorithm is highlighted as it will be the most suitable algorithm to use for query-independent ranking.

Criteria	**Query-Independent Ranking Algorithms**			
	PageRank	**Weighted PageRank**	**HITS**	**Focused Rank**
Ranking	Based on a page's link structure	Based on a page's popularity	Based on the weight of hub and authority	Dependent on a user reaching a page randomly
General	Page's rank is divided evenly amongst all outgoing links; Recursive and iterative; and Uses a damping factor.	Page's rank is not divided evenly amongst all outgoing links	Iterative and views world wide web as a directed graph containing hubs and authorities	Based on PageRank algorithm and the Focused Surfers model
Limitations	Computation can take long if there are too many pages	Only considers link structure of a page, not a page's content. Less relevant pages returned	Has too many constraints making it inefficient for real-world application	Limited information available

Table 3-12: Comparison of Query-Independent Ranking Algorithms

3.10 Frameworks and Systems in the Legal Domain

An analysis of studies conducted within the legal domain and systems designed for the legal domain is presented in this section.

3.10.1 ROSS and IBMs WATSON

ROSS is an intelligence tool for supporting legal research activities which is built upon a legal artificial intelligence framework called Legal Cortex in conjunction with IBM's Watson technology (Houlihan, 2017). ROSS makes use of NLP and machine learning capabilities to identify appropriate legal authorities to specific questions. This implies that ROSS is used for IE and knowledge acquisition as mentioned in Section 3.5.1. ROSS was tested by Blue Hill Research, an organisation that creates research programs to assess artificial intelligence solutions in real-world legal use cases. The purpose of the test was to determine the quality of ROSS in terms of:

- IR;
- User confidence and usability;
- Research efficiency; and
- Business value and return on investment (ROI).

The test consisted of 16 legal researchers who were provided with a set of questions that model real-world questions faced daily by legal practitioners. The test used ROSS to supplement traditional legal research tools such as Boolean searching and natural language searching. The legal researchers were divided into groups and assigned a legal platform to perform the test on. The legal platforms used were Westlaw and LexisNexis. Each group was constrained by the type of tool they could use, namely:

- Boolean search – this group of researchers could only use keyword searching that was narrowed by Boolean logic;
- Natural language search – this refers to parsing a query that is entered in plain English, into a search algorithm to identify content based on the query. This group of researchers could only search by using plain English;
- ROSS and Boolean search – this group of researchers had to use ROSS and keyword searching; and
- ROSS and natural language searching - this group of researchers had to use ROSS and natural language searching.

In terms of IR, ROSS outperformed Boolean searching and natural language searching. ROSS returned a higher percentage of relevant authorities, relevant results, and had a better normalised discounted cumulative gain. With regards to user confidence and usability, participants were required to complete a survey after completing their questions. The results of the survey revealed that ROSS scored the highest for both usability and confidence. For research efficiency, the time taken to complete all questions was observed. Time taken was divided into time spent on research, writing answers, and unused time. the results indicated that when ROSS was used as a supplement to Boolean searching, research time was 36.5 minutes and 36.7 minutes for ROSS supplementing natural language searching. Research times for Boolean and natural language searching without ROSS supplementation were 52.3 minutes and 47.2 minutes respectively.

Overall, the results from the test indicated that when ROSS was used with Boolean searching and natural language searching there were improvements as opposed to using tools without ROSS. The improvements were in:

- Research time;
- Improved identification of relevant authorities;

- Less non-relevant results; and
- Improved prioritised placement of relevant authorities in search results.

3.10.2 Exploratory Analysis of Legal Documents Using Unsupervised Text Mining Techniques

This study conducted by Wagh (2014), proposes the application of an unsupervised text mining technique called clustering to group legal documents to improve searching for legal documents. Clustering is the process of partitioning objects into groups called clusters. Clustering is unsupervised as its aim is to reveal hidden structures within a set of data and does not make use of any input parameters (Cornuéjols, Wemmert, Gançarski, & Bennani, 2018). Generally, legal information is categorised under various headings in a semi-structured manner that can be used to quickly interpret law. Although many online legal databases provide access to such information, the retrieval is Boolean based, and it is only possible to access the information by searching for keywords. It is for these two reasons that Wagh (2014) believes clustering can be utilised to improve the quality of the information retrieved. Wagh (2014) used an online legal database in India called Manupatra to download 47 judgements based on the search query “patents act”. The layout of the judgements from the Manupatra database is different to what is used in the ALL SA judgements. The judgements from Manupatra are divided into the following different sections:

- Catchwords;
- Date of the judgment;
- Details about the court and the bench;
- Appellants;
- Respondent;
- Judges;
- Subject (categorisation viz civil);
- Rules/Order;
- Cases Referred;
- Disposition;
- Case Notes (Abstract of the case); and
- Detailed judgment given by the court.

However, for the study only catchwords and case notes were considered for clustering analysis. The documents were identified using a set of 15 to 25 catchwords. The documents were then divided by catchwords and case notes. Catchwords and case notes were processed individually. The methodology consisted of four processes (Figure 3-15), namely:

1. Data collection;
2. Data pre-processing 1;
3. Data pre-processing 2; and
4. Grouping the documents.

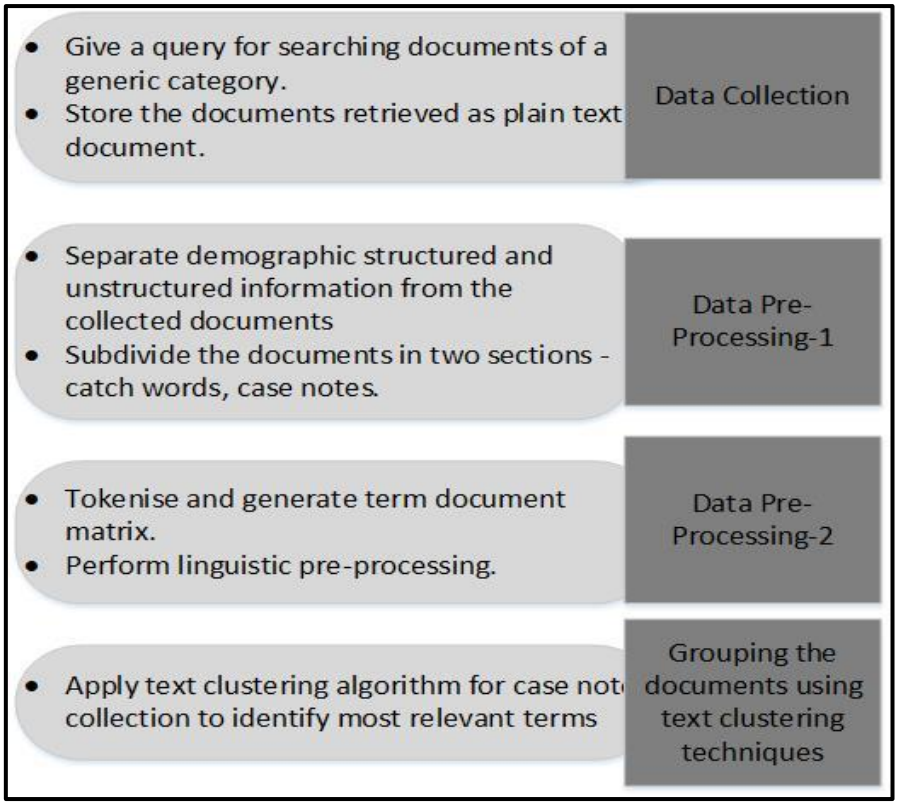


Figure 3-15: Methodology followed by Wagh (2014)

During **data collection**, documents were retrieved based on a given query for a generic category. The documents retrieved were then stored as plain text. During the **data pre-processing 1** phase, structured and unstructured information in each document were separated. The documents were then divided by catchwords and case notes. **Data pre-processing 2** involved the application of linguistic techniques such as tokenisation and creating a term document matrix. In the last phase, **grouping the documents**, a text clustering algorithm, called spherical kmeans (skmeans), was applied to the case notes to identify relevant terms. The results obtained from running the skmeans algorithm for two different cluster sizes resulted in clusters with related documents. The number of clusters produced on both runs were three and four respectively. Two shortcomings were identified in this study. The first shortcoming was that many of the sections such as cases referred to were not processed. Processing the other sections could contribute to better results. The second shortcoming was that no Information Storage process was discussed. The author only stated that the documents retrieved would be stored as plain text, but no discussion was provided on where the documents would be stored.

3.10.3 Automating Legal Research through Data Mining

Manually performing legal research is a time-consuming process. Legal researchers have the option of using two methods, namely catalogues and search engines (Firdhous, 2010). It is common to see a combination of these two methods, which is then known as portals. However, finding information on either one of the two methods still produces unsatisfactory results. Keyword searching is commonly used in search engines but often results in many false returns. The study conducted by Firdhous (2010) presents a methodological framework to automate the process of identifying and retrieving appropriate information to support legal decision-making. The framework consists of a combination of multiple text mining techniques. Firdhous' (2010) framework uses a term-based text mining system and a vector space model for developing the framework. The architecture of the methodology is illustrated in Figure 3-16:

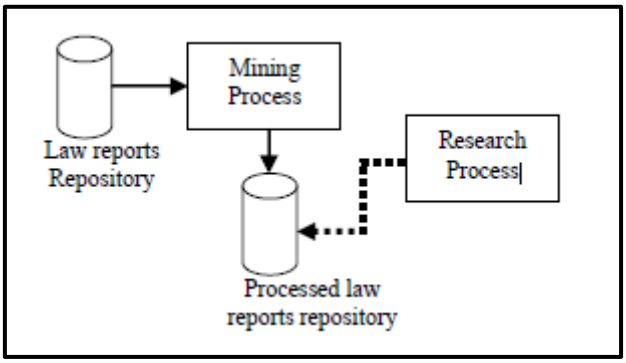


Figure 3-16: Architecture of Firdhous' (2010) Proposed Framework

The architecture of Firdhous' (2010) proposed framework consists of four processes, namely:

- A mining process;
- A research process;
- A law reports' repository; and
- A processed law reports; repository.

The mining process was applied to an entire collection of law reports in a repository. The law reports used consist of a head and detail section. For Firdhous' (2010) study, only the head section was processed. During the **mining process**, each law report was analysed and information that could be used for legal research was recorded in a processed law reports repository. The mining process consists of linguistic pre-processing that requires tokenisation and part of speech tagging. Once linguistic pre-processing was completed, term generation and term weighting were applied. Term generation produced a set of terms that were associated with a law report whilst term weighting assigned each term found a specific score to indicate its importance in relation to a goal. Once all law reports were processed, the research process began using the processed law reports repository. During the **research process**, the input received from the user was analysed and the required information was extracted and compared to all legal reports to identify matching reports. When evaluating the framework, only Fundamental Rights cases that were filed at the Supreme Court of Sri Lanka were used. Firdhous (2010) states the reasons for using only these cases are that it is easy to find records about Fundamental Rights and the Supreme Court's decisions are binding upon lower courts in Sri Lanka. These cases were put into the law reports repository and evaluation continued by using different user input text as search text. The results obtained from the evaluation indicated a high accuracy of reports retrieved. The most relevant case record had the highest similarity score and was located at the top of all other returned cases for the given query. Furthermore, the results remained

the same when the search text was changed without altering its meaning. The methodology used by Firdhous (2010) can help guide in recommending the MAC but an additional process for ranking the data will be required.

3.10.4 Wagh vs Firdhous

Sections 3.10.2 and 3.10.3 investigated research conducted within the legal domain by Wagh (2014) and Firdhous (2010). Table 3-13 compares the two studies along with the concepts investigated in these studies.

Author	**Wagh (2014)**	**Firdhous (2010)**	**Comparison**
Steps			
1	Data Collection from Manupatra Database	Populate Law Reports' Repository with Sri Lankan Fundamental Rights Cases	Both studies require data to be collected from a source
2	Data Pre-Processing 1: separate structured and unstructured information and divide documents into catch words and case notes	Mining Process: analyse cases and store in a new processed repository. Apply linguistic techniques (tokenisation and POS). Then apply term generation and term weighting	Wagh (2014) performs processing in two separate steps whilst Firdhous (2010) processes in one step. Both studies use the NLP techniques tokenisation and POS
3	Data Pre-Processing 2: Apply linguistic techniques (tokenisation) and create a term document matrix	Research Process: receive user input and return relevant results	Wagh (2014) continues processing data whilst Firdhous (2010) starts providing input and returns information
4	Grouping Documents: apply clustering algorithm		Wagh (2014) applies clustering algorithm to obtain results whilst Firdhous has completed all steps
5	Evaluation and Results: applied technique to 47 judgements. The clusters produced similar results	Evaluation and Results: created a Java based system. High accuracy results were achieved for precision and recall	Both studies obtained results that were satisfactory to the authors

Table 3-13: Comparison of Wagh's (2014) and Firdhous' (2010) Research

Both researchers' aim was to improve the search process for finding information to aid in legal decision making. Wagh (2014) proposed an unsupervised text mining technique called clustering whilst Firdhous proposed a framework to improve searching for information. Wagh (2014) used a four-step methodology whilst Firdhous (2010) used a framework that consisted of three processes. An inspection of the methodology and framework revealed that while both use different techniques, the underlying processes followed are similar. The first step of Wagh's (2014) methodology is identical to Firdhous' (2010) first process as both require data to be collected from a source. Wagh's (2014) second and third step involves processing the collected data. During the second and third steps, unstructured and structured information is separated, and linguistic techniques are applied. Similarly, the second process of Firdhous' (2010) framework processes the collected data. The second process separates useful data into a new repository after which linguistic techniques are applied. The fourth step of Wagh's (2014) methodology is to group and return documents based on a clustering algorithm.

Firdhous' (2010) third process requires the researcher to provide input to extract the required information.

Wagh (2014) evaluated the proposed technique by applying the technique to 47 judgements retrieved by using the query "patents act". The clusters produced contained similar documents (Section 3.10.2). Firdhous (2010) evaluated the framework by creating a Java-based prototype system that made use of libraries and object orientation. The results from the evaluation revealed a high accuracy rate for both precision and recall. Results for precision revealed that 93% of the time the most relevant case was found at the top of the returned information. Results for recall revealed that 88% of the time a different set of case records were returned to the user. The methodology used by Firdhous (2010) can be used as a guide in creating the prescriptive model for recommending the MAC. The prescriptive model would require input from a user, perform IE instead of linguistic techniques, have a new repository created and populated, and query the new repository. The research conducted by Wagh (2014) will not apply in creating the prescriptive model as no text mining will be needed to aid in recommending the MAC.

3.10.5 Legal Domain Software

This section will investigate the software available for use in the legal domain. Four types of software were identified, namely RAVN Applied Cognitive Engine, Equivo's Zoom System, Beagle AI System, and eBravia.

3.10.5.1 RAVN Applied Cognitive Engine (ACE)

RAVN ACE is an AI platform that supports applications that automatically organise, discover and summarise important information from documents and unstructured data. RAVN ACE utilises a combination of information processing in the form of IE, NLP and semantic technologies to connect to and work through all sources of information kept on a system (RAVN Systems, 2016). Sources of information can be document management systems, online repositories, customer relationship management systems, shared files, and content management systems.

RAVN ACE has three applications that can be applied to the legal domain. These three applications are (RAVN Systems, 2016):

- RAVN Extract;
- RAVN Connect Enterprise; and
- RAVN Refine.

RAVN Extract can be used to summarise, analyse, and perform IE. The areas where RAVN Extract can be used are:

- Contract analysis;
- Due diligence;
- Real estate; and
- Financial documents.

With regards to **contract analysis**, RAVN Extract is used to analyse contracts and detect anomalies or key points of information. An example of how legal organisations can use RAVN Extract is to provide clients with contract analysis and make use of it in assessments that have risk and compliance issues. With regards to **due diligence**, RAVN Extract automates due diligence by uploading documents and then applying clustering techniques to the content to place similar documents in a class. Documents include contracts and portable document formats. With regards to **real estate**, RAVN Extract can be applied to commercial real estate data extraction, title deed extraction, and to the sale of shopping

centres. In terms of commercial real estate data extraction, RAVN Extract can be used to analyse an organisation's competitor marketing brochures and extract key points of information such as tenant names, prices, and important dates. A similar process is followed to extract title deeds. In terms of the sale of shopping centres, RAVN Extract makes use of clustering techniques to produce a visual map of shopping clusters. The visual map indicates a shop's financial value based on the shop's lease within the shopping centre. With regards to **financial documents**, RAVN Extract can be applied to various financial agreements to extract the agreement's structure. The structure generally includes clauses and terms that might have become outdated over time. Once the first part of extraction is completed, the application then extracts key definitions from the contract and then exports the data into various systems or Excel spreadsheets.

RAVN Connect Enterprise is used to locate, capture, and manage knowledge and experience produced by an organisation. This application identifies implicit and explicit links between data and people using a graph database. **RAVN Refine** is used to clean, categorise, and store data. Furthermore, RAVN Refine applies various controls and policies to deal with data retention, duplicate data, and remove unnecessary data.

3.10.5.2 Equivo's Zoom System

Equivo is a software development company that creates text analytic software. Their platform, called Zoom, is specifically aimed at the e-discovery process within the legal domain. Zoom organises large collections of documents whilst quantifying and visualising the decision space. Zoom uses six types of applications to analyse data, namely (Equivo, 2012):

- Near-duplicates;
- Email threads;
- Language detection;
- Search;
- Themes; and
- Relevance.

Near-duplicates is a clustering process in which similar documents are grouped together. Using near-duplicates allows for similar documents to be grouped together without the user having to accidently discard the other document. **Email threads** is used to obtain and reconstruct all email conversations and identify unique emails within a collection. **Language detection** is used to identify the main language that a document is written in. **Search** allows for data to be explored so that lawyers can familiarise themselves with a case before relevance training can begin. The **Themes** application analyses documents within a collection and creates thematic connections between the content, and then presents all themes detected in a list for a user to drill-down into and find documents. The **Relevance** application organises a group of documents based on the documents' relevance scores. This application must first be trained by a human before it can be used. The applications for email threads, and themes make use of NLP. Search and relevance applications make use of IR. Additionally, themes also make use of IR to present detected themes.

3.10.5.3 Beagle AI System

Beagle AI is an AI platform aimed at automatic contract analysis. Part of the Beagle AI framework uses NLP by tagging units of text known as clauses. The framework uses a binary classifier to tag clauses. Four solutions are offered by Beagle AI, namely (Beagle Inc., 2018) :

- Contract Analysis;
- Licence Analysis;

- Corporate Compliance; and
- Regulations and Law.

The **Contract Analysis** solution analyses a contract and highlights information related to parties, parties' responsibilities, and liabilities. A contract is displayed by means of graphs and charts. The **Licence Analysis** solution analyses licenses and provides feedback based on the analysis. The **Corporate Compliance** solution analyses documents and compares these documents to an organisation's corporate rules. The **Regulations and Law** solution is aimed for team members in an organisation. The solution analyses contracts, agreements, and corporate policies to create a database. This database is then used to link with team members' experience to aid in making decisions.

3.10.5.4 eBravia

eBravia is an organisation that specialises in contract analysis. eBravia makes use of IE. The solutions offered by eBravia are (eBravia, 2018):

- Contract Analyser;
- Diligence Accelerator;
- Lease Abstractor;
- Bespoke.

The **Contract Analyser** stores, retrieves, and analyses contracts. With regards to analysis, the Contract Analyser extracts information from current and legacy contracts. Users can then populate the extracted information into a database, Microsoft Excel, or Microsoft Word. The Contract Analyser makes use of supervised machine learning to recognise various concepts within a contract. The **Diligence Accelerator** solution aims to mimic the due diligence process by extracting key concepts from legal documents and then populating specific templates. Supervised machine learning is also applied to the Diligence Accelerator solution. The **Lease Abstractor** solution is aimed at people who deal with real estate. The Lease Abstractor aims to automate the lease abstraction process to reduce time and costs that are usually associated with lease abstraction. Similar to Contract Analyser and Diligence Accelerator, the Lease Abstractor also uses supervised machine learning to recognise concepts and key words within leases. **Bespoke** is a solution aimed at users who want to perform customised analysis on their contracts. Existing eBravia software is customised to cater to a user's specific needs. Customisation can be performed by eBravia employees or by the user. Table 3-14 summarises the systems analysed with their features and the techniques used.

Systems in the Legal Domain			
System	**Authors**	**Features**	**Techniques**

RAVN ACE	RAVN Systems (2016)	• Organises, discovers, and summarises important information;
		• Uses combination of information processing, NLP, and semantic processing;
		• Processes documents and unstructured data; and
		• Has three applications, namely:
		  ○ RAVN Extract;
		  ○ RAVN Connect Enterprise; and
		  ○ RAVN Refine.
			• NLP; and
			• IR.

Equivo	Equivo (2012)	• Performs text analytics;
		• Focuses on e-discovery within legal domain;
		• Uses six types of applications for analysis, namely:
		  ○ Near-duplicates;
		  ○ Email threads;
		  ○ Language detection;
		  ○ Search;
		  ○ Themes; and
		  ○ Relevance.
			• NLP;
			• IE;
			• IR; and
			• Clustering.

Beagle	Beagle Inc. (2018)	• AI platform;
		• Using tagging and a binary classifier; and
		• Offers four solutions:
		  ○ Contract Analysis;
		  ○ Licence Analysis;
		  ○ Corporate Compliance; and
		  ○ Regulations and Law.
			• NLP

eBravia	eBravia (2018)	• Performs contract analysis; and
		• Offers four solutions:
		  ○ Contract Analyser;
		  ○ Diligence Accelerator;
		  ○ Lease Abstractor; and
		  ○ Bespoke.
			• IE

Table 3-14: Summary of Systems within the Legal Domain

3.10.6 Summary of Frameworks and Systems in the Legal Domain

Figure 3-17 illustrates the common processes shared by the frameworks and extant systems (Section 3.10.1 to Section 3.10.5). Both frameworks and extant systems used four common processes coupled with certain techniques. The processes followed are:

1. Collect data from a source;
2. Pre-Process analysis;
3. Process data; and

4. Store information in repository

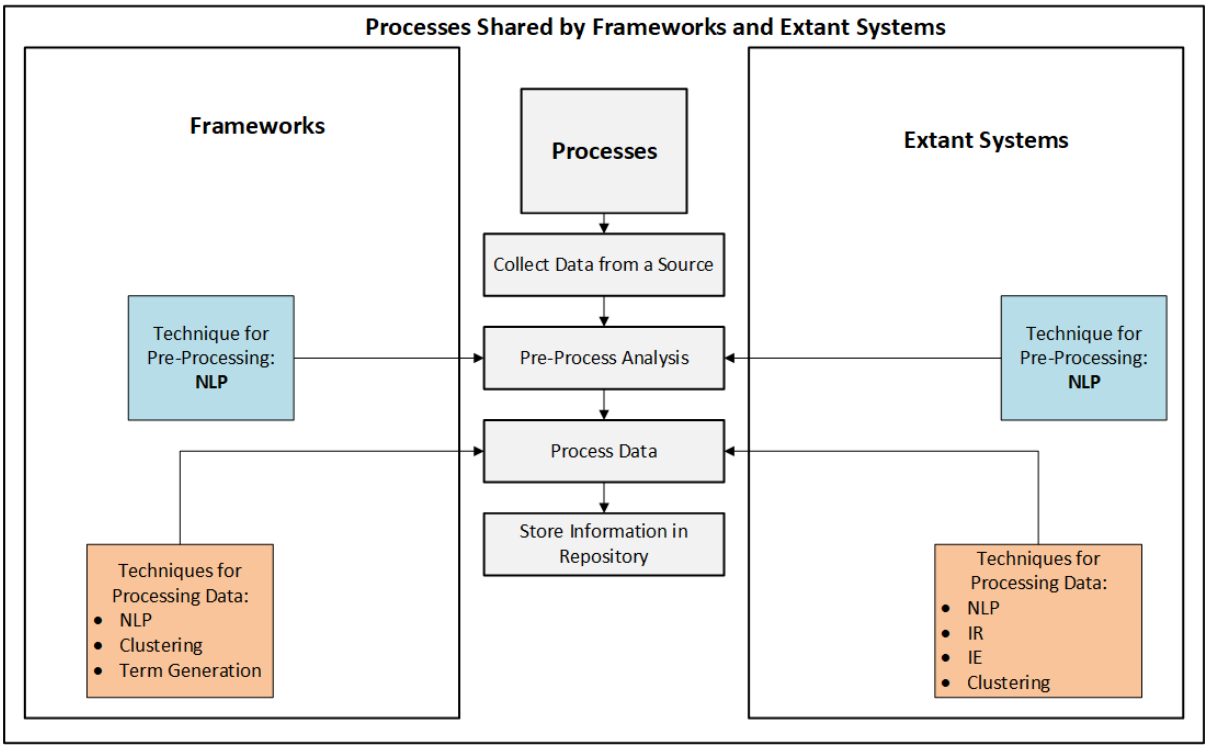


Figure 3-17: Processes and Techniques Shared by Frameworks and Extant Systems

The second process, pre-process analysis, makes use of NLP. In the third process, process data, different techniques were used such as NLP, IR, IE, clustering, and term generation. Based on Figure 3-17, a model can be created that incorporates the shared processes and some of the techniques used by the frameworks and extant systems. The model would need to collect data from a source, perform pre-processing analysis, process the data using IE techniques, and save the data.

3.11 IE Model

Based on the concepts investigated in the literature (Section 3.2 to Section 3.9) and the frameworks and methodologies (Section 3.10) a proposed IE model was derived and is depicted in Figure 3-18. The IR Process in Figure 3-18 is similar to the Collect Data from a Source process identified in Figure 3-17. The IR Process as modelled in Figure 3-3 receives a query and returns data from a source to be further processed. The IE Process and Information Storage Process in Figure 3-18 are the same as Pre-Process Analysis, Process Data, and Store Information in a Repository from Figure 3-17 as the data would need to be processed and stored for later use.

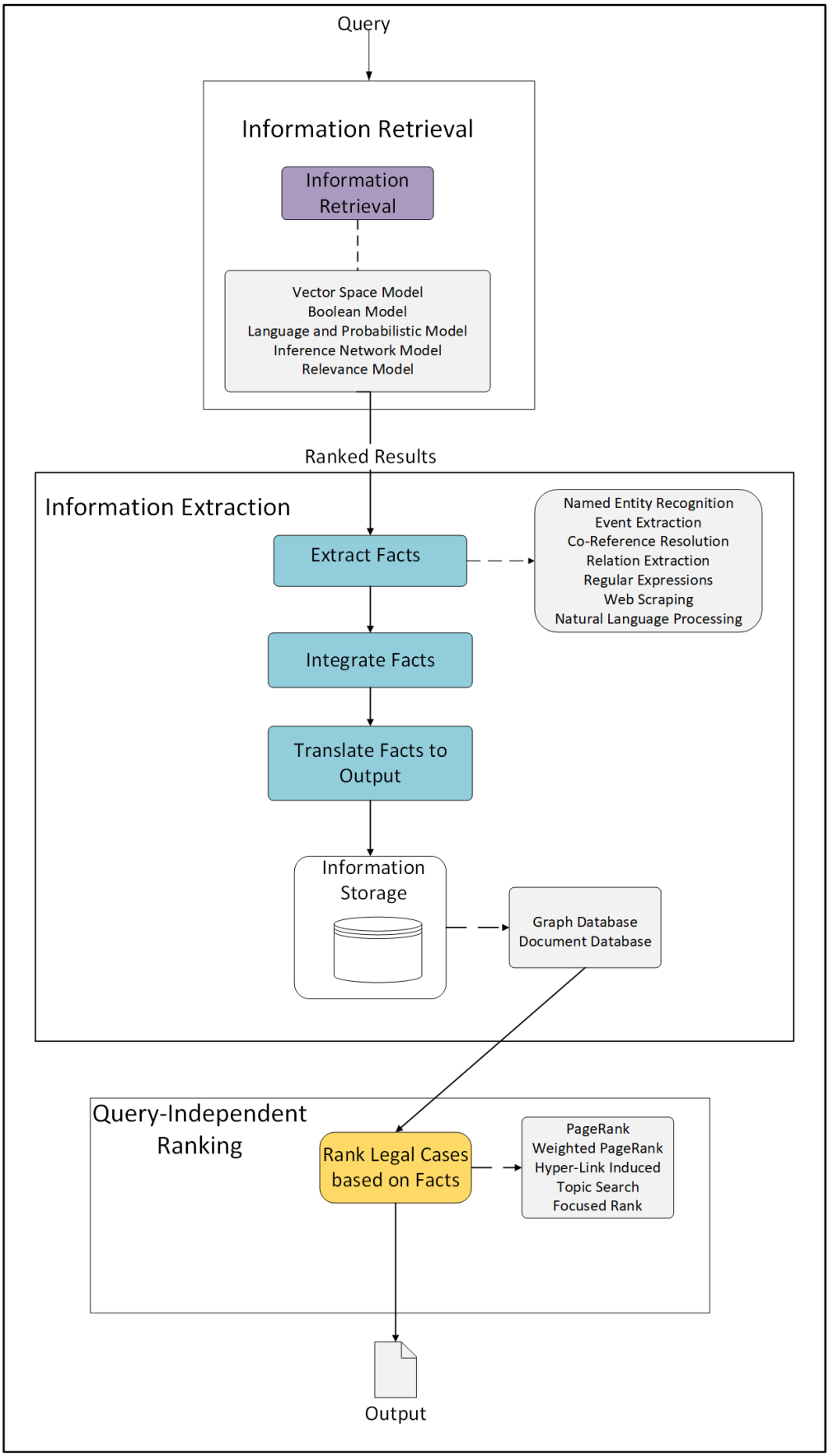


Figure 3-18: IE Model

The IR process in Figure 3-18 accepts a query from a user and processes the query via four processes to display output based on the information processed. The four processes deal with IR, IE, Information Storage, and Query Independent Ranking. The IR process will process a user's query to return a set of ranked results. The first round of ranking is influenced by the type of IR model implemented. As discussed in Section 3.3 there are four IR models that can be used to implement the IR process. The Boolean model should be avoided as its binary nature can produce inaccurate results. The Vector Space Model could be considered due to its popular usage and the way it ranks documents.

The IE process will extract, integrate, and translate facts from documents. Section 3.4 to Section 3.7 investigated multiple options that are available to extract facts from documents. Depending on the structure of the documents, general IE techniques can be used to extract facts. Alternatively, Natural Language Processing or Regular Expressions can be used to extract facts. After all required facts have been processed, the facts must be saved for future reference and to avoid duplicate processing.

Section 3.8 investigated graph and document databases as an option for storing information instead of relational databases. Using a graph database would allow for the extracted facts to be stored in nodes that are connected to each other by specific relationships. Using a document database would allow for the extracted facts to be stored as documents and have multiple types of data embedded into a single document.

To further process and get valuable information from the information that has been extracted, query-independent ranking algorithms can be applied. Different algorithms can be used. Section 3.9 investigated algorithms such as the PageRank, Weighted PageRank, HITS, and Focused Rank algorithms. The PageRank makes use of links to rank pages. Similarly, legal cases can be connected to other legal cases if there is a referral. This referral can act as a link that can be parsed through the PageRank algorithm. The Weighted PageRank would not be suitable in recommending the MAC as it only considers the link structure and not the content of the legal case. This would mean that the "division" attribute of a legal case would be overlooked. The division refers to which court a case is being appealed. The "division" attribute can play an important role as divisions contribute to the importance of a legal case. The HITS ranking algorithm uses too many constraints which could prevent the MAC from being produced. The Focused Rank algorithm has limited information available for it, which could result in limited support for future development.

In terms of IE, the model would have to make use of one of the techniques discussed in throughout this chapter (Table 3-3). The model would also have to cater for the problems identified in literature for processing text (Table 3-15). It is likely that these problems could occur as legal documents could contain ambiguous words, inconsistencies, or words that are either unnecessary or important. Table 3-15 summarises the problems that can be encountered when processing text.

Problem	Section	Author/Source
Ambiguity of words	3.4	Sumathy and Chidambaram (2013)
Inconsistencies because of special formats, acronyms, and abbreviations	3.4	Gurusamy and Kannan (2014)
Unnecessary and confusing words	3.2	Gurusamy and Kannan (2014)
Identifying meaningful keywords	3.2	Gurusamy and Kannan (2014)

Table 3-15: Problems Encountered when Processing Text

3.12 Conclusions

This chapter reported on the first activity of the DSR Methodology, namely Problem Identification. Multiple challenges in processing text can occur and should be addressed accordingly. The absence of pre-processing tasks on text can bring about poor results returned to a user. Additionally, ambiguity,

inconsistencies, and special formatting can all negatively affect text processing. Different methods for processing text were investigated and various models and algorithms were identified from literature for IR. Analysis of IR resulted in the illustration of a generalised IR process with accompanying processes and techniques (Figure 3-3). The process of IE can be used to aid IR by extracting facts from data that IR has processed. Techniques such as general IE, NLP, web scraping, and regular expressions were identified to perform IE. The facts extracted after IE should be stored for future use. Instead of storing extracted facts in traditional relational database management systems, graph and document databases were investigated as alternative options. Literature suggested that graph and document databases brought better performance, flexibility, and scalability compared to relational database management systems. The frameworks and systems investigated in the legal domain all performed variations of IE and NLP. The frameworks and methodologies were aimed at grouping cases and improving research by returning appropriate information to users. None of the studies performed ranking on the data. The frameworks and systems investigated performed IE and NLP on various legal documents, not only legal cases. Further inspection of the concepts investigated in the literature review revealed that different techniques from each concept can be incorporated to create a prescriptive model to recommend the MAC.

This chapter fulfilled the following research objectives:

- **RO1**: *Identify the problems experienced when processing text as identified by literature and within a real-world context.*

This chapter partially fulfilled the following research objectives:

- **RO3:** *Determine what techniques and algorithms can be used to recommend the MAC.*

By addressing these ROs, two deliverables were obtained namely, an expanded list of problems (Table 3-15), and techniques and algorithms to process text (Figure 3-3, Table 3-3, and Table 3-12). The expanded list of problems consisted of problems encountered when processing text as identified by sources in literature. Table 3-16 provides a summary of the different IE techniques that were investigated in this chapter. These deliverables will contribute to expanding on the objectives for a solution.

IE Technique	NER	CO	RE	EE	Web Scraping	NLP	Regular Expressions
What it does	Identifies expressions related to an entity	Identifies multiple mentions of a particular entity	Detects and classifies predefined relationships between entities in a body of text	Identifies events and creates detailed structured set of information	Extracts information from a website	Analyses natural language text	Uses patterns to process specific bits of text
When to use it	To extract descriptive information about an entity and complete a template	To identify all mentions of an entity in a body of text	To identify relationships between different entities	To have a detailed set of information about events	To automatically extract data from a website	To identify meaning from bodies of text containing natural language	To extract text, validate input text, search for text, or replace text

Table 3-16: Summary of Different IE Methods

The next chapter will review findings from questionnaires to identify problems experienced in the real-world context of the legal domain and will present the design process and proposed model for the legal domain.

Chapter 4: The Real-World Context of the Legal Domain

4.1 Introduction

The previous chapter reported on an extensive review of literature related to concepts for IR, IE, frameworks, methodologies, and tools that can be used to create a model within the legal domain. This chapter reports on the second and third activities of the DSR methodology namely, Definition of Objectives for a Solution and Design and Development (Figure 4-1). The problem within the legal domain is identified, potential objectives and solutions will be determined, and two artefacts to address the problem are designed in this chapter. Different processes were followed to create the proposed artefacts (Section 4.2). Legal citations and the structures and features of a legal case are investigated (Section 4.3). Problems faced within the legal domain will be explicated via questionnaires (Section 4.4). The environment in which the case study falls will be reviewed (Section 4.5) along with the current architecture used by the case study (Section 4.6). Problems that could be encountered when processing legal cases will be reviewed (Section 4.7). Objectives and requirements for the proposed solution are identified (Section 4.8). The following research objectives are investigated in this chapter:

- **RO1**: *Identify the problems experienced when processing text as identified by literature and within a real-world context.*
- **RO2**: *Identify the attributes of a court case that can be used to aid in recommending the MAC.*
- **RO3:** *Determine what techniques and algorithms can be used to recommend the MAC.*

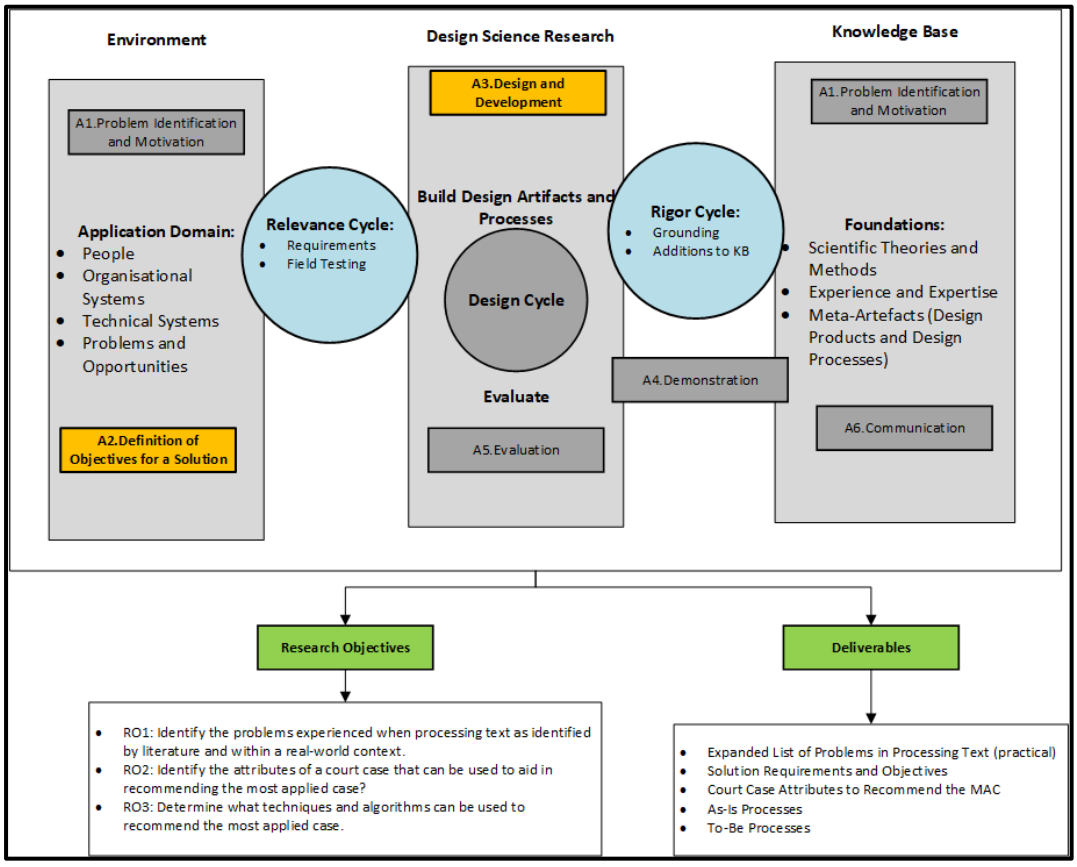


Figure 4-1: Chapter 4 DSR Activities

4.2 The Artefact Design Process and Research Methods

The literature review allowed the researcher to gain an understanding of the different techniques that can be used to create a prescriptive model for recommending the MAC and to identify problems reported by researchers. To identify the problems experienced by legal researchers, the researcher sent out a series of questionnaires to the experts from LexisNexis. As part of the DSR methodology, an artefact is required to solve a practical problem. The artefact suggested for this research problem is a prescriptive artefact (Section 2.3.3). The prescriptive artefact is a model that would recommend the MAC for a field of law. Two methods were performed during the relevance cycle of DSR, namely sending out questionnaires, and analysing existing systems. A literature review was conducted during the rigor cycle. Table 4-1 summarises the methods used during the research. The information obtained through the relevance and rigor cycles are passed through to the design cycle.

Methods Used in Research			
Method	**Reason**	**How Method Conducted**	**Cycle Addressed**
Questionnaires and email correspondence	To further explicate the problem and obtain a participant profile	Via email	Relevance Cycle
Literature Review	To get an understanding of the techniques, models, and tools available within the problem domain	Analysing and critically reviewing textbooks, academic articles, and articles posted online	Rigor Cycle
Extant Systems Analysis	To analyse existing systems within the problem domain	Comparing the systems	Relevance Cycle

Table 4-1: Summary of Methods used in Research

4.3 Legal Cases in South Africa

Principles followed to write legal citations as well as the structure and features of a typical South African legal case will be presented in this section. The principles indicate how the legal citations should be written in different areas of a legal case while the structure of a legal case must be considered to extract information.

4.3.1 Legal Citation Principles

The writing of legal citations is governed by citation principles. Citation principles are followed by a legal practitioner to write legal citations. The different types of citation principles are illustrated in Figure 4-2. There are four categories of citation principles, namely:

- Full address principles;
- Other minimum content principles;
- Compacting principles; and
- Format principles.

Full address principles refer to the completeness of the address or identification of a cited document in terms that allow a user to retrieve a document easily. **Other minimum compacting principles** refer to the inclusion of additional information apart from a retrieval address. Additional details include author names and the year a decision is made. **Compacting principles** reduces the space taken by additional information by means of abbreviations and use of principles that eliminate redundancy. **Format principles** refers to punctuation, typography, and the order of items within a citation. These four principles are used throughout a legal case when referencing other legal cases for precedent.

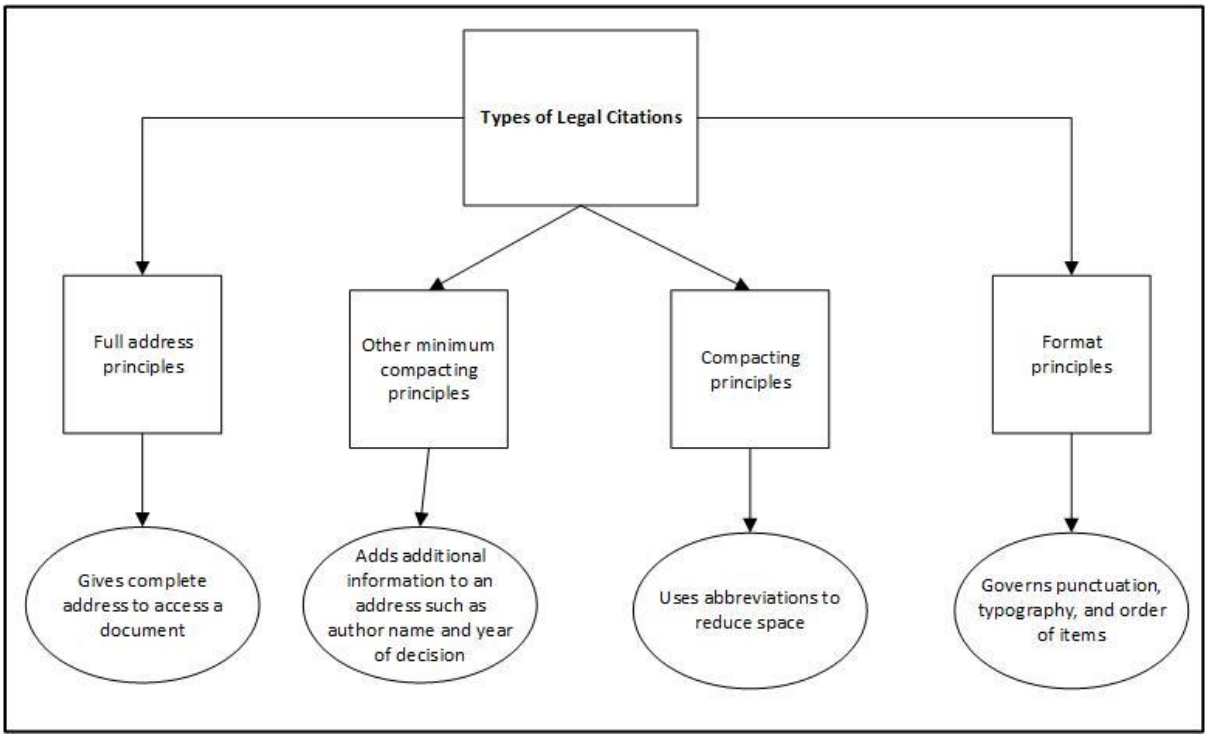


Figure 4-2: Types of Legal Citations (Martin,2013)

4.3.2 Structure of a Legal Case in South Africa

A well-structured legal case report aids in improving clarity, conciseness, and helps ensure that the reasoning process for a case is complete (Dessau & Wodak, 2003). A legal case report follows a semi-structured format as there is no set layout that a case must adhere to. The research analysed multiple legal cases by reading and comparing them to each other. During the analysis, common sections of information were identified. An example of parts of a legal case obtained from LexisNexis is provided in Appendix D. A typical legal case consists of sections that provide the following:

- General data;
- An editor's summary;
- Notes;
- Cases Referred To (CRT) in the judgement; and
- The final judgement of the case.

The **general data** section of a legal case provides basic information related to a case such as (Figure 4-3):

- Division;
- Date;
- Case number;
- Before (the judges who sat on the case);
- Sourced by; and
- Who summarised the case.

Azisa (Pty) Ltd v Azisa Media CC and another
[2002] 2 All SA 488 (C)

Division:	Cape of Good Hope Provincial Division
Date:	27 November 2001
Case No:	6215/00
Before:	Nel J
Sourced by:	C.Webster and AD.Maher
Summarised by:	S.Pillay
Parallel Citation:	2002 (4) SA 377 (C)

Figure 4-3: General Data About a Case

The **division** refers to which court a case is being appealed in. There are five courts within South Africa that are organised in a hierarchy of supremacy, illustrated in Figure 4-4. The five courts based on the hierarchy of supremacy are the Constitutional Court, Supreme Court of Appeal, High Courts, Special Courts, and Tribunals, Councils and Commissions.

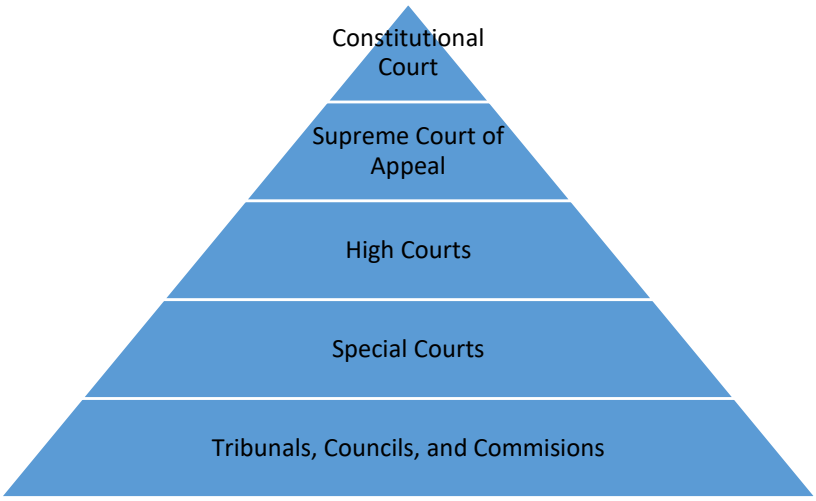


Figure 4-4: Hierarchy of Courts in South Africa (Author's own work)

The **Constitutional Court** deals with all cases that are constitutional related and is the highest-ranking court in South Africa. A decision made in the Constitutional Court is binding and must be followed by all courts. The **Supreme Court of Appeal** deals with appeals made from the High Courts. The Supreme Court of Appeal is the highest-ranking court to which criminal and civil cases can be heard. All decisions made in the Supreme Court of Appeal are binding and must be followed in all lower courts unless the Constitutional Court over-rules a Supreme Court's decision. **High Courts** hear all criminal and civil cases as well as some constitutional cases (Barrett-Grant & Heywood, 2003). **Special Courts** only hear specific cases (Webbers Attoryneys, Notaries, 2017). Special Courts consist of various other courts such as the Labour Appeal Court and Small Claims Court. **Tribunals** are court of justices used to settle specific types of disputes (Yambu, 2018). **Councils** are an advisory body who assist authoritative

figures with legal matters while **commissions** grant authority from government to individuals to perform certain acts, exercise jurisdiction or perform the duties of an office (Black et al., 1990).

Date and **case number** refer to when the case was held as well as a unique number to identify each case. All legal cases are heard by an odd number of **judges** to ensure a unanimous decision is always reached. **Sourced by** indicates who sourced the information and **Summarised by** indicates who summarised the legal judgement.

The **Editor's Summary** provides a shortened description of an entire case and is aimed at people who do not have enough time to read through an entire case. **Notes** refer to areas and procedures of law that a user can refer to. In the legal case, a list of **CRTs** are provided. These CRTs are cases that were used to support any arguments (Figure 4-5 and Figure 4-6). The **final judgement** of the case provides the ultimate decision made by the judges.

Page 490 of [2002] 2 All SA 488 (C)

Holmes v North Western Motors (Upington) (Pty) Ltd 1968 (4) SA 198 (C)

Joubert v Enslin 1910 AD 6

Nair v Naicker 1942 NPD 3

National Bank of South Africa Ltd v Leon Levson Studios Ltd 1913 AD 213

Nell v Mulbarton Gardens 1976 (1) SA 294 (W)

Peregrine Group (Pty) Limited v Peregrine Holdings Limited 2000 (1) SA 187 (W)

Pivot Point v Registrar of Companies 1980 (4) SA 74 (T)

Randburg Town Council v Kerksay Investments (Pty) Ltd [1997] 4 All SA 121 (1998 (1) SA 98) (A)

R v Venter 1907 TS 1910 - **F**

Versveld v SA Railways and Harbours 1937 CPD 55

Figure 4-5: First Example of CRTs

Cases referred to in judgment

Governing Body of the Juma Musjid Primary School and others v Essay NO and others (Centre for Child Law and another as amici curiae) 2011 (8) BCLR 761 ([2011] ZACC 13) (CC) - **Approved**	380
Government of the Republic of South Africa and others v Grootboom and others 2000 (11) BCLR 1169 ([2000] ZACC 19; 2001 (1) SA 46) (CC) - **Referred to**	385
Harksen v Lane NO and others 1997 (11) BCLR 1489 ([1997] ZACC 12; 1998 (1) SA 300) (CC) - **Referred to**	386

Page 371 of [2016] 1 All SA 369 (SCA)

Head of Department, Mpumalanga Department of Education and another v Hoërskool Ermelo and another 2010 (3) BCLR 177 ([2009] ZACC 32; 2010 (2) SA 415) (CC) - **Referred to**	383
Pretoria City Council v Walker 1998 (3) BCLR 257 ([1998] ZACC 1; 1998 (2) SA 363) (CC) - **Referred to**	386
Section 27 and others v Minister of Education and another [2012] 3 All SA 579 (2013 (2) SA 40) (GNP) - **Referred to**	375

Figure 4-6: Second Example of CRTs

To recommend the MAC, various sections of information from a legal case needs to be extracted. The extracted information can be used to build up a new information storage structure and be parsed through a query-independent ranking algorithm (Section 3.9). Information from a legal case that can be useful in recommending the MAC are a legal case's general data and the list of all CRTs. The general data that can be useful are the case's title, division, date, and case number. These bits of general data can be used to create a smaller version of the legal case along with the CRTs. The case number can be used to identify each legal case. The title and date can provide more information about a legal case while the division can be used to determine the legal case's supremacy (Figure 4-4). The hierarchy of divisions can influence the value a legal case has. For example, assume there are three legal cases, where the first legal case is an active case in the High Court, and the other two cases were heard in a Special Court and the Supreme Court of Appeal. In terms of supremacy, the decision made on the case in the Supreme Court of Appeal overrules the decision made on the case in the High Court. This implies that if the nature of the active case is identical or similar to the case heard in the Supreme Court of Appeal then the judge sitting on the active case has to make the same decision as that in the Supreme Court of Appeal. This means that the value of the case heard in the Supreme Court of Appeal is higher than the value of the case heard in the Special Court.

All the CRTs of a legal case should be extracted to aid in the ranking process. When extracting the details of a CRT it will be useful to extract the CRT's title, year, journal, and the action taken on that case. The CRT's title and year can help identify the CRT in the journals to which it belongs. The action taken on the CRT will be important as the action determines how valuable the CRT will be for recommending the MAC. Actions such as 'Referred to' or 'Applied' will hold a higher value than an action like 'Distinguished' as it means that CRT can be possibly be used to build up a defence for a legal researcher. Table 4-2 summarises the court case attributes that can be used to recommend the MAC.

Court Case Attributes		
Section of Legal Case	**Attribute**	**Reason**
General Data	Title	Build a summarised version of the legal case
	Division	Determine a legal case's value
	Date	Build a summarised version of the legal case
	Case Number	Build a summarised version of the legal case To identify each case
CRTs	Title	Build a CRT object
	Year	Build a CRT object
	Journal	Build a CRT object
	Action	Determine the value of the CRT

Table 4-2: Court Case Attributes that can be used to Recommend the MAC

4.4 Problems Faced at LexisNexis

The results of the questionnaires and a description of the participant profiles will be reported on in this section. The aim of the questionnaires was to explicate and more clearly understand the problems within the legal domain at LexisNexis.

4.4.1 Aim of Questionnaires and Participant Profiles

Questionnaires are used to easily generate data from any number of respondents at a low cost (Johannesson & Perjons, 2012). Questionnaires were completed by five experts from LexisNexis. Due to the level of detail and complexity relating to existing processes and technical infrastructure it was deemed more appropriate to use questionnaires than interviews. In the context of this research, experts were defined as individuals who were highly knowledgeable in the legal domain's processes and systems. The experts from LexisNexis worked in different departments related to IT and editing

of legal cases. The experts provided valuable information with regards to practical problems within the legal domain and processes followed at LexisNexis. The results obtained from the questionnaires was used to create a set of requirements that the prescriptive artefact must address. Table 4-3 and provides a summary of the profile of the five experts (E1 to E5) from LexisNexis that completed the questionnaires.

Position Held	Expert	Work Experience	Qualification	Responsibilities at LexisNexis
Technical Development Manager and Solutions Architect	E1	Software Development Manager for six years and a Senior Developer for two years	BSc Electrical and Electronics Engineering	Manages technical research projects. Architect for various software and processes
Senior Editor	E2	Previously a practicing Attorney. Senior Editor since 2008	B.A (Psychology and Criminology) (UDW); LLB (UKZN)	Co-ordinates and runs Judgments Online product. (Now moved into Publishing Co-ordinator Position in New Business and Content Development Team)
Managing Editor	E3	Practised as an Attorney prior to joining LexisNexis. Currently the Managing Editor for Law Reports	Law (LLB) (UKZN)	Manages online law reports and content
LegalCitator Editor	E4	BLC Editor at LexisNexis for five years Advocate at KZN Society of Advocates for three years, Lecturer at UKZN for three years	LLB, LLM	Updating and maintaining the LegalCitator content
Editor and Developer of LegalCitator	E5	24 Years	B.Proc, Attorney	Editor at LexisNexis

Table 4-3: Profile of Experts

4.4.2 Findings from Questionnaires

Questionnaires were sent to the experts at LexisNexis, followed by a visit to their offices in Durban. The visit formed part of the relevance and rigor cycles in DSR and aided in completing the second activity of the DSR methodology. The visit to the head offices allowed the researcher to address various aspects of the relevance cycle's application domain such as people, technical systems, and problems and opportunities. The feedback from the experts verified and clarified the research problem. This section discusses the findings from the questionnaires and emails. The findings from the first questionnaire confirmed the problem and objectives stated in Chapter 1 and provided more detail.

4.4.2.1 Findings from First Questionnaire

The Problem

Two high level problems were identified namely, a primary and secondary problem. The primary problem was that of recommending the MAC for a field of law to a legal researcher. The experts all

stated that too much time is spent on finding the MAC. The secondary problem was that LexisNexis has no formal systems or processes in place to help legal researchers find the MAC.

Existing Systems

LexisNexis uses a specialised in-house product called **LegalCitator** that allows users to perform basic searching and provides an analysis of judgements. The search functionality is provided by the technology called ElasticSearch. LegalCitator does not use any algorithms for mining or extracting data but LexisNexis is trying to get LegalCitator to perform entity extractions.

When asked why the software such as IBMs Watson or ROSS (Section 3.10) are not used at LexisNexis it was explained to the researcher that licencing issues and competitors already using the software prevented LexisNexis from implementing any of the software.

Data and Processes

The data available is Case Law and Legislative data. All data is available in XML format. The data obtained from LexisNexis is semi-structured whilst the data stored on LexisNexis' databases are structured. The data is collected and inserted into the system by various editors who are responsible for different law reports/publications. Data for the LegalCitator is entered into the LegalCitator desktop application whilst data for other law reports are entered into a stylised Microsoft Word document and later converted to XML. The rate at which data for reports/publications is updated depends on the publication but data for LegalCitator is updated monthly.

4.4.2.2 Findings from Second Questionnaire

To further understand the research problem, a second questionnaire was sent to experts at LexisNexis. The findings from this questionnaire verified and validated the problems and requirements.

Data, IR, and Text Mining

No formal IR processes are followed at LexisNexis. However, LexisNexis makes use of visual pattern identification and document meta-data markup. All data is referenced on a SQL database whilst the content is stored in semi-structured XML format and replicated to the production environment. Converting of data to XML is done using a tool called Link Management Tool. LexisNexis contains 100 000 law reports all from various time periods. No text mining, therefore, no IE, is performed by the LegalCitator system. LexisNexis believes that text mining could allow for clients to get information faster.

General Information

According to LexisNexis, the primary objectives of the artefact are:

- To process and extract ALL SA legal cases;
- To save extracted legal cases for future use; and
- To help in recommending the MAC.

4.4.2.3 Additional Information Obtained (from emails and site visit)

Processes for Adding a Case to the LegalCitator Database

In terms of processing cases, the researcher aimed to find out what processes are currently followed to transform a case from its raw state as a judgement to entering all the case's data into LegalCitator. It was found that cases were manually processed by employees with legal backgrounds.

Categorising a legal case is done manually by a person known as an Editor. An Editor has a legal background but only focuses on the editing of legal documents that are uploaded to systems online. The Editor reads through the legal case and then uses his/her legal knowledge to determine which category of law the case fits into. A specific process is followed to enter a case into the database and LegalCitator. The process includes adding the content into a stylised template, proof reading, and adding additional information.

The following tasks are performed by Editors and are illustrated in the As-Is business process (Figure 4-7):

1. Retrieve case from Q Drive that contains all unprocessed judgements;
2. Apply corporate styling to the case;
3. Proof read the case;
4. Add relevant information obtained, from Gracies Database, such as area of law and names of judges;
5. Add keywords and summaries;
6. Add parallel citations;
7. Send final case to the Electronic Product Team;
8. Build case to the live site;
9. Store case on BLC Database; and
10. Legal practitioners use the final case from task 7 to enter details into the BLC Database.

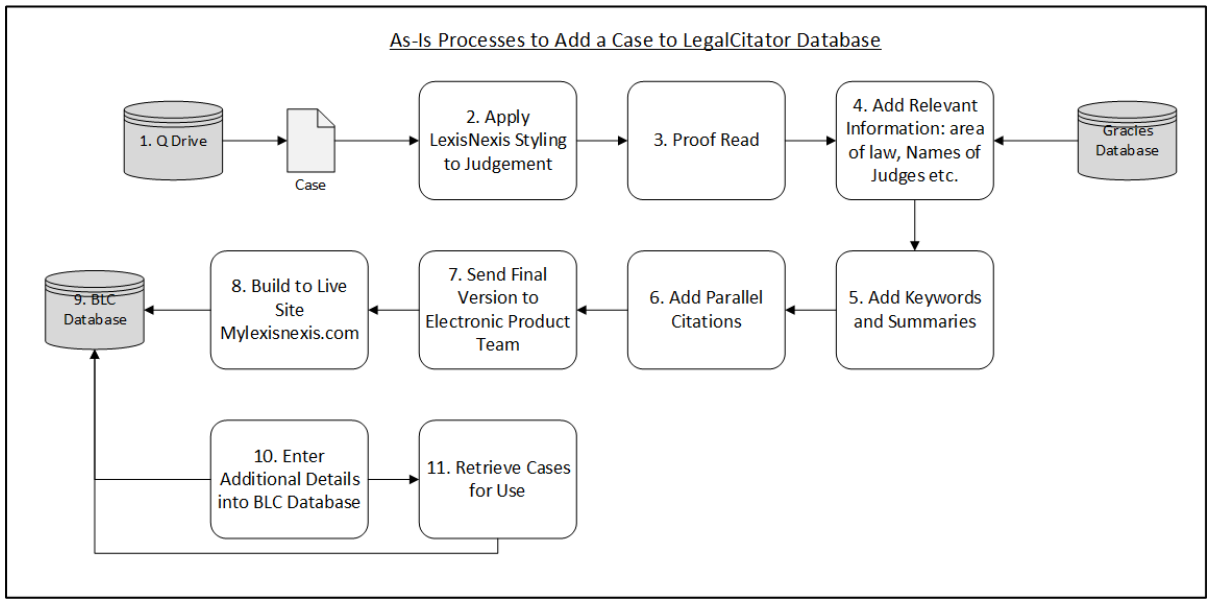


Figure 4-7: As-Is process at LexisNexis

Additional details in task 10 can be the category of a case. Categorising a case requires identification of various aspects such as the type of court the case was heard in and the legislation used within the case. Keeping track of a word-count of specific words can be used to categorise a case but it does not always reflect a true image of what category a case belongs to. During task 11 LexisNexis can use the cases for any purpose such as finding the MAC.

Databases

Two databases are used with regards to processed cases, namely the BLC Database and the Gracies Database. Final versions of all cases are stored on the BLC Database whilst the Gracies Database contains all subject-indexed data. Gracies Database is used to refer to permanent headings such as

areas or categories of law and breaking down keywords from a judgement. These keywords are then added to the case that is processed. The BLC Database is used by employees to add in a case's details such as the category of law and judges' names. The schema, obtained from LexisNexis, for the BLC Database can be seen in Appendix E With regards to finding the MAC, the details of an informal process conducted by LexisNexis was provided which is investigated in Section 4.5.

Legal Citations and Finding the MAC

With regards to legal citations, it was found that the style used to reference citations in South Africa was different to styles used in other countries, particularly the United States of America. The contents of a legal citation also vary depending on the structure of the publication cited and the frequency of the publication. Parallel citations refer to one case that comes from different publications or sources.

4.5 Systems at LexisNexis

Finding the MAC for a field of law can be a long and tedious process. Legal researchers must read through countless cases to find similar facts and applicable cases worth using. Based on findings from questionnaires in Section 4.4.2.2, it was determined that LexisNexis has no formal processes and tools to aid in finding the MAC for a field of law. Furthermore, no range of time taken to find a case could be provided as the time is different for every situation. An informal process was explained on how researchers find the MAC for a field of law. The informal manual process consists of the following tasks which are performed manually by legal researchers:

1. Determine the research area;
2. Determine the problem that needs to be solved e.g. "Unfair dismissal";
3. Search for cases in the research area;
4. Extract ±20 cases returned from the search process;
5. Read through the 20 cases to find similar facts to current case;
6. Look for cases that use "Applied to"; and
7. Repeat process until applicable cases are found.

The steps from the informal process help identify which attributes of a legal case can be used to recommend the MAC. Based on task 2, the division in which the legal case was heard will have to be extracted. A summarised version of the case can be created by extracting the date, case number, and all cases referred to. Based on task 6, specific words regarding referred to cases will have to be extracted. These words can include "Applied to" as indicated in task 6. It is suspected that problems will arise when processing the text of a legal case. LexisNexis employees make use of two systems namely, the Mylexisnexis.co.za website and the LegalCitator. Mylexisnexis.co.za can be used when performing tasks such as finding the MAC or any other research while LegalCitator is used to process and publish legal cases to the Mylexisnexis.co.za website.

An extant systems analysis was conducted to determine how information is accessed and analysed. Marr (2016) states that all new case data is entered and stored on databases. Mylexisnexis.co.za is an online search system developed in-house that analyse judgements. The system is powered by the Elasticsearch search engine. ElasticSearch is an open-source search engine that uses a full-text search engine library called Apache Lucene (Gormley & Tong, 2015). Elasticsearch stores an entire object or document and indexes the content of each document to make it searchable. With this search engine, a user can perform functions such as filtering, searching, and indexing on documents instead of rows of columnar data. Figure 4-8 illustrates a summary of the Mylexisnexis.co.za system.

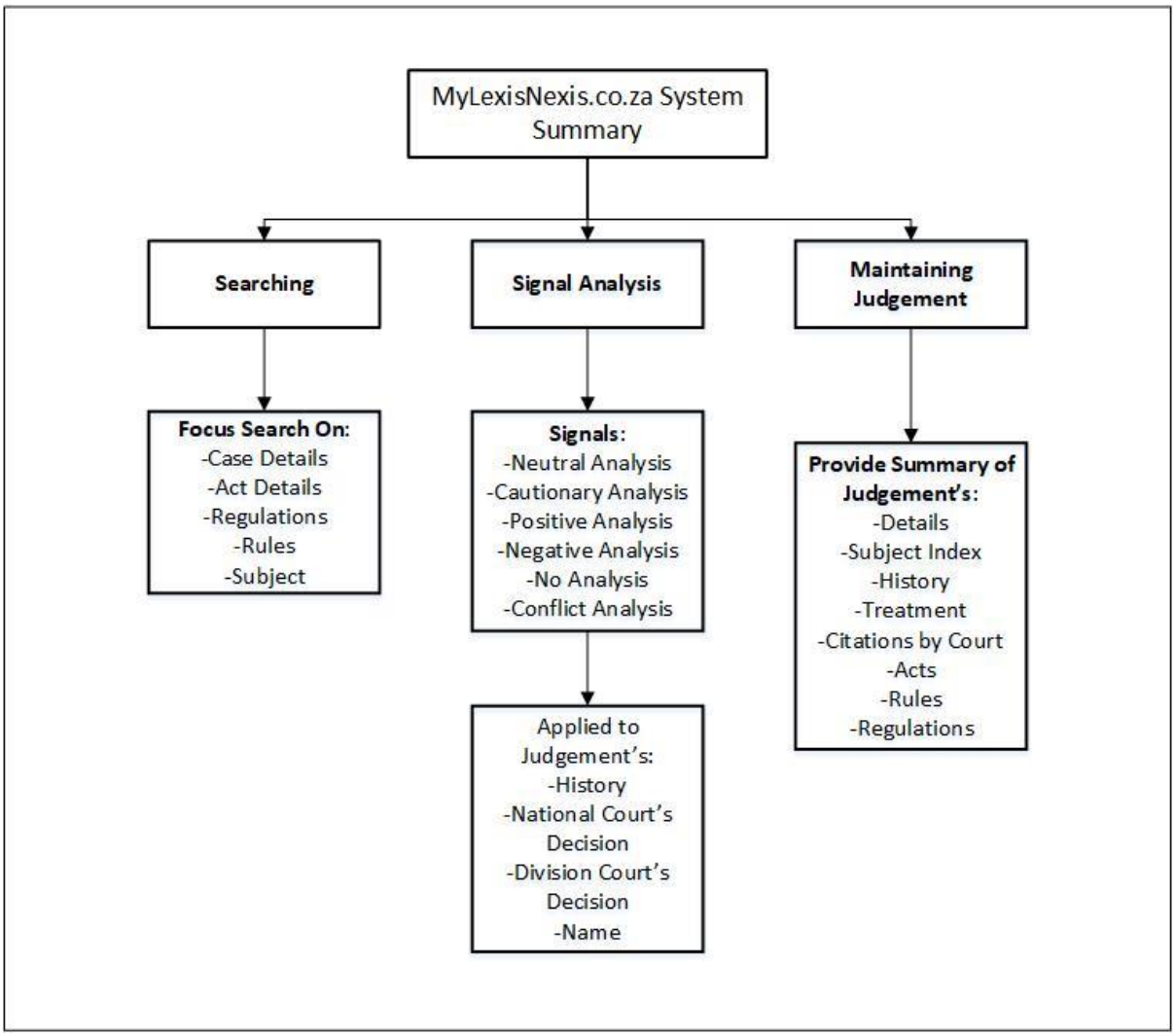


Figure 4-8: MyLexisNexis.co.za System Summary (Author's own work)

The Mylexisnexis.co.za system performs the following functions (LexisNexis, 2016) :

- Searching;
- Signal analysis; and
- Maintaining judgements;

When searching for information on Mylexisnexis.co.za, users currently have the option to focus their search on five categories of attributes, namely:

- Case details;
- Act details;
- Regulations;
- Rules; and
- Subject.

Searching on **case details** allows the user to enter an array of information to refine their search. Users can enter details regarding a judgement's name, citation, subject, dates, or a judge's name. Searching on **act details** allows the user to obtain information regarding specific acts by letting the user enter information such as act name, number, and year. Searching on **regulations** allows a user to enter information such as a regulation's name, number, and the type of regulation. If searching on **rules**,

the user must select a country, jurisdiction, and division. A similar layout is provided when searching on **subject** with the addition of a division field to further refine the search. Searching based on **subject** was the most commonly used option for searching.

Signal analysis is used to provide an immediate appraisal of each judgement to aid users in determining which cases they might be interested in. Signals that a judgment can receive are:

- **Neutral analysis** – a judgement has not been analysed by a court of law to affect its value in terms of precedence or interpretation;
- **Cautionary analysis** – a judgement has been analysed by a court of law in a way that suggests it should be re-examined;
- **Positive analysis** – a judgement has been analysed by a court of law in a way that suggests it can be relied upon as authority;
- **Negative analysis** – a judgement has been analysed by a court of law in a way that suggests it should not be relied upon as authority;
- **No analysis** – the judgement has not been analysed by a court of law at all; and
- **Conflict analysis** – a judgement has been analysed by at least two National Courts in a conflicting way resulting in the analysis being unresolved by the LegalCitator.

Signals are given to four features of a case, namely:

- Judgement history;
- National courts' decision on the judgement;
- Division courts' decision on the judgment; and
- Judgement name.

Once a user has selected their required results, LegalCitator provides a summary of each judgement. The summary contains details regarding the following:

- Judgement details;
- Subject index;
- Judgement history;
- Judgement treatment;
- Judgements cited by court;
- Acts, ordinances and by-law;
- Rules; and
- Regulations.

Before content can be published to the Mylexisnexis.co.za system, the content has to be processed on LegalCitator by LexisNexis users.

LegalCitator is an in-house system developed by LexisNexis to maintain and extract judgements from four series of law reports. LegalCitator requires the user to perform many tasks and once completed, allows the user to publish updates to the live site, Mylexisnexis.co.za. The following series of law reports are supported by LegalCitator:

- ALL SA;
- Butterworth's Constitutional Law Reports;
- Butterworths' Labour Law Reports; and
- South African Law Reports.

The main features that LegalCitator allows the user to perform are the following:

- View tables;
- View legislation;
- Maintain cases:
    - View existing cases;
    - Create new record of a case;

Sixteen **tables** are available for users to view. A screenshot of the list of tables available can be seen in Appendix G: Screenshots from LegalCitator. In terms of **legislation**, LegalCitator contains all Acts and Ordinances for law in South Africa that users can view. When **viewing a case**, the user can enter four different types of information to retrieve the required cases, namely:

- Citation;
- Case name;
- Case number; and
- Judgement date.

When **creating a new record**, all case details must be added to LegalCitator. Fourteen fields must be completed when creating a new record. Six of the fourteen fields are compulsory and must be completed to proceed with creating the case. A screenshot of the fields that must be completed when creating a new case can be seen in Appendix G: Screenshots from LegalCitator. The compulsory fields are the following:

- Division;
- Case number;
- Citation;
- Judgement date; and
- Originator.

The user has the option to edit existing records or newly created records by adding additional information. Additional information is added under eleven tabs, namely:

- Judgement details;
- Case history;
- Parties and appearances;
- Judges;
- Citations;
- Subjects;
- Words and phrases;
- Rule references;
- Act/ordinances;
- Regulations; and
- Cases cited.

Judgement details refer to all case details that would have initially been entered when the case was created. **Case history** requires a preceding case to be entered. The **parties and appearances** tab refer to all persons who take part in the current case as attorneys or advocates for applicants and respondents. The **judges** tab documents all judges who sat in on a case. A judge's title must be inserted and whether a judge delivered and was part of the majority. **Citations** require all parallel citations be added to the case. When a parallel citation is added, the citation's name, division, and country must be added. There must also be an indication if the citation is a primary citation or not. The **subjects** tab

keeps track of all legal concepts addressed within a legal case. The **words and phrases** tab keep track of any key words mentioned throughout the legal case. The **acts/ordinances** tab consists of all legislation referred to in a case. All instances of legislation are captured as separate records. As such, information such as an act/ordinance number, name, and section number must be captured. **Regulations** provide a list of all regulations that a court referred to**. Cases Cited** consists of all cases that have been cited in a current cast. Once all information has been entered, the BLC and Gracies database can be updated. The updated information is then made available on the live site, mylexisnexis.co.za.

4.6 Architecture of LegalCitator

Software architecture consists of a collection of components that interact with each other based on a specified pattern (Garlan & Shaw, 1993). Various architectural patterns exist but a common pattern used for business applications is a three-tier architectural pattern (Buschmann, Maunier, Rohnert, Sommerlad, & Stal, 1996).

A high-level design of the architecture used in LexisNexis' LegalCitator product was created by the researcher and vetted by experts at LexisNexis. The resultant architecture can be seen in Figure 4-9. Various servers and databases work in conjunction to ensure the smooth running of the LegalCitator A three-tier architectural pattern consists of layers for presentation, application logic, and data (Chen et al., 2003). LegalCitator uses a three-tier architectural pattern. The first tier is the presentation layer that consists of the screens that MyLexisNexis users and LexisNexis staff interact with. The presentation layer manages all user input, output and display of information. MyLexisNexis users interact with screens on the Mylexisnexis.co.za website through a web browser whilst LexisNexis staff users interact with screens on the LegalCitator system that is locally installed on staff computers. The second tier is the application logic layer that controls all business logic based on users' requests. The third tier is the database layer that stores and models data required by LegalCitator. The names of the servers in the architecture are as follows:

- Web server – ResearchWeb.01;
- Application server – ResearchWeb.02; and
- Database server – Research Database.

To display requested data for a MyLexisNexis user the following process is followed by LegalCitator:

1. User inputs a request via a web browser;
2. The request is sent via hyper-text transfer protocol (HTTP) to the ResearchWeb.01 web server;
3. The ResearchWeb0.1 web server passes the request on to the ResearchWeb.02 application server;
4. The appropriate services/functions within the application server are called and utilised based on the request;
5. The results from the services/functions are used to look-up the required data from the Research Database server;
6. The Research Database server sends the required data to the ResearchWeb.01 web server; and
7. The ResearchWeb.01 web server sends the response in HTML format to the MyLexisNexis user.

The process to display requested data for LexisNexis users is similar to the process followed for MyLexisNexis users. However, steps requiring the web server are not needed. As such, the following processes are followed:

1. Users inputs a request via locally installed LegalCitator program;
2. The request is sent to the ResearchWeb.02 application server;
3. The appropriate services/functions within the application server are called and utilised based on the request;
4. The results from the services/functions are used to look-up the required data from the Research Database server;
5. The Research Database server sends the required data to the ResearchWeb.02 web server; and
6. The output is displayed to the LexisNexis user.

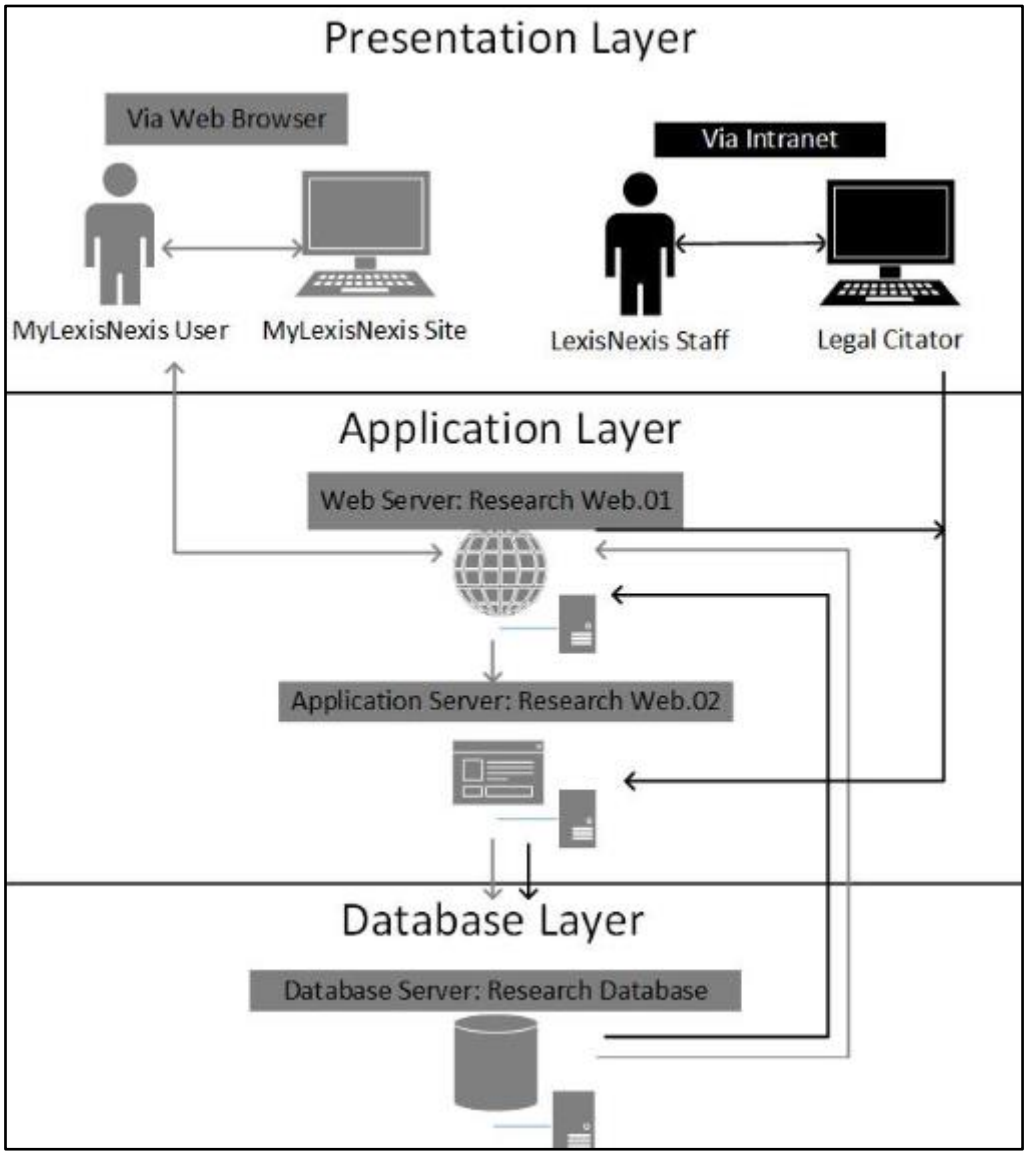


Figure 4-9: Architecture of LegalCitator System at LexisNexis

4.7 Problems Encountered in Processing Legal Cases at LexisNexis

This section reports on the problems that could be encountered when processing legal cases and, on the problems, experienced at LexisNexis. The informal manual process used to find the MAC in Section 4.5 was used to identify where problems from literature could occur.

4.7.1 Problems in Processing Legal Cases

Chapter 3 identified problems in literature that are encountered when processing text, namely:

- Ambiguity of words;
- Inconsistencies because of special formats, acronyms, and abbreviations;
- Unnecessary and confusing words; and
- Identifying meaningful keywords.

These problems will all be experienced when processing legal cases to recommend the MAC. Ambiguity can occur during the second and third processes of finding the MAC as search queries could contain ambiguous words that refer to the problem, research area, and a different concept. Processing legal cases for step 4 and step 5 can be hindered due to unnecessary and confusing words, and inconsistencies. This would require the text of the legal cases to be reduced to only the information needed to recommend the MAC. Additionally, inconsistencies can occur in steps 4 and 5 due to the style in which legal citations are written. Step 5 and 6 would require meaningful keywords to be identified. These keywords include the phrase “Applied to” and other actions taken on the list of cases referred to.

4.7.2 Problems Experienced at LexisNexis

Based on the findings from the questionnaires and email correspondence, LexisNexis experienced the following problems:

- Lack of access to proprietary systems due to licensing issues;
- No formal IR processes followed;
- Time wasted on searching for the MAC;
- No IE techniques are used;
- No query-independent ranking algorithms are used;
- LegalCitator does not have functionality to recommend the MAC; and
- Different formats of legal citations.

Licencing issue prevent LexisNexis from using existing proprietary systems. This results in LexisNexis having to find alternative options such as creating the systems themselves. Having no formal IR processes or functionality to recommend the MAC results in time wasted. Experts revealed that legal researchers must read through countless amounts of text before finding useful information. This is an issue that can be resolved by means of IE techniques and the use of query-independent ranking algorithms. IE could be used to identify and extract important facts from legal cases while query-independent ranking algorithms can be used to rank legal cases. The absence of formal processes prevents valuable resources from being allocated to other aspects of a legal researcher’s activities. Lastly, legal citations are formatted differently to each other and differently in other countries. This can result in issues when reading the legal citations and ultimately lead to more time wasted.

Table 3-15 can be expanded to include additional problems encountered at LexisNexis (Table 4-4):

Problem	Section	Author/Source
Ambiguity of words	3.4	Sumathy and Chidambaram (2013)
Inconsistencies because of special formats, acronyms, and abbreviations	3.4	Gurusamy and Kannan (2014)
Unnecessary and confusing words	3.2	Gurusamy and Kannan (2014)
Identifying meaningful keywords	3.2	Gurusamy and Kannan (2014)
Lack of access to proprietary systems due to licensing issues	4.4.2	LexisNexis Questionnaires
No formal IR processes followed	4.4.2	LexisNexis Questionnaires
Large amounts of time spent on searching for the MAC	4.4.2	LexisNexis Questionnaires
No IE techniques used at LexisNexis	4.4.2	LexisNexis Questionnaires
No query-independent ranking algorithms used at LexisNexis	4.4.2	LexisNexis Questionnaires
LegalCitator does not have functionality to recommend the MAC	4.4.2	LexisNexis Questionnaires
Different formats of legal citations (to each other and other countries)	4.4.2	LexisNexis Questionnaires

Table 4-4: Problems with Processing Text in the Legal Domain

4.8 Objectives, Requirements, and To-Be Processes for a MAC Model

The MAC Model will be a prescriptive model that uses IE to recommend the MAC and will from here on be referred to as the MAC Model. Table 4-5 tabulates the functional requirements that the MAC Model must meet whilst Table 4-6 tabulates the non-functional requirements. The MAC model should be able to make a recommendation of the MAC. However, to achieve this, facts from legal cases must be extracted and CRTs must be identified. After facts have been extracted, the facts should be saved into a database.

Number	Requirement
R1	Recommend the MAC for a field of law
R2	Extract data from legal cases
R3	Populate a database
R4	Identify CRTs
R5	Store extracted facts

Table 4-5: Requirements of a MAC Model

The MAC Model should reduce the amount of time spent by legal researchers on looking for information. Time can be reduced by extracting the important court case attributes (Table 4-2). The MAC Model must be able to process large amounts of legal cases quickly to help researchers get as much information possible in a short time. Lastly, the MAC Model should have an accuracy of 85% to eliminate the informal manual process that is used.

Non- Functional Requirement	Problem
Reduce research time	Large amounts of time spent on research
Process large amounts of legal cases quickly	Legal researchers would need to process multiple legal cases at once
Have 85% accuracy	To eventually eliminate using the manual, informal process to find the MAC

Table 4-6: Non-Functional Requirements

Table 4-7 maps the requirements to the features. R1 will require the implementation of IR intermediate stages (Section 3.2) and IE techniques (Section 3.4 to Section 3.7). R2 and R4 will require the an implementation of the IE process (Section 3.4) along with specific IE techniques (Section 3.5 and Section 3.7). R3 and R5 will require an IE process to first be implemented followed by the creation of a database (Section 3.8.1 and Section 3.8.2).

Requirement	Recommended Approach	Author
R1	To recommend the MAC, IR intermediate stages and IE techniques must be implemented.	Roshdi and Roohparvar (2015) Piskorski and Yangarber (2013) Choudhary and Burdak (2012)
R2	To extract data from legal cases, IE processes and techniques must be implemented	Abdelmagid et al. (2015) Piskorski and Yangarber (2013) Chopra et al. (2013)
R3	To populate the database, facts need to first be extracted.	Abdelmagid et al. (2015)
R4	To identify CRTs will require the implementation of Regular Expressions and NLP techniques such as tokenisation and stop-word removal.	Prasse et al. (2015) Goyvaerts & Levithan (2009) Piskorski and Yangarber (2013)
R5	To store extracted facts will require the implementation of a database.	Pokorný, Valenta, and Kovačič (2017) Robinson et al. (2015) MongoDB (2018c)

Table 4-7: Requirements Mapped to Recommended Approaches

To meet the requirements a set of processes must be followed (Figure 4-10 to Figure 4-13). Figure 4-10 illustrates at a high level the To-Be process that is represented by the MAC Model. The process is triggered by receiving a query for the MAC. This query will then be parsed through the processes of the proposed model. Once all processing is completed, the final output will be a recommendation of the MAC.

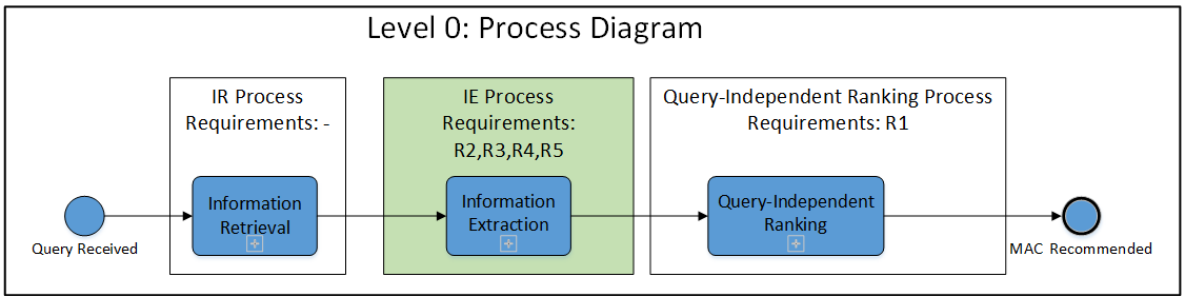


Figure 4-10: High Level Process of MAC Model

Figure 4-11 expands on the IR process of the proposed model. Once a query has been received, it must first be validated. An invalid query will require the user to enter another query. If the query is valid it will be parsed through to the Vector Space Model after which a set of ranked results will be returned to the user. This first round of ranking is based on the Vector Space Model.

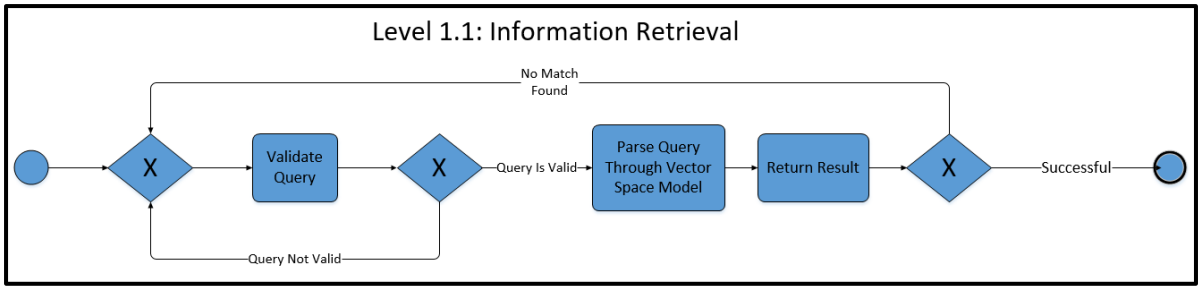


Figure 4-11: IR Process for MAC Model

Figure 4-12 expands on the IE process of the proposed model. Once a set of ranked results have been returned to the user, the user will then select cases he/she thinks will be appropriate. The selected cases will then have its content extracted, integrated, and saved to a database. The process will end if facts are either successfully or unsuccessfully saved to the database. The process would be unsuccessful if facts could not be saved to the database.

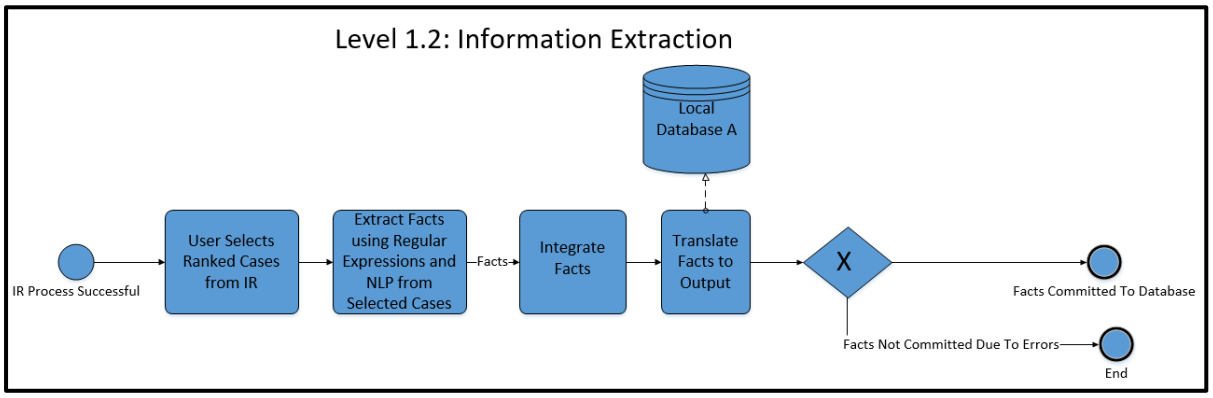


Figure 4-12: IE Process for MAC Model

Figure 4-13 expands on the query-independent ranking process of the proposed model. Once facts have been saved to the database, the database can then be queried, and the contents can be parsed through an adaption of the PageRank algorithm to perform the query-independent ranking of the cases. After cases have been ranked, they will be displayed to the user.

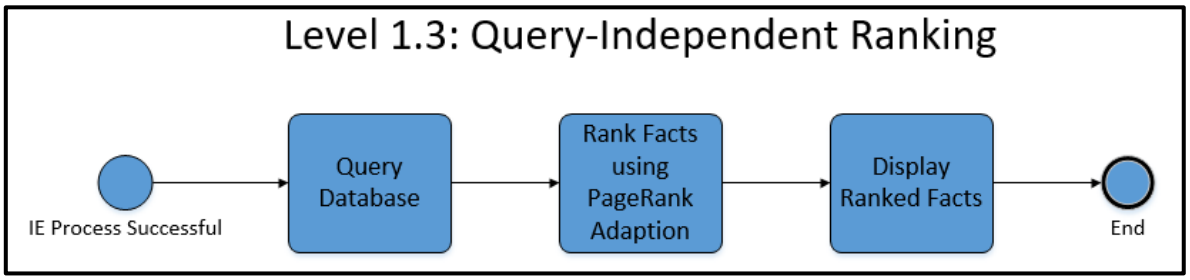


Figure 4-13: Case Ranking Process for MAC System

4.9 Conclusions

This chapter reported on the second and third activities of the DSR methodology namely, Definition of Objectives for a Solution and Design and Development. Analysis of a legal case revealed that different attributes of a legal case provided various information about the legal case. The general data and CRTs of a legal case can be extracted and used to aid in recommending the MAC. In particular, a legal case's division, case number, and the CRTs' action will be important to extract to aid in recommending the MAC. Questionnaires completed by experts from LexisNexis, revealed the problem

of recommending the MAC for a field of law (Table 4-4). Finding the MAC is a tedious process with no formal process followed. No formal processes were followed by LexisNexis in terms of IR and data was stored on various locations. Many of the processes followed such as entering data, searching for data, and determining the MAC were manual. Analysis of the existing systems at LexisNexis revealed that these systems were used for capturing data and searching for information.

This chapter fulfilled the following research objectives:

- **RO1**: *Identify the problems experienced when processing text as identified by literature and within a real-world context.*
- **RO2**: *Identify the attributes of a court case that can be used to aid in recommending the Most Applied Case.*
- **RO3:** *Determine what techniques and algorithms can be used to recommend the Most Applied Case.*

Coupled with the literature review and existing systems, the findings from the questionnaires aided in establishing a set of requirements for the proposed model to recommend the MAC. The proposed MAC Model will comprise of four processes, namely IR, IE, Information Storage, and Query-Independent Ranking, and will mimic the informal manual process followed by experts to recommend the MAC. The informal process highlighted which attributes of a court case can be used to recommend the MAC (Table 4-2).

By fulfilling these research objectives, five deliverables were provided. The deliverables are an expanded list of problems in processing text, the solution objectives and requirements (Table 4-5), the court case attributes that can be used to recommend the MAC (Table 4-2), the As-Is processes at LexisNexis (Figure 4-7) and the To-Be processes for the MAC Model (Figure 4-10 to Figure 4-13). The next chapter will report on the evaluation plan and the development of the prototypes based on the deliverables from this chapter.

Chapter 5: Development, Demonstration, and Evaluation

5.1 Introduction

The previous chapter investigated the real-world problem of finding the MAC within the legal domain and introduced the proposed MAC Model along with the MAC Model's requirements. This chapter continues to report on the third DSR activity namely, Design and Development, as well as the fourth activity, Demonstration (Figure 5-1). The proposed model will be presented (Section 5.2) along with software to be used (Section 5.3) and the architecture (Section 5.4).Different evaluation strategies are available for DSR (Section 5.5). The Incremental Prototyping Approach is used within the design cycle of DSR to create iterative prototypes (Section 5.6). An overview of the design and development of the prototype will be presented (Section 5.7). The main research objective will be investigated in this chapter:

- Ro_M : To develop a model to recommend the Most Applied Case for a field of law

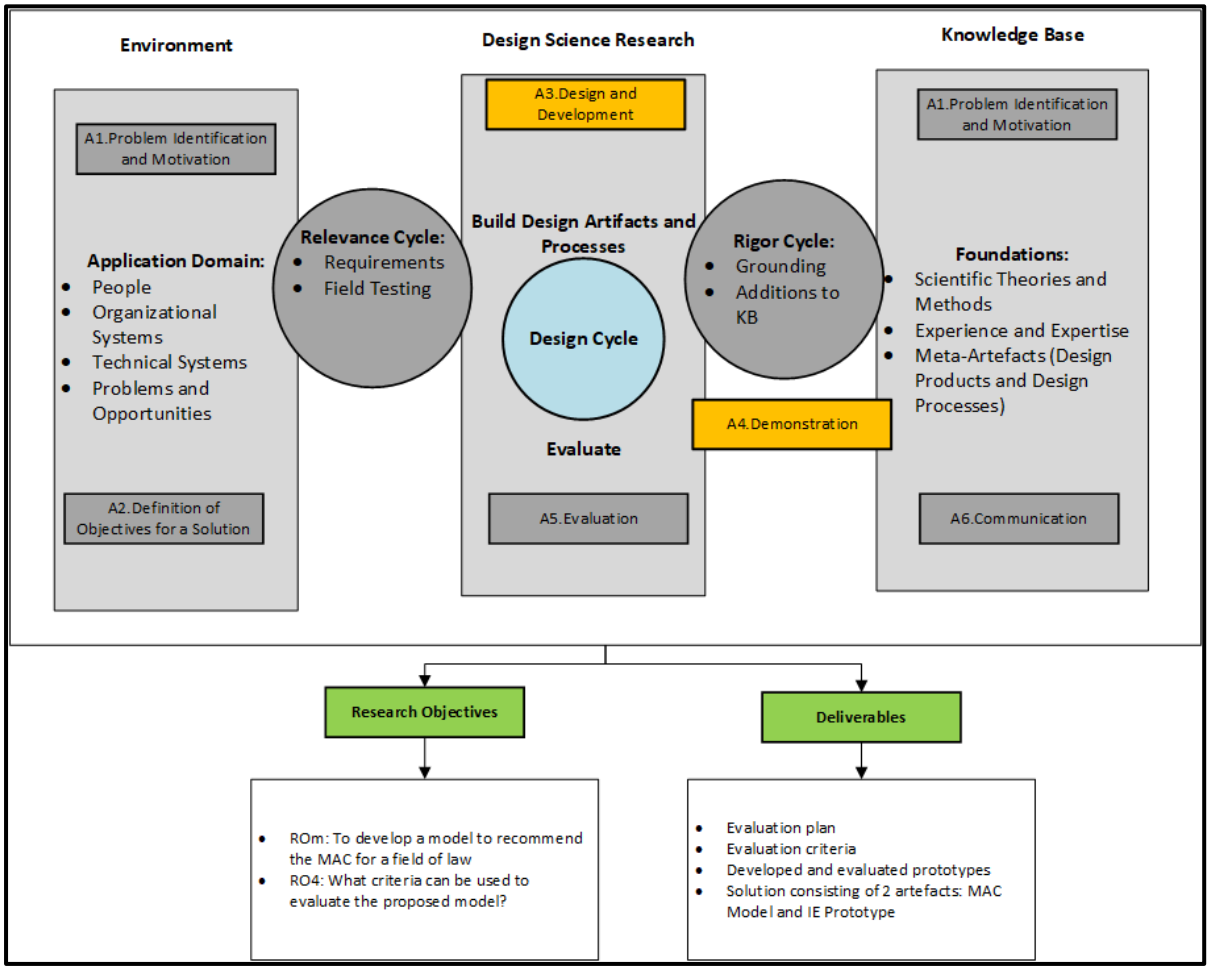


Figure 5-1: Chapter 5 DSR Activities

5.2 The Proposed MAC Model

The proposed MAC Model's IE process will be implemented as a proof-of-concept of the model in a prototype called the IE Prototype. Various technologies must be used to create the proposed MAC Model. The MAC Model is different from the IE Model in Figure 3-18 as the MAC Model is developed

based on the findings from literature in Chapter 3 and the requirements established in Section 4.8. The MAC Model can consist of four processes that address IR, IE, Information Storage, and Query-Independent Ranking and is illustrated in Figure 5-2.

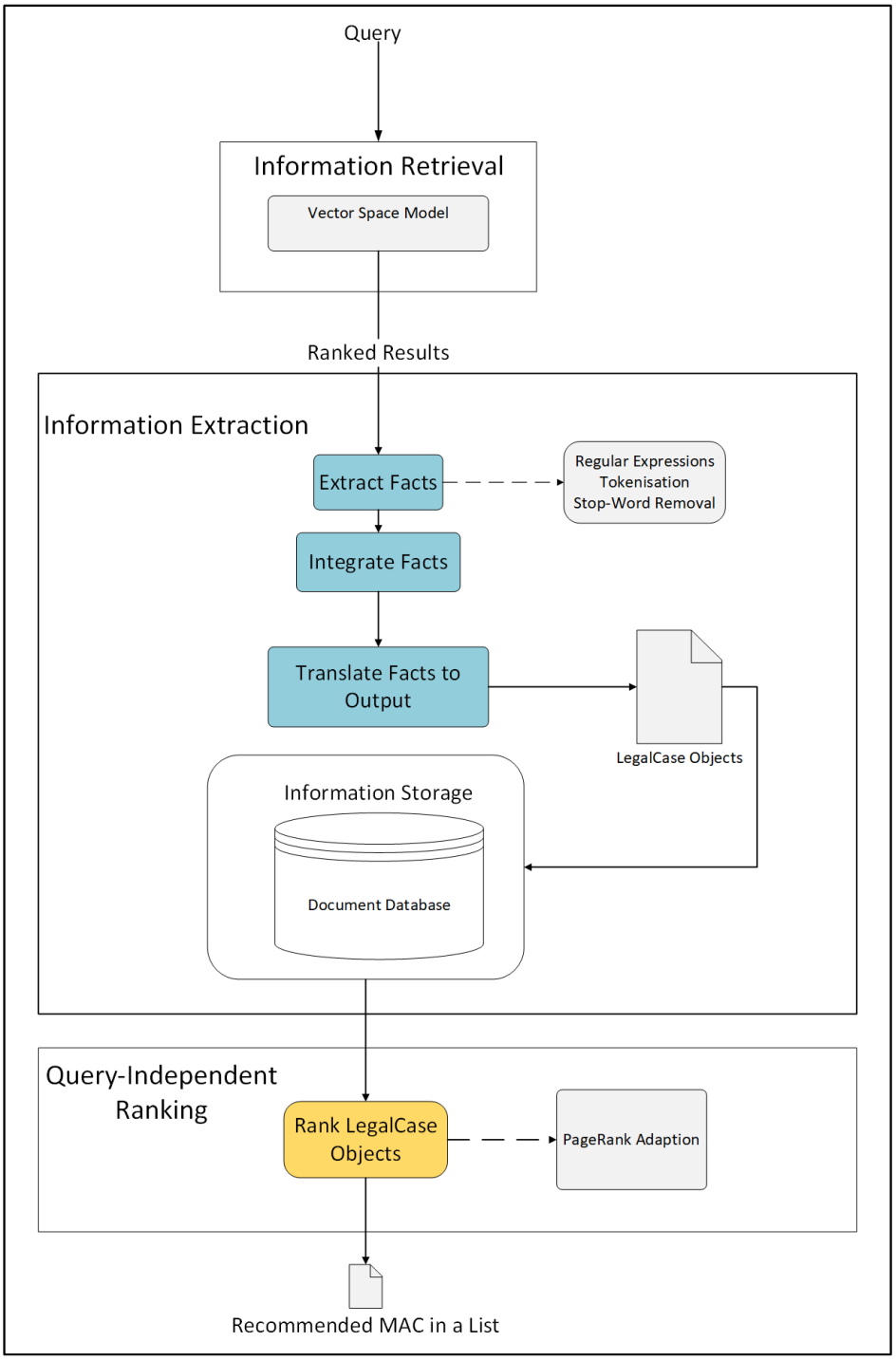


Figure 5-2: The MAC Model

The first process, IR, deals with the retrieval of cases based on a user's query (Section 3.2). The user's query will be parsed through an IR model after which a set of ranked legal cases will be returned to the user. The IR model selected is the Vector Space Model as it is the most commonly used IR model reported in literature (Table 3-2). This first form of ranking is known as query-dependent ranking as a query provided from a user is required. In the second process, IE, a user will select legal cases returned

from the IR process. Specific facts from the general data section and the list of CRTs of a legal case will then be extracted (Table 4-2). Facts will be extracted by using a combination of NLP techniques (Section 3.5) and regular expressions (Section 3.7). The NLP techniques will be used to process the text and the regular expressions will be used to extract facts from the processed text. The facts will be integrated and translated into LegalCase objects that will then be saved in the Information Storage process (Section 3.8).

The Information Storage process will make use of a graph database or document database to save LegalCase objects that are created from the IE process. Figure 5-3 illustrates the graph model used to model the graph database. ALL SA cases contain basic information about a case and a list of cases that were referred to. The graph database will store two types of nodes, namely 'Case' and 'Ref-To-Case' nodes. Nodes representing a 'Case' will contain properties for a case's title, date of case, division, and unique case number. The 'Ref-To-Case' nodes will contain properties for a referred-to case's title, year of case, the journal in which the case can be found, and the action taken on the referred-to case. 'Case' and 'Ref-To-Case' nodes will be connected by the action that a 'Case' node took on a 'Ref-To-Case' node. In Figure 5-3 the actions taken are applied, followed, and distinguished. Using a document database will result in the LegalCase objects being stored as documents that would have a similar structure to Figure 3-14. The fourth process, query-independent ranking, will rank and return the LegalCase objects created in a list with the first LegalCase object being the recommended MAC. An adaption of the PageRank algorithm can be used (Table 3-12).

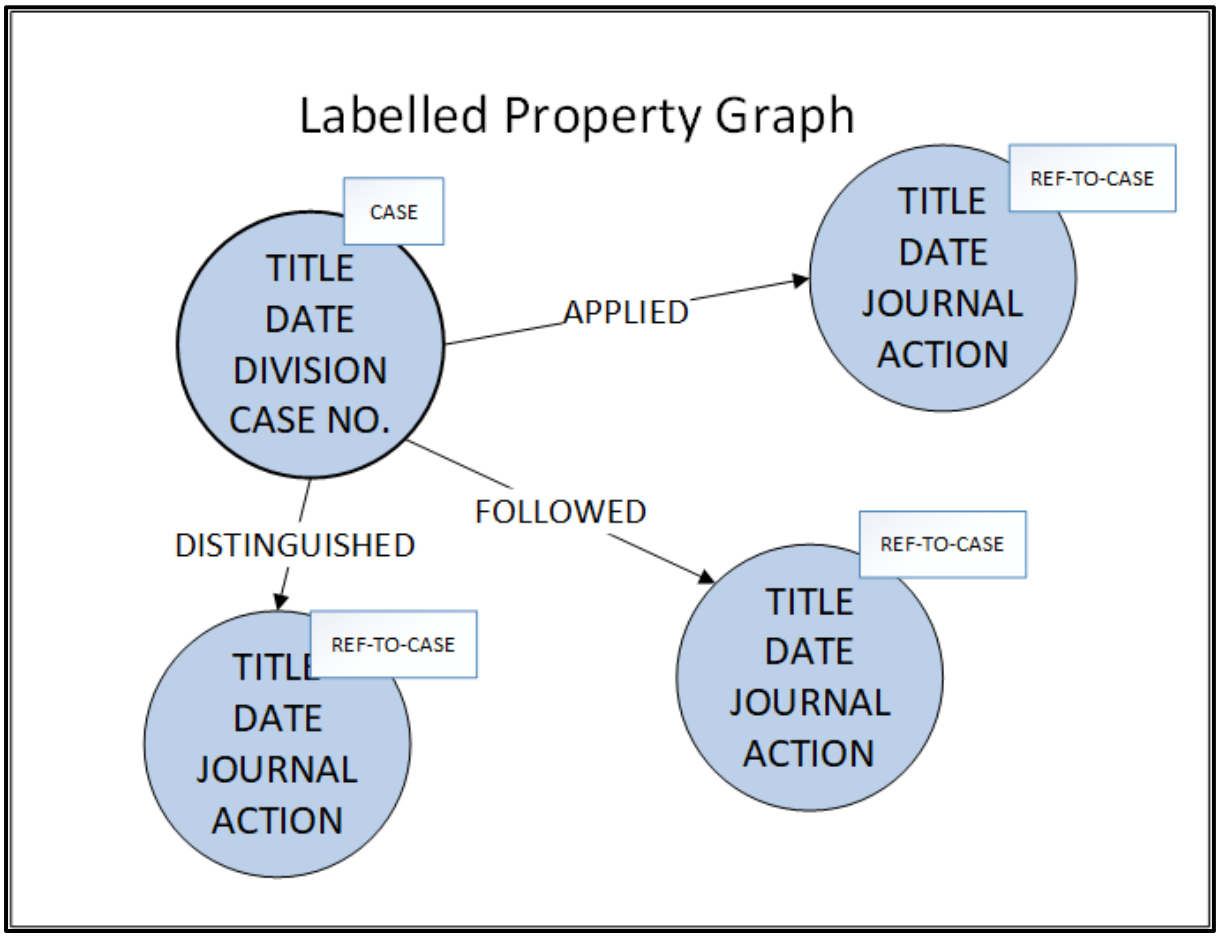


Figure 5-3: Graph Model of Graph Database

5.3 Summary of Software used in the IE Prototype

To create the IE Prototype, different technologies must be used and connected. The development language called Python will be used to create the IE process of the model. The IE process will use Python libraries RE, LXML, and Zipfile. The RE library will enable regular expressions to be created and executed on text in a legal case. The LXML and Zipfile libraries will allow .docx formatted legal cases' XML content to be accessed and parsed for extraction. To setup and run the databases, the Neo4j desktop application can be used for the graph database and the MongoDB Compass desktop application can be used for the document database. The Neo4j Python library will be used to allow the IE process to interact with the graph database. Similarly, MongoDB's Python library can be used to allow interaction between the IE process and the document database. Table 5-1 summarises the technologies used to create the IE Prototype.

Software	Technique Addressed	Use
Pycharm IDE	IE and Information Storage	To develop the proposed model using the Python language
Neo4j - desktop application	Information Storage	To setup the graph database
Neo4j - Python library	Information Storage	To allow for the IE process to interact with the Neo4j graph database
MongoDB	Information Storage	To setup the document database
MongoDB Compass -desktop application	Information Storage	To manage the document database
RE - Python library	IE	To create regular expressions
LXML - Python library	IE	To parse legal cases in .docx format
Zipfile - Python library	IE	To extract the XML contents of a .docx formatted legal case

Table 5-1: Technologies used to Create the MAC Model

5.4 Three Layered Architecture of a MAC System

The MAC system can be built using a three-tier architecture. Figure 5-4 illustrates how the processes of the MAC Model relate to the three layers and maps LexisNexis' architecture to the MAC Model.

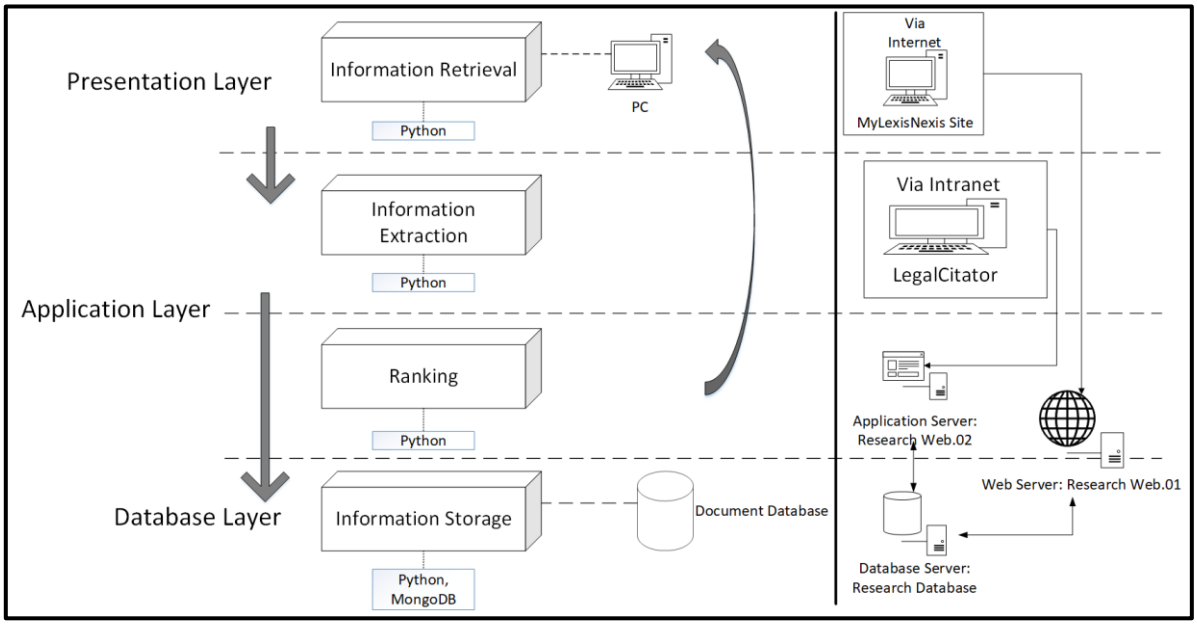


Figure 5-4: Architecture of the MAC System

In Figure 5-4 IR will form part of the presentation and application layers. The presentation layer will receive input from the user in the form of a query and allow the user to select cases. The query is then parsed through the IR process which will return cases for the user to select for processing. IE will then process the selected cases. IE will only be part of the application layer as the user will not have control over the extraction of the selected cases. The extracted facts are then processed to become LegalCase objects and parsed to the database layer to be saved. Information Storage will form part of the database layer and use a document database. The extracted facts are then parsed to a query-independent ranking algorithm. Query-independent ranking will form part of the application layer and return results to the presentation layer. Ranking will make use of an adaption of the PageRank algorithm to rank the selected cases and recommend the MAC. Once cases have been ranked, the output will be sent back to the presentation layer for the user to view.

Figure 5-4 also maps the architecture used by LexisNexis to the MAC System. LexisNexis users interact with either the Mylexisnexis.co.za website or the LegalCitator which are both found on the presentation layer. The Mylexisnexis.co.za website is linked to the Research Web.01 server while the LegalCitator is linked to the Research Web.02 application server. Both servers are connected to the Research Database server and pass information back to the presentation layer.

5.5 Evaluation Strategies and Methods for DSR

Evaluation is an essential activity when performing DSR. During evaluation, outputs such as design artefacts, theories, and information systems must be examined. These examinations act as evidence that a newly created artefact from DSR works or achieves the requirements for which it was designed (Venable, Pries-Heje, & Baskerville, 2012). When evaluating a design artefact various characteristics must be examined against the requirements of the artefact (Hevner, March, Park, & Ram, 2004). An appropriate strategy consisting of evaluation methods must be followed. Following a strategy will determine how well the design artefacts supports a solution to its assigned problem (Peffers et al., 2007).

The Framework for Evaluation in Design Science Research (FEDs) was developed to guide researchers in developing an appropriate strategy to evaluate artefacts that are created during DSR (Venable, Pries-Heje, & Baskerville, 2016). The FEDs Framework is built on two dimensions, namely the functional purpose of an evaluation and the paradigm of the evaluation. The two dimensions can make use of different evaluation methods and strategies (Figure 5-5).

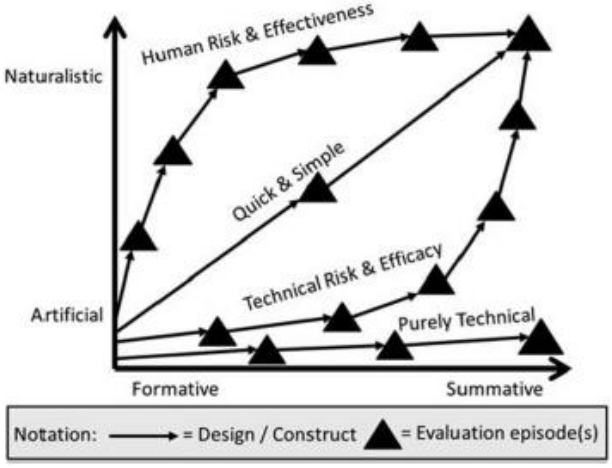


Figure 5-5: FEDS Framework with Different Evaluation Strategies (Venable et. al, 2016)

5.5.1 Functional Purpose of Evaluation

The functional purpose of an evaluation is the first dimension of the FEDs Framework. This dimension answers the question *Why an evaluation must occur.* Venable et al. (2016) identify two types of evaluations based on the first dimension, namely:

- Formative evaluations; and
- Summative evaluations.

Formative evaluations aid in iteratively improving an artefact that is being developed and produce empirically based interpretations that act as a basis for successful action to improve the characteristics or performance of an artefact. Summative evaluations determine the extent to which outcomes match the intended expectations and focus on meanings and support decisions that influence the selection of an artefact within a real-world context. Therefore, summative evaluations produce empirically based interpretations that provide a basis for creating shared meanings about an artefact (Venable et al., 2016).

5.5.2 Paradigm of Evaluations

The paradigm of evaluations is the second dimension of the FEDs Framework. This dimension answers the question *How to evaluate* by referring to various evaluation methods. Venable et al. (2016) differentiate between artificial and naturalistic evaluations for the paradigm dimension. Artificial evaluations can be empirical and non-empirical and are mainly used to test a design hypothesis. Artificial evaluations include:

- Laboratory experiments;
- Simulations;
- Criteria-based analysis;
- Theoretical arguments; and
- Mathematical proofs.

Naturalistic evaluations determine the performance of a solution in a real-world environment. Evaluating a solution in a real-world context allows for all complexities that exist in a real-world context to influence the evaluation. Naturalistic evaluations include:

- Case studies;
- Field studies;
- Field experiments;
- Surveys;
- Ethnography;
- Phenomenology;
- Hermeneutic methods; and
- Action research.

Choosing between artificial and naturalistic evaluations depends on the needs and resources of the DSR project (Venable et al., 2016). Artificial evaluation is generally a simpler, straightforward, and less costly form of evaluation. Artificial evaluation requires reductionist abstraction from a natural setting and can sometimes be unrealistic. Artificial evaluation is also said to be unrealistic in the sense that it does not involve real users, systems, or problems. Naturalistic evaluations can be more difficult as many confounding variables must be considered. However, naturalistic evaluations allow for critical face validity and rigorous assessment of effectiveness of an artefact.

5.5.3 FEDs Evaluation Strategies

The FEDs Framework has four different strategies that can be implemented by the researcher, namely (Venable et al., 2016):

- Quick and simple;
- Human risk and effectiveness;
- Technical risk and efficacy; and
- Purely technical.

The **quick and simple strategy** involves minimal formative evaluation and moves quickly to summative and naturalistic evaluation. This strategy requires only a few evaluation episodes and is low in cost. Additionally, the quick and simple strategy promotes quick project conclusion and might not be reasonable to follow when faced with design risks. The **human risk and effectiveness strategy** uses formative evaluations and progresses towards summative evaluations at the end of the strategy. Formative evaluations begin with artificial formative evaluations but later change to naturalistic formative evaluations. Towards the end of the human risk and effectiveness strategy, summative evaluations are used to rigorously asses the effectiveness of the artefact.

DSR Evaluation Strategy	About Strategy	Circumstance Selection Criteria	Functional Purpose	Paradigm of Evaluation
Quick and Simple	Minimal formative evaluation; Prefers summative and naturalistic evaluation; Few evaluation episodes; Low in cost; Promotes quick project completion.	If a small and simple construction is needed; and If there is low social, technical risk, and uncertainty.	Formative Evaluation; and Summative Evaluation.	Naturalistic
Human Risk and Effectiveness	Starts with formative evaluations; Ends with summative evaluations.	If the design risk is social or user orientated; If it is cheap to evaluate with real users in real context; and If the critical goal of the evaluation is to establish that the benefit of the artefact will continue in a real-world context.	Formative Evaluation; and Summative Evaluation.	Artificial; and Naturalistic.
Technical Risk and Efficacy	Starts with formative evaluations; Uses formative artificial evaluations; Progresses to summative artificial evaluations; valuations.	If major design risk is technically orientated; If its prohibitively expensive to evaluate with real users within a real-world context; and If it is a critical goal to determine if the benefits experiences are due to the artefact and not something else.	Formative Evaluation; and Summative Evaluation.	Artificial; and Naturalistic.
Purely Technical Artefact	Involves no human users at all.	If artefact is purely technical.		Artificial

Table 5-2: Summary of FED Strategies Adapted from Venable et al. (2016)

The **technical risk and efficacy strategy** uses iterative artificial formative evaluations but later changes to summative artificial evaluations. Using summative artificial evaluations allows for the efficacy of an artefact to be determined. Towards the end of this strategy, naturalistic evaluations are used. The **purely technical strategy** is only used if no human users are required. This strategy favours artificial evaluations over naturalistic evaluations. A summary of the four strategies as well as when to choose each strategy is provided in Table 5-2.

5.6 Incremental Prototyping Approach and Evaluation Plan

This section will report on the incremental prototyping approach for creating prototypes. An evaluation plan for the prototypes will then be presented.

5.6.1 Incremental Prototyping

A prototype is an estimated version of a product of interest that can follow one or more dimensions. The first dimension refers to the extent to which a prototype is physical as opposed to analytical. The first dimension implies that a prototype can be tangible or intangible. Tangible prototypes are physical artefacts of a product that are built for testing and experimentation whilst intangible prototypes are used for analytical purposes and are usually mathematical, visual, or computer simulations. The second dimension refers to the extent to which a prototype is comprehensive as opposed to focused. Comprehensive prototypes implement the majority, if not all, of a product's attributes. Comprehensive prototypes can therefore be fully-scaled and fully operational versions of a product. Focused prototypes only implement one or few attributes of a product (Ulrich & Eppinger, 2012). For this research, an analytical, non-tangible, focused prototype will be created by means of incremental prototyping.

Incremental prototyping involves the gradual evolution of an artefact through building individual prototypes (Carr & Verner, 1997). Each prototype requires phases related to requirements, design, implementation, and testing to be followed. Therefore, implying that a working version of the artefact is produced from the first prototype onwards. Various advantages are obtained by following incremental prototyping such as (Sarker, Faruque, Hossen, & Rahman, 2015):

- Working software is generated quickly;
- Easier to test and debug;
- Managing risk is easier; and
- Provides flexibility.

Working software is produced as each prototype must go through requirements, design, implementation, and testing phases. **Testing and debugging** prototypes become easier as developers only need to concentrate on one particular prototype that has limited code. **Managing risk** becomes easier as all areas susceptible to risk are identified and handled during each iteration. In terms of **flexibility**, changing an artefact's scope and requirements becomes less costly.

5.6.2 Functional Purpose, Paradigm, and Strategies

The functional purpose of conducting the evaluations in this research is to evaluate two artefacts, namely the theoretical MAC Model and the practical artefact (the IE Prototype). The theoretical model must be evaluated to determine how well the model meets the requirements of recommending the MAC. The practical artefact must be evaluated to determine if there will be any problems with the prototype's design.

The theoretical MAC model was designed based on literature, extant systems analysis, and questionnaires completed by experts from LexisNexis (Section 5.2). The practical artefact was developed as a proof of concept of the MAC Model's IE process. The practical artefact, hereafter referred to as the IE Prototype, consists of two processes namely, the IE process and the Database process. Evaluation of the IE Prototype will determine how well the chosen techniques and algorithms (RO3) are in performing IE to recommend the MAC.

The evaluation strategy selected for this research is the technical risk and efficacy strategy due to the technical nature of the research problem. The technical risk and efficacy strategy was also chosen as it will be too expensive to solely evaluate the artefacts with real members from LexisNexis. The technical risk and efficacy strategy focuses on formative evaluations in the start of evaluations but progressively moves to summative evaluations. The prototypes will be evaluated by conducting iterative formative evaluations and summative naturalistic evaluations.

The aim of having iterative formative evaluations is to detect and eliminate any potential functional issues that the prototypes could incur. Once development of the IE Prototype has finished, summative-naturalistic evaluations will occur. Summative-naturalistic evaluations will consist of a set of experiments to determine the prototypes' performance under real-world conditions. These experiments can be made up of real-world cases that were used by legal researchers to find the MAC for a legal dispute.

Five evaluation techniques will be followed throughout the summative-artificial evaluations:

- Analytical;
- Experiment;
- Observational;
- Descriptive; and
- Testing.

For **Analytical**, the practical artefact's architecture, optimisation, and dynamic ability will be tested. The artefact's architecture must be evaluated to determine how well the artefact will fit into LexisNexis' technical architecture. The artefact must be tested to determine its bounds of optimality. In terms of being dynamic, the code will be debugged and analysed to eliminate unnecessary components and ensure that the artefact is compatible with other software that forms part of the architecture. With regards to **Experiment**, controlled experiments will be conducted to ensure that all properties of the practical artefact are evaluated. For **Observational**, a case study will be used. Examples of previous cases used for a dispute can be obtained from LexisNexis and put through the artefact to determine the efficiency and accuracy of the IE Prototype. **Descriptive,** well designed scenarios from the observational technique can also be used to show the artefact's use. In terms of **Testing**, both functional and structural testing must be conducted to ensure that the practical artefact's architecture is sound and that all sections of the code work properly. Following these techniques will in turn allow for different properties to be evaluated.

5.6.3 Evaluation Criteria

Three criteria identified by Jouili and Vansteenberghe (2013), Chen (2016), Kabakus and Kara (2017), and Frekjmr, Hertzum, and Hornbaek (2000) can be used to evaluate the performance of the IE Prototype (Table 5-3). The three criteria are:

- Effectiveness;
- Scalability; and
- Execution time.

The first criterion, **effectiveness**, refers to the degree a system can achieve specific goals (Frekjmr et al., 2000). Effectiveness of the IE Prototype can be determined by measuring the accuracy. Accuracy refers to the closeness of agreement between an observed value and the actual value (Menditto, Patriarca, & Magnusson, 2007). The output produced for each legal case from the IE Prototype will be measured against the expected output. The following equations are used to determine the accuracy:

$$Xi = \frac{Ai}{Bi}$$

Equation 5.6-1: Difference Ratio

Where:

Xi = the difference ratio for a legal case I;

Ai = the CRT output for legal case I that differs from the expected CRT output for legal case I; and

Bi = the expected CRT output for legal case i.

$$Y = \frac{\sum_{i=1}^{n} A_i}{\sum_{i=1}^{n} Bi}$$

Equation 5.6-2: Total Difference Ratio

Where:

Y = total difference ratio for CRTs;

Ai = the total CRT output for legal case I that differs from the expected CRT output for legal case I; and

Bi = the total number of CRT output for legal case i.

An ideal value for the difference ratio is 0 indicating that there is no difference between the output and the expected output. For this research, an error margin of 10% will be applied, implying that a difference ratio of 0.1 or less will be acceptable (Conroy, 2016).

The second criterion, **scalability,** refers to a system's ability to accommodate and process an increasing work load (Bondi, 2000). Scalability can be tested by putting different workload sizes through the IE Prototype (Jouili & Vansteenberghe, 2013). This will require a connection to the database to be established followed by the time taken to populate the database with data to be recorded. Chen (2016) further states that a scalable system can result in a higher maturity. Implying that the system can handle more users.

The third criterion, **execution time**, refers to the time taken to perform actions in a system or database (Chen, 2016). For this research, execution time will be recorded for the two processes of the IE Prototype namely, the IE process and Database process. For the IE process, time will be recorded for extracting and creating LegalCase objects. For the Database process, time will be recorded for writing the LegalCase objects as key-value pairs and inserting them into the database (Kabakus & Kara, 2017). Shorter processing times can indicate an efficient and better performing system or database. This in turn can have an impact on scalability.

All three criteria can affect each other. The effectiveness of the IE Prototype can affect the scalability and execution times. An inaccurate IE Prototype could result in unnecessary extractions resulting in

more workloads for the prototype to process. Execution time can affect the scalability as longer execution times will mean that larger workloads will take longer to process. Table 5-3 summarises the evaluation criteria for the IE Prototype.

Evaluation Criteria				
Criteria	**Importance**	**How to Test**	**Effect on Other Criteria**	**Author**
Effectiveness	To determine how accurately the IE Prototype achieves its goals of performing IE and populating the database.	Observe the number of extractions obtained versus the actual number of extractions.	Can affect the scalability and execution times of the IE Prototype.	Frekjmr, Hertzum, and Hornbaek (2000)
Scalability	To determine the IE Prototype's ability to handle different sized loads of data.	Parse different sized workloads through the IE Prototype.	Can contribute to maturity.	Jouili and Vansteenberghe (2013)
Execution Time	Smaller execution times can result in a more responsive system or database.	Use different sized workloads	Can affect scalability.	Chen (2016) Kabakus and Kara (2017)

Table 5-3: Evaluation Criteria for an IE Prototype

5.7 Overview of Prototypes and Evaluation

In accordance with incremental prototyping approach, four phases were followed when creating the IE Prototype. These four phases related to requirements, design, implementation, and testing. Two iterations were followed to create the IE Prototype. Table 5-4 provides a summary of the evaluations conducted.

Dimension	What is addressed	Evaluation Type	Methods
Functional Purpose	Why the IE Prototype must be evaluated	Iterative Formative Summative -Naturalistic	Incremental Prototyping
Paradigm of Evaluation	How the IE Prototype will be evaluated	Artificial Naturalistic	Laboratory Experiments Case Study

Table 5-4: Evaluation Summary

Table 5-5 and Table 5-6 summarises the experiments and evaluation process that were conducted for the IE Prototype. The IE Process was implemented over an iterative formative evaluation through three experiments and investigated web scraping, regular expressions, tokenisation, and stop-word removal. A summative naturalistic evaluation was also conducted through two experiments of the IE process. The Database process was also implemented over an iterative formative evaluation. The experiments for the Database process evaluated the graph and document databases.

Iteration	Evaluation Type	Process	(N)	Techniques	Documents
1	Iterative Formative	IE Process	3	Web Scraping Information Regular Expressions Tokenisation	TD (PDF and .docx) T1 – T3 U1 – U10
	Summative Naturalistic	IE Process	2	Regular Expressions Tokenisation Stop-Word Removal	U1 – U10
2	Iterative Formative	Database Process	1	Graph Database	-
	Summative Naturalistic	Database Process	2	Document Database	U1 – U10

Table 5-5: Experiment Summary

A total of 15 documents were used for testing. TD PDF and TD.docx were documents created by the researcher to use as initial tests for performing IE. Documents T1 to T3 were legal cases obtained from LexisNexis and used to build the MAC System. Documents U1 to U10 were unseen legal cases also obtained from LexisNexis and used to test the MAC System.

The evaluations will use the technical risk and efficacy strategy as well as iterative formative evaluations and summative-naturalistic evaluations. Experiments and testing will be done to determine the IE Prototype's effectiveness/accuracy and scalability. Execution time will be recorded during the evaluation of the IE Prototype as more documents will be used during the evaluation as opposed to the 15 documents for experiments.

Evaluation Type/Strategy	Purpose	How it is Done	Metrics
Technical Risk and Efficacy Strategy	To determine the Benefits Experienced of the Artefact	Formative and Summative Evaluations	Effectiveness/Accuracy Scalability
Iterative Formative Evaluations	To detect and eliminate any functional issues in prototypes	Experiments/Testing	
Summative-Naturalistic Evaluations	To determine prototype's performance under real-world conditions	Experiments	

Table 5-6: Evaluation Process

5.7.1 Iteration 1

Iteration 1 of the IE Prototype consisted of five experiments. The first experiment made use of web scraping while the remaining four experiments performed IE directly on legal case documents. The libraries used to run the experiments were Selenium, Tika, RE, Lxml, Neo4j, and MongoDB. The IE Prototype addressed requirements R2, R3, R4 and R5. Tika was not investigated in Chapter 3 as it was only used to parse the contents of a PDF document. Table 5-7 summarises the experiments that were conducted for the formative evaluation of iteration 1 and Table 5-8 summarises the experiments that were conducted for the summative evaluation of iteration 1.

Experiment	Technique	Processes	Result
1: Web Scraping	Web Scraping Information	(1) Extract Facts	Unsuccessful Extraction
2: Performing IE Directly on PDFs	Regular Expressions Tokenisation	(1) Extract Facts	Successful Extraction
			Unsuccessful Extraction
3: Performing IE Directly on .docx	Regular Expressions Tokenisation Stop-Word Removal	(1) Extract Facts; (2) Integrate Facts; and (3) Translate Facts to Output.	Successful Extraction
			Successful Extraction but Additional Cleaning Required

Table 5-7: Summary of Formative Evaluations for Iteration 1

Experiment	Round	Technique	Processes	Test Documents	Result
4: Testing on Unseen Legal Cases, Cleaning, and Additional Programming	First Test	Regular Expressions Tokenisation Stop-Word Removal	(1) Extract Facts; (2) Integrate Facts; and (3) Translate Facts to Output.	T2 and T3	Partial Extraction
	Data Cleaning Part 2				Cleaned up to a point. Minor inconsistencies prevented complete cleaning.
	Additional Programming				Improved regular expressions to cater for different legal cases
	Second Test				Successful extraction but minor inconsistencies prevented ideal extraction
5: Testing on Unseen Legal Cases	Round 1	Regular Expressions Tokenisation Stop-Word Removal	(1) Extract Facts; (2) Integrate Facts; and (3) Translate Facts to Output.	U1-U10	CRTs extracted but changes to the MAC System can bring better results
	Round 2				Results improved but CRTs without any action are merging with other CRTs
	Round 3				Successful extraction of CRTs

Table 5-8: Summary of Summative Evaluations for Iteration 1

A summary of the documents used in the experiments is provided in Table 5-9. The documents used for each experiment is highlighted as well as a description of the experiment.

Document Name	Used in	Details
TD PDF and TD.docx	Experiment 2 Round 1	TD was created by the researcher to test extracting facts. TD was in PDF format. Facts related to date, title, mobile number, email address, and web address were to be extracted.
	Experiment 3 Round 2	TesterDoc1 was converted to MS Word .docx (TD.docx) format to perform IE on.
T1	Experiment 2 Round 2	T1 was a legal case obtained from LexisNexis. T1 is in PDF format and was used as a base to build the MAC System to extract facts.
	Experiment 3 Round 2	T1 was converted to a MS Word .docx file to perform IE on.
T2	Experiment 4	T2 was a legal case obtained from LexisNexis. T2 was used to further build the MAC System as it provided a different structure than T1.
T3	Experiment 4	T3 was a legal case obtained from LexisNexis. T3 was used to further build the MAC System as it provided a different structure than T1 and T2.
U1-U10	Experiment 4	U1-U10 were used as unseen cases to test the MAC System.

Table 5-9: Summary of Test Documents used in Experiments

5.7.1.1 Iteration 1: Experiment 1: Web Scraping

The aim of using web scraping was to access legal cases from the Mylexisnexis.co.za website and extract the facts from the legal cases online. To achieve the extraction, a script was written that could automatically login to the Mylexisnexis.co.za website and proceed to locate and download the required files. Two methods were used to perform the automatic login. The first method required the login details to be sent via a Python dictionary to the Mylexisnexis.co.za website. The login details required were the username, password, and a security token. The first method did not work as security protocols on the Mylexisnexis.co.za website prevented the automatic login from occurring. It is likely that submitting the security token caused the security protocols to deny access. A second method was used to work with the security protocols. During the second method a library called Selenium was used. Automatic login with Selenium was successful as only the username and password were required. However, the experiment was unsuccessful as the legal case files could not be accessed as the script was unable to locate the required HTML tags containing the legal cases.

5.7.1.2 Iteration 1: Experiment 2: Performing IE Directly on Legal Case Documents

The aim of experiment 2 was to load and extract facts from documents using the libraries Tika and RE. Initially, facts were loaded and extracted from a PDF document created by the researcher (Appendix H). The Tika library was used to load and parse the PDF document's contents and the RE library was used to create and apply regular expressions to extract specific facts from the content. Facts related to date, title, mobile number, email address, and web address were successfully extracted.

After facts from the PDF created by the researcher were loaded and extracted, a set of three test legal cases, referred to as T1 toT3, were then tested. The test legal cases were obtained from LexisNexis. Following successful extraction of T1, cases T2 and T3 were ran through the program. Facts from T1 such as the case's title, division, date, case number, before, and CRTs were extracted by using regular expressions. However, errors were encountered when trying to extract the CRTs from T1. Multiple attempts were made at altering the regular expressions and tokenising the text, but none were successful. After additional research was conducted it was found that parsing PDF documents to perform IE are not ideal as PDFs are inconsistently formatted or the text of the PDF can be images. In

this case, T1, T2 and T3 were found to have inconsistent formatting. These inconsistencies made it difficult to extract all required facts from the legal cases and as such another approach was required. This resulted in an unsuccessful extraction of PDF formatted legal cases. Figure 5-6 illustrates the results from performing IE on T1. Figure 5-6 shows the facts extracted for T1 namely, the title, division, date, case number, and the judges who heard the case. The value "None" is displayed for the CRTs that should have been extracted.

```
Minister of Basic Education and others v Basic Education for All and others [2016] 1 All SA 369 (SCA)

Division: SUPREME COURT OF APPEAL

Date: 2 December 2015

20793/2014
Before: A CACHALIA, N DAMBUZA, CH LEWIS, MS NAVSA and XM PETSE JJA

None

Process finished with exit code 0
```

Figure 5-6: Results from Experiment One Phase Two

5.7.1.3 Iteration 1: Experiment 3: Performing IE Directly on Legal Case Documents

Experiment 3 used the libraries Lxml and RE. The first aim of Experiment 3 was to load and extract facts from a Microsoft (MS) Word .docx formatted document. Experiment 3 first used a MS Word .docx document that was created by the researcher. The Lxml library was used to parse the contents of a MS Word document and the RE library was used to create and apply regular expressions on the document's content. Facts related to email address and dates were to be extracted. The method used for Experiment 3 is different from Experiment 2 as a MS Word document in .docx format is parsed instead of a PDF document. A MS Word .docx document is essentially like a zip file that contains multiple files. The file of interest is the 'word/document.xml' file as it contains the text of the document in different tags. Experiment 3 extracted the text from the 'word/document.xml' file and parsed the text into a tree. The tree was then processed to look for and extract 'paragraph' tags. The paragraph tags were processed, and regular expressions were applied to the paragraphs to extract required facts. The overall process followed to perform IE on the legal documents is illustrated in Figure 5-7.

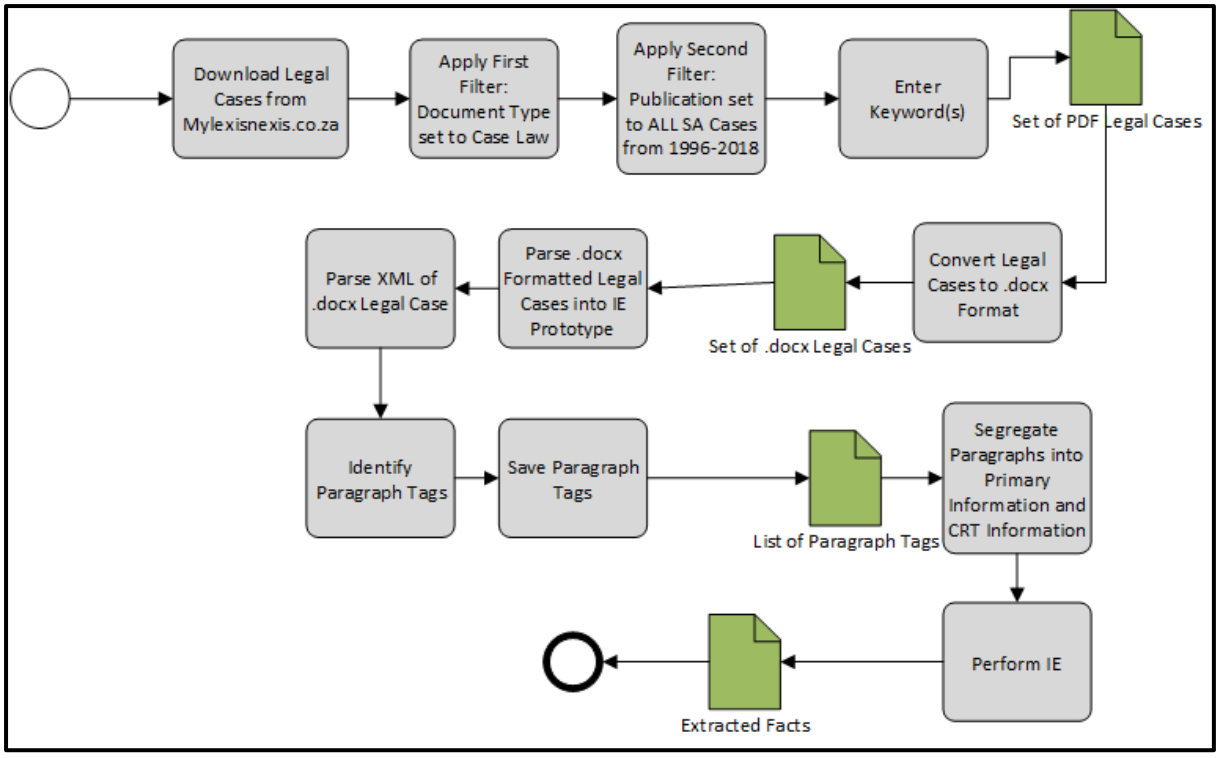


Figure 5-7: The IE Process of the MAC Model

The second aim was to clean the text and perform extraction on T1 as a MS Word .docx formatted legal case. The cleaning process required stop-words to be removed before performing IE. It was found that legal cases can contain large amounts of text. To avoid applying the regular expressions on all words in the text, the required sections, namely the general data and CRTs, containing the facts for extraction of the legal case were identified and processed separately from the rest of the text. The general data and CRTs were successfully extracted and returned in the form of lists. However, additional cleaning of the extracted facts was required to save the facts to a database. Figure 5-8 illustrates the results from using the Lxml and RE libraries.

```
-------------------------Extracted information-------------------------
Minister of Basic Education and others v Basic Education for All and others [2016] 1 All SA 369 (SCA)
['Division:SUPREME COURT OF APPEAL']
['Date:2 December 2015']
['Case No:20793/2014']
['Cases referred to in judgment', 'Governing Body of the Juma Musjid Primary School and others v Essay NO', u'and others (Centre for Child Law and anoth
Cases referred to in judgment Governing Body of the Juma Musjid Primary School and others v Essay NO and others (Centre for Child Law and another as ami
```

Figure 5-8: Results from Experiment Two Phase Two

5.7.1.4 Iteration 1: Experiment 4: Cleaning Extracted Facts and Additional Programming

Experiment 4 required the program to run on the remaining test cases, T2 and T3, and perform additional cleaning so that the facts could be translated into LegalCase objects and populated in the database. When testing with T2 and T3, it was found that some facts were not being extracted or were not extracted correctly. This occurred for two reasons, namely the regular expressions were not designed to handle text that was in different formats, and the legal cases were inconsistently formatted. The issue was overcome using two approaches. The first approach altered or created new regular expressions to cater for text in the different formats. The second approach performed a second round of cleaning in which additional stop-words were removed. Extraction of the general data and CRTs from T2 and T3 was partially successful.

Upon further inspection, it was found that the CRTs would require separate processing to be inserted into the database. This required the section of the legal case regarding CRTs, to be isolated from the rest of the text and have separate regular expressions created to perform the extraction. The CRTs for T1 were successfully extracted. However, when testing T2 and T3 not all the CRTs were extracted. This required additional regular expressions to be written and a third round of cleaning to remove unnecessary words and characters. T2 and T3 were then retested with the result that all CRTs were extracted but some unnecessary words were extracted and included with the CRTs. The final set of unnecessary words and characters occurred due to the inconsistency of the legal case's formatting, but this should not prevent all extracted facts from being inserted into a database. Table 5-10 summarises the facts extracted from T1 to T3 where a tick indicates extraction and a tilde indicates partial extraction.

	Legal Cases		
Primary Attribute	**T1**	**T2**	**T3**
Case Name	✓	✓	✓
Case Division	✓	✓	✓
Case Date	✓	✓	✓
Cases Referred To Attribute	✓	~	~

Table 5-10: Attributes Extracted from the Legal Cases (T1, T2, and T3)

5.7.1.5 Iteration 1: Experiment 5: Testing MAC System on Unseen Legal Cases

For Experiment 5, ten legal cases (U1 to U10), were downloaded from the Mylexisnexis.co.za website. The purpose of using unseen legal cases was to determine how effective the IE Prototype performed on data that it had not previously processed. Different keywords were used to return and download the unseen legal cases. The downloaded legal cases were then converted to MS Word .docx format and parsed through the IE Prototype. An expected number of results for the general data and CRTs was compared to the actual results obtained and various calculations based on Equation 5.6-1 and Equation 5.6-2 were made to determine the IE Prototype's effectiveness. Table 5-11 summarises the number of CRTs present in each unseen legal case that was to be extracted.

Document	**Number of CRTs**
U1	36
U2	12
U3	28
U4	2
U5	20
U6	29
U7	16
U8	11
U9	8
U10	4

Table 5-11: Number of CRTs to be extracted from Unseen Cases

Three rounds of experiments were conducted during Experiment 5 that used the legal cases U1 to U10. The first round of experiments parsed the unseen legal cases through the IE Prototype to obtain and analyse the results. Round one revealed that improvements to the IE Prototype were required. Particularly to the part of the IE process that deals with extracting the CRTs. Extractions were categorised as being either perfectly extracted, partially extracted, or not extracted at all. Perfect extractions implied that the specific property was extracted without any issues and contained no unnecessary text. Partial extractions referred to properties that were extracted but contained some unnecessary text. No extractions implied that a property was not extracted by the IE Prototype. Table

5-12 summarises the number of extractions performed for the general data for all the unseen legal cases for Rounds 1,2, and 3.

Result	Round 1 and 2 Frequency (n)	Round 3 Frequency (n)
Name extracted perfectly	8	8
Name extracted partially	1	1
Name not extracted	1	1
Division extracted perfectly	8	9
Division extracted partially	0	0
Division not extracted	2	1
Date extracted perfectly	8	9
Date extracted partially	0	0
Date not extracted	2	1
Case Number extracted perfectly	5	9
Case Number extracted partially	0	0
Case Number not extracted	5	1

Table 5-12: Number of Extractions for General Data of Unseen Legal Cases

During the first and second round the frequency for the primary attribute extractions remained constant. For the first and second rounds most of the name, division, and date properties were extracted perfectly. Each of these properties had 8/10 perfect extractions. The extraction for the case number was split evenly between being perfectly extracted and not extracted at all. During the third round, minor changes to the IE Prototype were made to improve the extraction. The result of the third round saw better results for the name, division, and date attributes. A big change was seen with the number of case number attributes that were extracted. There was an increase from 5 to 9 case numbers extracted. Following these extractions are the CRT extractions for the first round that are summarised in Table 5-13. The first round's CRT extraction results were mixed. Some unseen legal cases had large differences between the expected number of CRTs to be extracted and the actual number of CRTs that were extracted. This required the IE Protype's code to be inspected, altered, and run through the unseen legal cases for a second round. Upon inspection of the results, it was found that the many CRTs had merged into one line. The cause for the merge was due to keywords in the CRT not being catered for in the IE Protype. This resulted in the IE Protype processing multiple CRTs as one CRT. To resolve the issue, the list of keywords used by the IE Protype was extended to include the keywords that were not catered for. Legal case U10 would not be processed by the IE Protype as there were issues in parsing the document through the function that extracts the 'word/document.xml' file. Table 5-13 shows the difference ratio for each unseen legal case during the first round. It was observed that 30% (n=3) of unseen cases approximated a difference ratio of 1, 20% (n=2) of unseen legal cases approximated a difference ratio of 0.625. The remaining unseen legal cases all had different values for their difference ratios. Using Equation 5.6-2, a total difference ratio of 0.403 was observed. This indicates that 40% of observed unseen legal case CRTs extracted were different from what was expected.

Legal Case	Number of CRTs	CRTs Extracted	Difference	Difference Ratio
U1	36	26	10	0.28
U2	12	0	12	1.00
U3	28	5	23	0.82
U4	2	0	2	1.00
U5	20	21	1	0.05
U6	29	28	1	0.03
U7	16	6	10	0.63
U8	11	10	1	0.09
U9	8	3	5	0.63
U10	4	0	4	1

Table 5-13: Result of Round 1 Extraction for CRTs

During the second round, the unseen cases were parsed through the IE Protype again to determine if the results had improved. During the second round, there was a small improvement in the results. In particular, legal case U1's number of CRTs extracted increased from 26 to 33. However, the remaining unseen cases' number of extractions remained the same from round one. Table 5-14 summarises the results obtained from the second round of extracting CRTs.

Legal Case	Number of CRTs	CRTs Extracted	Difference	Difference Ratio
U1	36	33	3	0,08
U2	12	0	12	1,00
U3	28	5	23	0,82
U4	2	0	2	1,00
U5	20	21	1	0,05
U6	29	28	1	0,03
U7	16	6	10	0,63
U8	11	10	1	0,09
U9	8	3	5	0,63
U10	4	0	4	1

Table 5-14: Result of Round 2 Extraction for CRTs

Upon further inspection it was found that in addition to some more keywords not catered for, CRTs were once again merging with each other. When inspected, it was found that the merged CRTs were those who had no action taken on them. This resulted in the prototype being updated to cater for the different keywords and cater for CRTs that had no action. To ensure that CRTs with no action were extracted, an additional pre-processing step was required. During this step, additional regular expressions were created to identify and mark the CRTs that had no action. It was observed that 30% (n=3) unseen legal cases had a difference ratio of 1, 20% (n=2) of unseen legal cases had a difference ratio of 0.625. The remaining difference ratios were difference for the rest of the unseen legal cases. A total difference ratio of 0.361 was observed. This indicates that 36% of observed unseen legal case CRTs extracted were different from what was expected. A third round was then conducted on the unseen legal cases.

The results obtained from the third round indicated an improvement with the IE Prototype. All CRTs for each unseen legal case were extracted as seen in Table 5-15.

Legal Case	Number of CRTs	CRTs Extracted	Difference	Difference Ratio
U1	36	35	1	0.03
U2	12	12	0	0.00
U3	28	28	0	0.00
U4	2	0	2	1.00
U5	20	24	4	0.20
U6	29	33	4	0.14
U7	16	15	1	0.06
U8	11	12	1	0.09
U9	8	8	0	0
U10	4	0	4	1

Table 5-15: Result of Round 3 Extractions for CRTs

It was observed that 30% (n=3) of unseen legal cases had a difference of 0 while the remaining unseen legal cases had a range of difference ratios. A total difference ratio of 0.006 was observed. This indicates that 0.6% of observed unseen legal case CRTs extracted were different from what was expected.

When inspecting the results of the third round, it was found that some extractions exceeded the actual number of CRTs in a case. Examples of this is evident with unseen legal cases U5, U6, and U8. Upon further inspection it was found that the reason for the extra extractions was due to the layout of a particular phrase that is sometimes found in the CRT section of a legal case. This phrase contains formatting that is also used when writing legal citations and as a result was included by the IE Prototype when processing the CRT section. To resolve this issue, these phrases were identified and removed. Figure 5-9 illustrates the number of extractions for each unseen legal case for the three rounds.

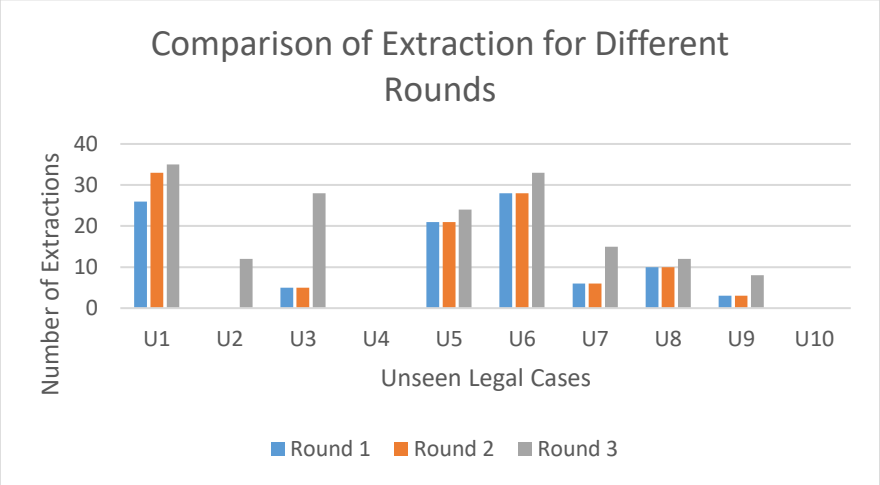


Figure 5-9: Comparison of Extraction for Different Rounds

5.7.2 Iteration 2

The second iteration of the IE Prototype consisted of two formative experiments which were conducted to setup and test the Database process of the IE Prototype. The first experiment investigated the setup of a graph database and the second experiment investigated the setup of a document database. Table 5-16 summarises the experiments conducted for iteration 2.

Experiment	Part	Test Documents	Result
1: Setup of Graph Database	-	-	Unsuccessful, nodes were being inserted as blank nodes with relationships
2: Setup of Document Database	Part 1	U1-U10	General data inserted correctly but CRTs not inserted correctly
	Part 2: Inspecting Code to Make Changes		Successful insertion of general data and CRTs

Table 5-16: Summary of Experiments Conducted for Iteration 2

5.7.2.1 Iteration 2: Experiment 1

Experiment 1 used the Python library provided by Neo4j and the Neo4j desktop application to locally create and connect to a graph database. During the first phase, dummy data was created and inserted into the graph database as nodes. The insertion of the data was successful. However, errors were encountered when trying to create the nodes with relationships. Blank nodes would be created with a label assigned to the relationship. Multiple attempts were made to resolve the error, but no solution could be found.

5.7.2.2 Iteration 2: Experiment 2

Experiment 2 consisted of two parts and used the Python library provided by MongoDB and the MongoDB Compass desktop application to locally create and connect to a document database. The data that was output from the extraction process was in the form of a list containing LegalCase objects. During part 1, the data to be inserted into the document database was converted to a Python dictionary to allow the data values to be associated with keys and allow for the keys and values to be searchable. Figure 5-10 illustrates how a LegalCase object is stored in document form.

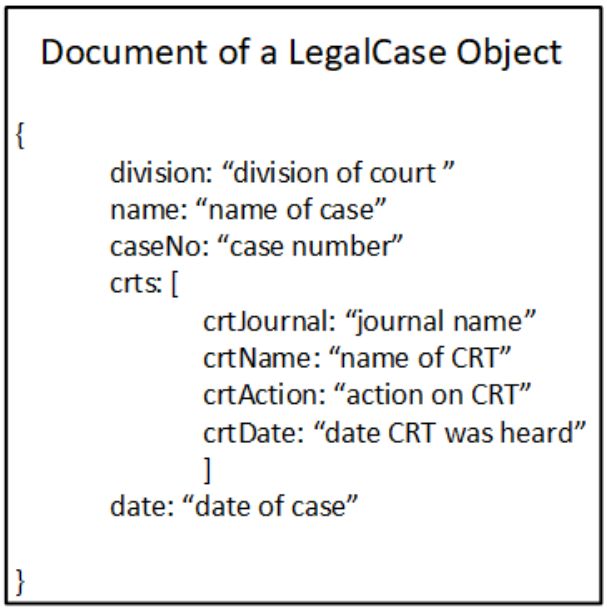


Figure 5-10: A Legal Case Stored as a Document

The data of a LegalCase object was inserted into the dictionary and an embedded approach was taken to insert the CRTs for a legal case. The CRTs were embedded into the document as a list containing Python dictionaries to store the CRTs. Initially, the data of a LegalCase object was successfully inserted into the document database. However, the CRTs were not inserted correctly as all CRTs extracted were being assigned to each LegalCase object. This issue was resolved in part 2 of the experiment. Upon

inspection during part 2, it was found that the update function applied to the list storing the CRTs was being applied to all instances where the list was used instead of the list being reset for a new LegalCase object. The update was changed to create a new empty list when a new LegalCase object was being processed. This resulted in the data being inserted correctly. Figure 5-11 illustrates how MongoDB created the schema to store the data. This is similar to the example illustrated in Figure 3-13.

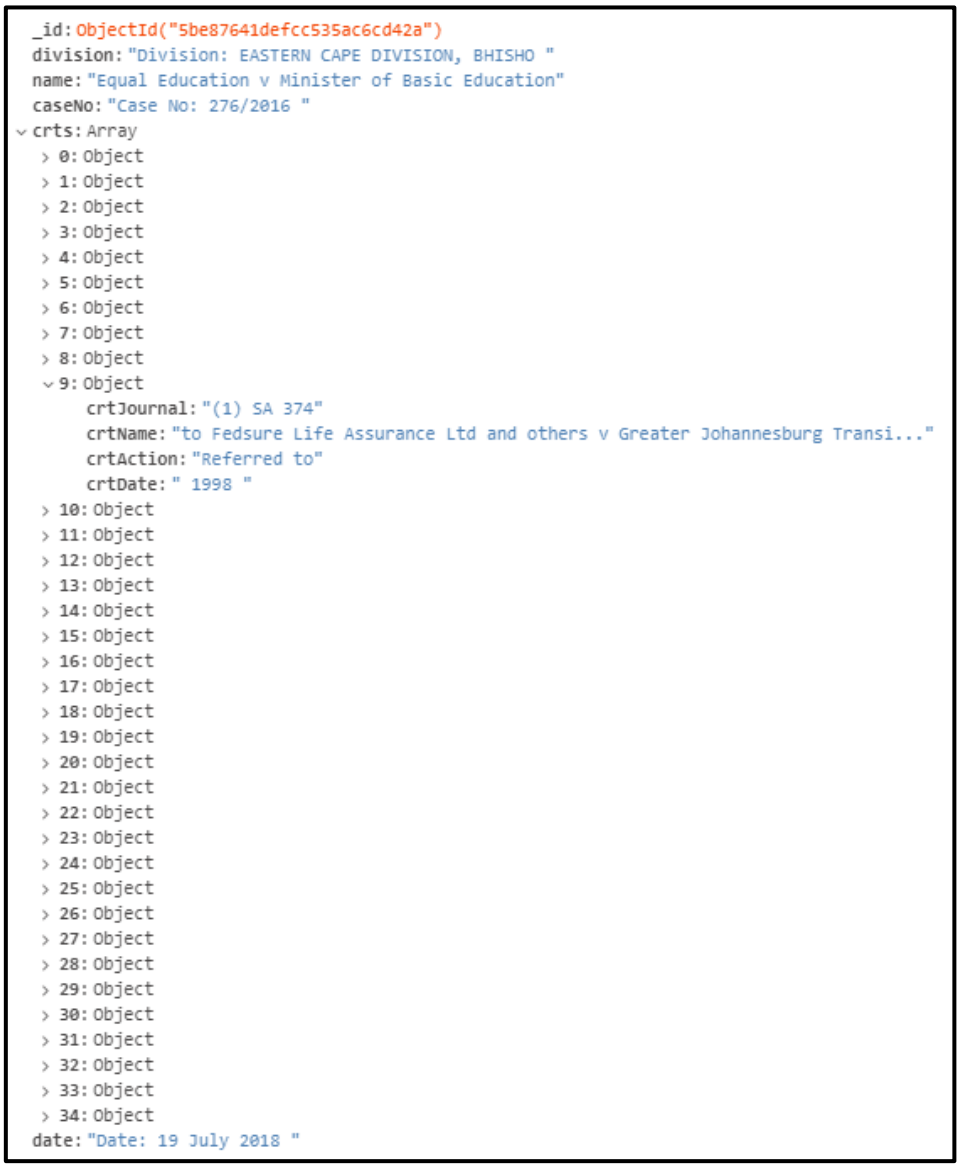


Figure 5-11: Screenshot of a Document in MongoDB

5.7.3 Analysis of Findings from Experiments

During the first iteration, five experiments were conducted that investigated web scraping and performing IE directly on documents in PDF and .docx formats. It was found that PDF documents can be inconsistently formatted resulting in poor extraction results. The use of a .docx formatted document is better as the paragraph tags that store text in XML can be accessed and processed. It was found that legal cases contained differently formatted text or sometimes had inconsistencies - highlighting problem 2 from Table 3-15 identified by Gurusamy and Kannan (2014). Facts could be extracted but the text had to be cleaned before ideal extraction could occur -highlighting problems 3 and 4 from Table 3-15 identified by Gurusamy and Kannan (2014). It was also found that some CRTs had no action taken on them. Separating a legal case into two parts allowed for primary attributes and

CRTs to be extracted. Facts were successfully extracted from the test documents (T1, T2, and T3) but three rounds were used to process and extract the unseen legal cases (U1 to U10). It was observed that U1 to U10 had a total difference ratio of 0.006 implying that 0.6% of the number of observed CRTs from unseen legal cases were different from what was expected. This is a good value as 0 is the ideal value to obtain. It was also observed that some CRTs had no actions and were in different formats.

During the second iteration implementation of the graph database was unsuccessful. Nodes were being inserted as blank nodes even though data was being parsed into them. No solution could be found for this issue which resulted in the implementation of the document base. Implementing the document database was successful. The document database was used to store LegalCase objects in the form of documents (Figure 5-10). A limitation of the experiments was that only MS Word .docx formatted documents could be processed by the IE Prototype.

5.8 Conclusions

This chapter reported on the third and fourth activities of the DSR methodology namely, Design and Development, and Demonstration. The FEDS Framework and Technical Risk and Efficacy strategy were used to develop an evaluation plan for the proposed MAC Model. Incremental prototyping allowed for an IE Prototype to be created through iterations. The IE Prototype consisted of two processes namely, the IE process and the database process. Technical issues and security protocols prevented the use of web scraping to access and extract legal cases online. Different options of performing IE directly on the legal cases were investigated during the experiments. The inconsistent formatting of PDF documents prevented ideal IE and as such, MS Word .docx formatted documents were used. The IE Prototype used a combination of regular expressions and NLP techniques such as tokenisation and stop-word removal to process and extract facts from legal cases. Graph and document databases were investigated as options to represent the Information Storage process of the proposed MAC Model. Technical issues with the Neo4j graph database vendor prevented complete implementation of the graph database. However, the document database was successfully implemented and used to store the processed legal cases. The design and development of the IE Prototype partially addressed the main research objective:

- Ro_M : To develop an information extraction model to recommend the Most Applied Case for a field of law.

In completing this chapter, three deliverables were produced namely, an evaluation plan (Table 5-6), a developed and evaluated prototype, and a solution to the problem in the form of two artefacts. The two artefacts are results of the MAC Model (Section 5.2) and IE Prototype. Table 5-17 summarises the processes of the MAC Model. The following chapter will report on the evaluations conducted.

MAC Model Processes	Literature	Figure	Table
IR	Section 3.2 Section 3.3	Figure 3-3	Table 3-2
IE	Section 3.4 Section 3.5 Section 3.6 Section 3.7	Figure 3-5	Table 3-3
Information Storage	Section 3.8	Figure 5-3 Figure 5-10	Table 3-11
Query-Independent Ranking	Section 3.9	-	Table 3-12

Table 5-17: Summary of the MAC Model Linked to Literature, Figures and Tables

Chapter 6: Analysis of Evaluation Results

6.1 Introduction

The previous chapter addressed the third and fourth activities of the DSR methodology namely, Design and Development, and Demonstration of the artefacts. The previous chapter also presented an evaluation plan to evaluate the proposed IE model and MAC System. This chapter reports on the fifth DSR activity namely, Evaluation (Figure 6-1). An evaluation of the IE Prototype was setup using 50 legal cases (Section 6.2) followed by an evaluation of the IE Prototype's scalability (Section 6.3). The research objective addressed in this chapter is:

- **RO4**: *Identify the criteria that can be used to evaluate the proposed model.*

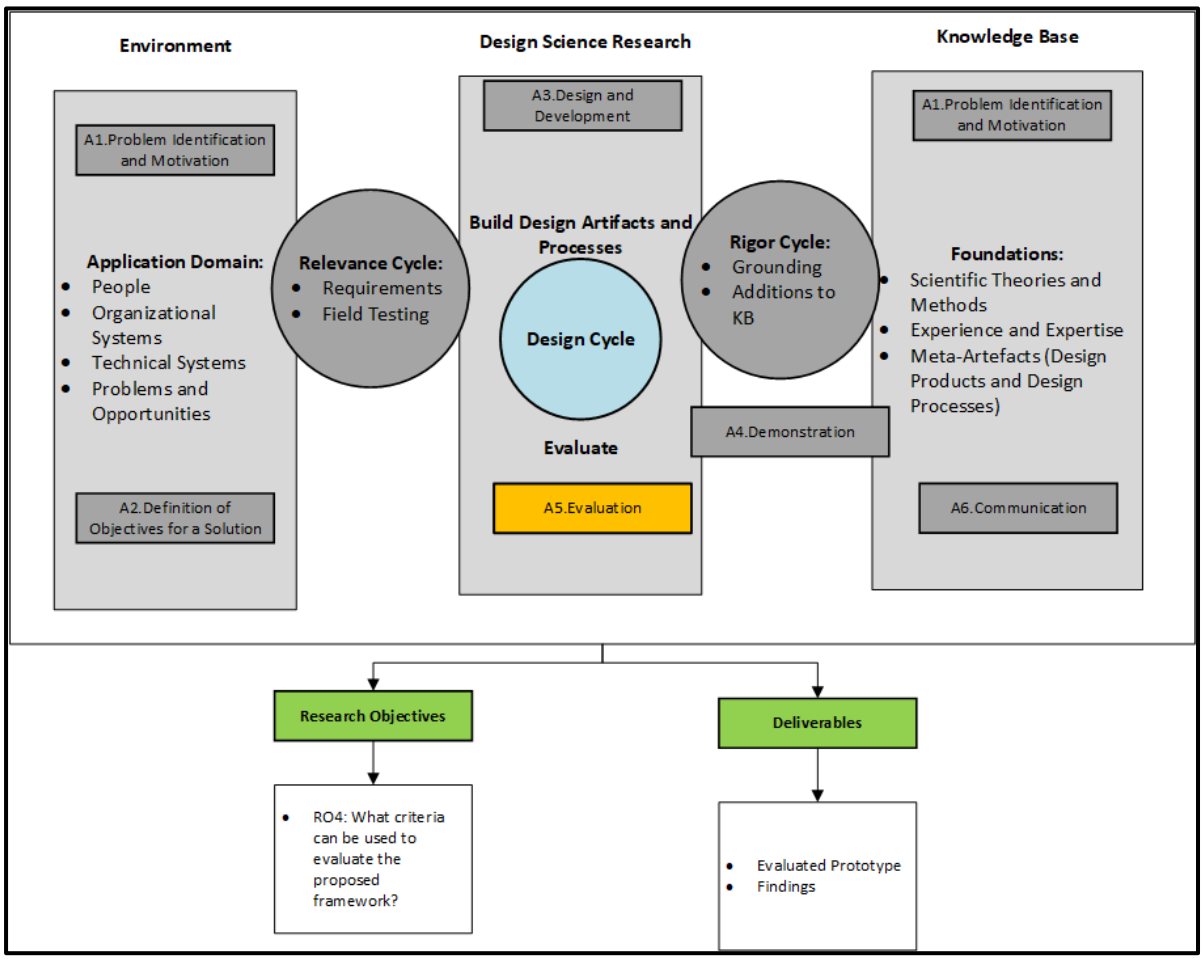


Figure 6-1: Chapter 6 DSR Activities

6.2 IE Prototype Evaluation using 50 Legal Cases

To evaluate the IE Prototype, legal cases had to be processed by means of a summative evaluation. For the evaluation, 50 legal cases were deemed sufficient for testing the proof-of-concept. The legal cases were obtained through the process provided in Figure 5-7 and named as F1 to F50. Therefore, the 50 legal cases were selected based on the legal cases being case law for the periods 1996 to 2018, for all fields of law within the ALL SA legal journals. For the purpose of this evaluation, a legal case was referred to as a 'source case'. The aim of the test was to determine the IE Prototype's effectiveness, scalability, and execution time.

6.2.1 Procedure

The steps that were followed to conduct the evaluation were as follows:

1. Download and format source cases;
2. Establish a connection to the database;
3. Parse the 50 .docx formatted source cases into the IE Protype; and
4. Analyse the results obtained.

Establishing a database connection required an instance of MongoDB to be created and connected to the MongoDB Compass desktop application. To analyse the results, categories were required to classify each source case as a whole and the CRTs extracted from the source case. The categories used to classify a source case were:

- Perfectly extracted – this means that the case was perfectly extracted;
- Partially extracted - this means that the case was partially extracted; and
- Not - this means that the case was not extracted at all.

A source case was assigned one of the above categories based on whether all cases, some of the cases, or none of the cases were extracted. This means that if source case A had five CRTs of which three CRTs were perfectly extracted and two were partially, then source case A was categorised as partially extracted.

Similarly, the categories used to classify a CRT were:

- Perfectly extracted – this means that all four attributes were extracted;
- Partially extracted – this means that only 1 to 3 attributes were extracted; and
- Zero – this means that zero attributes were extracted.

For each partial or no extraction further investigation to the possible reason were conducted, and the resulting reasons were grouped into themes of similar occurring reasons.

6.2.2 General Results

It was observed that from the 50 source cases (F1 to F50), 697 CRTs were expected to be extracted. However, the actual number of CRTs extracted was 731. The reason for having an additional 34 CRTs was due to instances of extra lines, splitting of CRTs, and noise.

From the analysis, three additional categories to Perfect, Partial, and Not were discovered namely, extra lines, splits, and noise. Instances of extra lines of text that were similarly formatted to CRTs were detected and extracted as a CRT. There were instances during the extraction where a CRT would be split into two parts, resulting in an additional CRT being extracted. It was also observed that there were instances of phrases in the source cases that were formatted like CRTs and this resulted in noisy facts being extracted. Noisy facts refer to phrases that were formatted like CRTs that were extracted. Table 6-1 summarises the expected number of CRTs with the actual number of CRTs extracted along with the difference ratios calculated for the 50 source cases.

Case	Nr CRTs	Nr CRTs Extracted	Absolute Difference	Difference Ratio
F1	45	46	1	0,02
F2	5	5	0	0
F3	42	43	1	0,02
F4	5	4	1	0,2
F5	6	6	0	0
F6	2	2	0	0
F7	15	11	4	0,27
F8	7	8	1	0,14
F9	5	5	0	0
F10	1	2	1	1
F11	7	8	1	0,14
F12	3	3	0	0
F13	15	18	3	0,2
F14	9	10	1	0,11
F15	33	33	0	0
F16	5	4	1	0,2
F17	6	8	2	0,33
F18	23	23	0	0
F19	8	4	4	0,5
F20	10	25	15	1,5
F21	10	10	0	0
F22	11	27	16	1,45
F23	23	24	1	0,04
F24	10	19	9	0,9
F25	23	19	4	0,17
F26	26	25	1	0,04
F27	12	13	1	0,08
F28	6	6	0	0
F29	15	15	0	0
F30	15	19	4	0,27
F31	18	19	1	0,06
F32	11	12	1	0,09
F33	6	6	0	0
F34	7	9	2	0,29
F35	10	11	1	0,1
F36	17	17	0	0
F37	27	21	6	0,22
F38	10	10	0	0
F39	75	60	15	0,2
F40	5	5	0	0
F41	6	6	0	0
F42	8	8	0	0
F43	13	13	0	0
F44	13	13	0	0
F45	12	11	1	0,08
F46	6	5	1	0,17
F47	15	17	2	0,13
F48	19	23	4	0,21
F49	13	17	4	0,31
F50	3	3	0	0
Totals	697	731	110	

Table 6-1: Difference Ratios for 50 Test Cases

Of the 50 source cases, it was found that 96% (n=48) of the source cases were categorised at partially extracted and 4% (n=2) were categorised as noisy cases. It was found that 19 partially extracted cases resulted in 19 instances of a case being split. Table 6-2 summarises the source cases that that were categorised.

Nr Perfects	0
Nr Partials	48
Nr Not Extracted	0
Nr Extra Lines	0
Nr Splits	0
Nr Noise	2
Total	**50**

Table 6-2: Number of Source Cases Categorised

Analysis of the attributes of CRTs extracted was also conducted. Appendix J provides a detailed table of the results of the CRTs that were extracted. The four attributes that are found in a CRT namely, title, date, journal, and action were analysed and classified based on the description in Section 6.2.1. Table 6-3 provides a summary of all 731 CRTs attributes extracted.

Attribute	Category	Frequency
CRT Title	Perfectly extracted CRT Titles	196
	Partially extracted CRT Titles	353
	Titles not extracted	99
	Extra instances	25
	Split instances	19
	Noise instances	39
Expected Nr of CRT Title Extractions		**731**
Actual Nr of CRT Title Extractions		**731**
CRT Date	Perfectly extracted CRT Dates	604
	Partially extracted CRT Dates	0
	CRT Dates not extracted	44
	Extra instances	25
	Split instances	19
	Noise instances	39
Expected Nr of CRT Date Extractions		**731**
Actual Nr of CRT Date Extractions		**731**
CRT Journal	Perfectly extracted CRT Journals	531
	Partially extracted CRT Journals	4
	CRT Journals not extracted	113
	Extra instances	25
	Split instances	19
	Noise instances	39
Expected Nr of CRT Journals Extractions		**731**
Actual Nr of CRT Journals Extractions		**731**
CRT Action	Perfectly extracted CRT Actions	292
	Partially extracted CRT Actions	3
	CRT Actions not extracted	353
	Extra instances	25
	Split instances	19
	Noise instances	39
Expected Nr of CRT Actions Extractions		**731**
Actual Nr of CRT Actions Extractions		**731**

Table 6-3: Summary of CRT Attributes Extracted

It was found that many of the titles contained an extra word in it, therefore preventing the title from being perfectly extracted. As a result, 48% (n=353) of CRT titles were partially extracted while 26% (n=196) were perfectly extracted. For CRT dates it was observed that 82% (n=604) of the dates were perfectly extracted while 6% (n=44) of the dates were not extracted. For CRT journals it was observed that 72% (n=531) of the journals were perfectly extracted while 15% (n=113) of the journals were not extracted. During the analysis, two discoveries were made. The first discovery was that there were instances of CRTs merging with other CRTs in a source case. This resulted in 11 CRTs merging into six partially extracted CRTs. The second discovery was that a total of eight CRTs were not recognised by the IE Prototype and as a result not extracted. These two discoveries possibly contributed to the low extraction of the facts, particularly the CRT actions. It was observed that 48% (n=352) of the actions were not extracted while 40% (n=292) of the actions were perfectly extracted.

Based on the guide provided by Conroy (2016) a 10% margin of error will be applied to the results.

6.2.3 Effectiveness Results

Effectiveness was measured by determining the accuracy of the IE Prototype. Equation 5.6-1 and Equation 5.6-2 were used to determine the accuracy. Table 6-4 summarises the ranges of difference ratios calculated for the 50 source cases.

Difference Ratio Range	Count of Instances	Percentage
0's	19	38
1	1	2
0,01 to 0,09	7	14
0,1 to 0,5	19	38
0,6 to 0,9	1	2
1 to 1,5	3	6

Table 6-4: Summary of Difference Ratio Ranges for 50 Source Cases

It was observed that 38% (n=19) of the source cases had a difference ratio of 0. Another 38% (n=19) of source cases had a difference ratio between 0.1 to 0.5. A total difference ratio of 0,157 was observed for all source cases. This indicates that 16% of the number of source cases extracted were found to be different from what was expected. Using a 10% margin of error results in 56% (n=28) of the source cases falling within the margin of error and 44% (n=22) of the source cases falling outside of the margin of error.

Equation 5.6-1 and Equation 5.6-2 were also applied to the perfectly extracted CRT attributes. Table 6-5 summarises the difference ratios for the perfectly extracted attributes.

Attribute	Frequency	Difference ratio
Titles	196	0.73
Number of CRT Dates	604	0.17
Number of CRT Journals	531	0.27
Number of CRT	292	0.60

Table 6-5: Difference Ratios for Perfectly Extracted Attributes

For CRT titles, a difference ratio of 0.731 was observed. Indicating that 73% of the number of the extracted CRT titles were different from what was expected.

For CRT dates, a difference ratio of 0.173 was observed. Indicating that 17% of the number of the extracted CRT dates were different from what was expected.

For CRT journals, a difference ratio of 0.273 was observed. Indicating that 27% of the number of the extracted CRT journals were different from what was expected.

For CRT actions, a difference ratio of 0.600 was observed. Indicating that 60% of CRT actions extracted were different from what was expected.

Considering all the perfectly extracted properties resulted in a total difference ratio of 0.444. Indicating that 44% of the number of extractions observed were different from what was expected. In addition, it was found that 3.4% (n=25) of the CRTs analysed contained extra lines, 2.6% (n=19) contained CRTs that were split into separate lines, and 5.3% (n=39) contained noisy information.

From the results obtained, it is seen that although 56% of CRTs extracted fall within the 10% margin of error, the accuracy of the attributes of the CRTs extracted can be improved. Factors contributing to a poor accuracy include presence of the extra lines, noisy data, and the presence of different formatted CRTs.

6.3 Scalability and Execution Time Evaluation

An evaluation was conducted to measure the IE Prototype's scalability. The scalability of the IE Prototype was evaluated in conjunction with measuring execution times as discussed in Section 5.6.3 to determine how well the IE Prototype could process an increasing amount of source cases. Scalability is important because in a real-world setting an unlimited amount of source cases could be parsed through the prototype and processing should occur quickly. Evaluating the scalability required both parts of the IE Prototype to be tested namely, the IE process and Database process.

6.3.1 Procedure

To test the scalability of the IE Prototype, the following procedure was followed:

1. Obtain source cases (Figure 5-7);
2. Organise source cases into batches;
3. Connect to the document database;
4. Parse source cases into the IE Prototype; and
5. Record extraction and insertion times.

The process illustrated in Figure 5-7 was followed, source cases were downloaded, converted, and parsed through the IE Prototype. A total of 102 source cases were used and organised into batches of 10 source cases. To measure the amount of time taken to insert the LegalCase objects into the database, a connection to the document database had to be established. The connection was made before the IE process could begin to avoid potentially wasting time after the IE process completed. The source cases were then parsed into the IE Prototype in increasing batches of 10 source cases at a the time. Two sets of recordings were taken. The first set of recordings were for time taken to perform the IE process on the source cases and create LegalCase objects. The second set of recordings was for time taken to insert the LegalCase objects as key-value pairs into the document database.

6.3.2 Scalability and Execution Time Results

Execution time refers to the total time taken to extract source cases and insert the source cases into the database. Table 6-6 summarises the time taken by the IE Prototype to extract the source cases and create LegalCase objects. The LegalCase objects were created after the required facts of a source case were extracted. It was observed that extraction time increased when the amount of source cases increased.

Batch	Nr Files	Time (seconds)
1	12	0.3
2	22	0.6
3	32	1.2
4	42	8.9
5	52	9.3
6	62	9.7
7	72	9.9
8	82	12.1
9	92	17.3
10	102	19.1

Table 6-6: Time taken to Extract the Source Cases

It was observed that for the 10 batches totalling 102 files, an average of 8.8 seconds was taken to extract the source cases. Since each source case does not have a set word limit it can result in a source case containing either few or many pages of information. This can cause extraction times to vary and in the instance of a source case having many words it can mean more processing is needed to locate the required facts from the source case. Another reason for an increasing time could be the lack of CPU processing power on the researcher's machine. Newer CPUs should be more advanced and can process information much faster than the CPU used on the researcher's machine.

Table 6-7 summarises the time taken to insert the LegalCase objects as key-value pairs into the document database. It was observed that the insertion time varied for populating the document database. Insertions into the document database were quick with an average of 0.10 seconds but no pattern could be found for the insertion times.

Nr LegalCase Objects	DB Insertion Time (seconds)
12	0.01
22	0.08
32	0.02
42	0.03
52	0.07
62	0.36
72	0.05
82	0.11
92	0.19
102	0.08

Table 6-7: Time taken to Insert Legal Case objects

The insertion times did not consecutively increase as each batch of source cases increased. It was observed that the longest time taken to insert LegalCase objects was 0.36 seconds for 62 Legal Case objects whilst the shortest time taken was 0.01 seconds for 12 LegalCase objects. The quick and unpredictable time is possibly due to the scaling method used by MongoDB which allows for large sets of data to be easily catered for. It is possible that the researcher's machine could have had a small impact on the insertion results. However, the results in Table 6-7 reveal that it would have been a negligible impact.

The processing time for extraction was satisfactory as the IE Prototype performed efficiently until 32 source cases were processed. It can be concluded that although the IE Prototype can process 32 source

documents in 1.2 seconds, processing should be done on a more powerful machine to ensure quick results. Especially since real-world settings could require more than 32 source documents to be processed. The document database's performance was also satisfactory well when inserting the batches of LegalCase objects. However, implementation should be performed on an external machine to ensure that all resources are allocated to the insertion process.

6.4 Conclusions

This chapter reported on the fifth activity of DSR namely, Evaluation. The IE process of the theoretical MAC Model was implemented in the form of the practical artefact, the IE Prototype. The IE Prototype was evaluated in terms of its effectiveness and scalability. The evaluation of the IE Prototype addressed the following research objective:

RO4: *Identify the criteria that can be used to evaluate the proposed model.*

The relevant criteria identified were effectiveness, accuracy, and scalability. For this reason the IE prototype was evaluated using these criteria. Based on the results from the evaluation, the majority of source cases extracted were categorised as partial extractions. Half of the source cases had difference ratios of 0 while the other half had difference ratios in the range 0.1 to 0.5. The results indicate that the accuracy of the IE Prototype needs to be improved. Changes to the IE Prototype that cater for more formats of CRTs could result in more source cases and CRTs being perfectly extracted. The IE Prototype performed satisfactory in terms of scalability with an average processing time of 8.87 seconds for 102 cases. Table 6-8 summarises all the results obtained from the experiments.

Results	**Source Cases Categories**	**Source Cases Categorised (n=50)**	**CRT Attribute**	**CRT Perfect Extractions (n=731)**
General Results	Perfect Extracted	0	CRT Titles	196
	Partially Extracted	48	CRT Dates	604
	Not Extracted	0	CRT Journals	531
	Nr Extra Lines	0	CRT Actions	292
	Nr Splits	0		
	Nr Noise	2		
Effectiveness	**Difference Ratio Range**	**Count of Instances (n=50)**	**Percentage**	**Total difference ratio of source cases**
	0	19	38	0,157819225
	1	1	2	
	0,01 - 0,09	7	14	
	0,1 - 0,5	19	38	
	0,6 - 0,9	1	2	
	1 - 1,5	3	6	
Scalability	**Average IE processing time (s)**	**Average insertion time (s)**	**Longest insertion time (s)**	**Shortest insertion time (s)**
	8.87	0.10	0.361	0.009

Table 6-8: Summary of Experiment Results

In completing this chapter, two deliverables were achieved. The first deliverable is the evaluated prototype, and the second deliverable is the findings from the evaluations (Table 6-8). The next chapter will conclude this research.

Chapter 7: Reflection, Conclusions, and Future Work

7.1 Introduction

The previous chapter addressed the fifth DSR activity of the DSR methodology namely, Evaluation. The MAC Model and the MAC System were evaluated, and the results were discussed. This chapter concludes the research by reviewing the ROs to determine whether the research is successful. This chapter will report on the sixth DSR activity of the DSR methodology namely, Communication (Figure 7-1). The RQ of this research was:

> ***RQ**: What techniques can be incorporated into a model that recommends the Most Applied Case (MAC) for a field of law?*
>
> ***RQ-Context**: What text processing techniques can be used to process legal cases at LexisNexis?*

The main research objective ($\mathbf{RO_M}$) of this research was:

> *To develop an information extraction model to recommend the Most Applied Case for a field of law.*

The DSR methodology was used throughout this research in the development of the theoretical and practical artefacts being the MAC Model and the IE Prototype. The fulfilment of the ROs through the DSR activities will be reported on followed by the theoretical and practical contributions of this research. Throughout the research various problems and limitations were experienced that indicated potential future research.

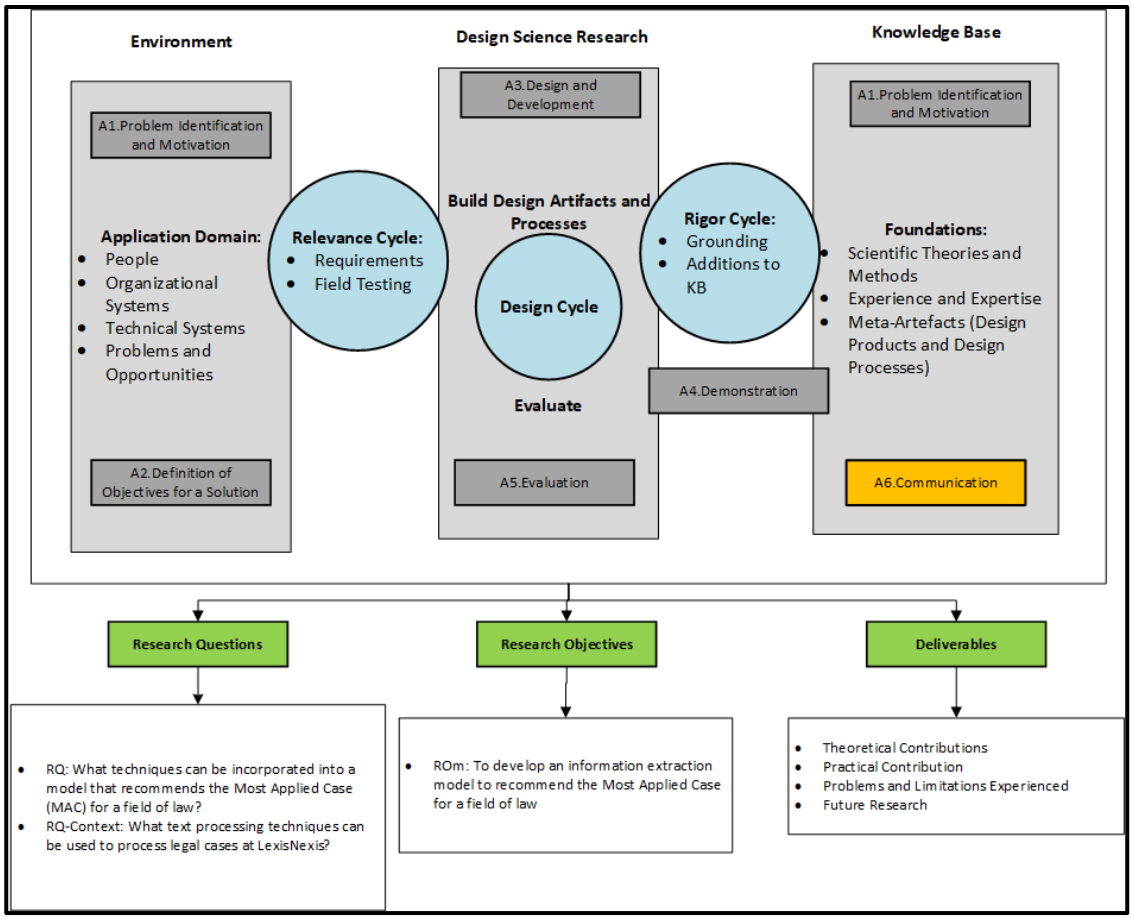


Figure 7-1: Chapter 7 DSR Activities

7.2 Fulfilment of Research Objectives

This research revealed that a combination of techniques can be used to create a prescriptive MAC Model to recommend the MAC for a field of law. Prescriptive models provide descriptions of possible future solutions and aid in constructing artefacts (Johannesson & Perjons, 2012). The MAC Model was designed based on the informal process provided by legal researchers at LexisNexis to recommend the MAC. The MAC Model consisted of four processes that used the techniques investigated in the literature review. The main research objective for this research was:

To develop an information extraction model to recommend the Most Applied Case for a field of law.

To fulfil the main research objective the following sub-objectives were derived and addressed:

RO1: *Identify the problems experienced when processing text as identified by literature and within a real-world context.*
RO2: *Identify the attributes of a court case that can be used to aid in recommending the MAC.*
RO3: *Determine what techniques and algorithms can be used to recommend the MAC.*
RO4: *Identify the criteria that can be used to evaluate the proposed model.*

The first research objective, RO1, was to **identify the problems experienced when processing text as identified by literature and within a real-world context.** The identification of problems helped in defining the research problem of recommending the MAC (Table 4-4). The problems were identified by means of the sources consulted in the literature review, and include ambiguity, special formats of text, abbreviations, and acronyms. The absence of pre-processing tasks can also make processing text difficult. These problems were experienced during the development of the IE Prototype during which pre-processing tasks were required, and special formats and abbreviations were continuously encountered. In terms of processing text, no IE techniques were used at LexisNexis.

The second research objective, RO2, was to **identify the attributes of a court case that can be used to aid in recommending the MAC***.* The informal manual process used to recommend the MAC aided in identifying the attributes required (Table 4-2). Consulting with the experts and analysing the legal cases also contributed to identifying the attributes. The result of the informal process, consultation, and analysis was a set of attributes that were to be extracted to create a LegalCase object.

The third research objective, RO3, was to **determine what techniques and algorithms can be used to recommend the MAC.** The informal manual process used to recommend the MAC helped guide the researcher on the type of techniques to investigate. A critical analysis of these techniques was investigated in the literature review. The techniques required to recommend the MAC would be a combination of IR, IE, information storage, and query-independent ranking. Five IR models were analysed and compared (Table 3-1 and Table 3-2) to determine which would be suitable to use in an IE model. Four categories of IE techniques were investigated namely, general IE techniques, web scraping, NLP, and regular expressions (Table 3-3). The use of the IE techniques depended on the end-goal of the user. To recommend the MAC, experiments investigating web scraping and regular expressions were conducted. For information storage, two types of NoSQL databases were analysed namely, graph and document databases (Table 3-11). Being NoSQL databases meant these two types of databases were schema-free and allowed for flexibility and scalability.

The fourth research objective, RO4, was to **identify the criteria that can be used to evaluate the proposed model.** The MAC Model was created, and a prototype of the IE process was implemented in iterations. Based on the functionality that the model had to perform, it was decided that the prototype should be effective, accurate, and scalable. The prototype had to extract facts as accurately as possible

and do so quickly even if many legal cases were parsed. The prototype was then evaluated based on the set of criteria. Table 7-1 summarises the reflection of the research based on the DSR guidelines (Table 2-1).

Guideline	Reflection
Guideline 1: Problem Relevance	Finding the MAC is a long manual process with no systems or automated processes in place. The MAC Model represents an automation of the manual process and can reduce legal researchers' research times.
Guideline 2: Research Rigor	Methods must be applied to construction and evaluation of the artefact being designed. A literature review, extant systems analysis, consultation with experts, incremental prototyping and experiments were used to construct and evaluate the MAC Model and its IE Prototype.
Guideline 3: Design as a Search Process	Designing an artefact must be an iterative process. All options should be used until a final, accepted artefact is achieved.
Guideline 4: Design as an Artefact	A prescriptive model called the MAC Model was the artefact created to recommend the MAC.
Guideline 5: Design Evaluation	An evaluation plan consisting of strategies and methods was designed to iteratively evaluate the prototype produced from the artefact.
Guideline 6: Research Contributions	Theoretical and practical contributions were made. Theoretical contributions include amongst others, the set of problems faced in the legal domain and challenges in processing text in the legal domain, and an IE model to recommend the MAC. The practical contribution consisted of the IE Prototype of the MAC Model that implemented the IE and Database Processes (Section 7.3).

Table 7-1: Reflection of the Research and DSR Guidelines

7.3 Research Contributions

The research contributions for this research were categorised as theoretical contributions and practical contributions. This section will report on both contributions. The MAC Model is both a theoretical and practical contribution. The theoretical contribution of the MAC Model consists of the theory that researchers can use while the practical contribution is for legal experts who work in the industry, and can apply the MAC Model to help automate and implement such processes.

7.3.1 Theoretical Contributions

The theoretical contributions of this research are:

- D1-Problems and challenges of processing text in the legal domain;
- D2- Attributes of a court case to recommend the MAC;
- D3- A MAC Model to recommend the MAC, consisting of:
    - IR techniques;
    - IE techniques;
    - Info Storage techniques; and
    - Query-independent ranking algorithms.

Problems included large amounts of time spent on searching for useful information, no formal processes to find the MAC, the lack of IE techniques to facilitate finding information, and the lack of query-independent ranking algorithms. Processing text in the legal domain was a challenge, particularly regarding the CRTs of a legal case. Upon inspection, it was found that CRTs can have

multiple formats, and in some instances no format was followed to represent the CRT. This required IE techniques to cater for multiple instances.

The informal process to recommend the MAC and analysis of the legal cases revealed that specific attributes can be extracted to aid in recommending the MAC. The attributes were a combination of general data and all CRTs found in a legal case. From the general data, the important attributes were the legal case's division and case number. As discussed in Section 4.3.2 the division can influence the supremacy a legal case has while the case number can be used to uniquely identify the legal case. From the CRTs, the title, year, journal, and action were extracted. The action determined how valuable the CRT would be in a defence case.

Based on the literature review and findings from the experts at LexisNexis, a set of requirements were derived for a prescriptive model to recommend the MAC (Table 4-5 and Table 4-6). This model was called the MAC Model (Figure 5-2). The MAC Model consisted of four processes and used multiple techniques derived from the informal manual process to recommend the MAC and a literature review. The analysis of the literature revealed that IR, IE, information storage, and query-independent ranking could be integrated to recommend the MAC for a field of law. A vector space model could be used to support the retrieval of legal cases. Regular expressions were used in conjunction with NLP techniques to perform the IE (Table 3-3). Two NoSQL databases namely, graph and document databases were investigated as options for information storage (Table 3-11). The ability to create relationships between data without the need for primary or foreign keys made graph databases ideal. However, technical issues with the graph database vendor prevented successful implementation and resulted in the implementation of a document database. An adaption of a query-independent ranking algorithm would be suitable for performing the ranking on legal cases to return the MAC. During the literature review, four types of query-independent ranking algorithms were reviewed (Table 3-12).

7.3.2 Practical Contributions

The practical contributions of this research are the MAC Model, the IE Prototype that consisted of the IE and Database process, and the three-tier architecture for the MAC System. The MAC Model integrates the concepts investigated in the literature review into a three-tier architecture. Concepts integrated were IR, IE, information storage, and query-independent ranking. As part of the MAC Model, an IE Prototype was implemented that processed and stored legal cases as LegalCase objects. The IE Prototype was developed through multiple iterations of experiments and evaluated. The results of the evaluation present opportunities for future researchers to explore.

The three-tier architecture of the MAC Model demonstrated how the model could be locally implemented without the use of any external servers and also mapped to the LexisNexis' architecture. For testing purposes, a local setup was enough but for commercial purposes the server-side implementation should be implemented externally.

7.4 Problems Experienced and Limitations of Study

Three different problems were experienced throughout the research. The first problem encountered was the security protocols on the Mylexisnexis.co.za website. These security protocols prevented the use of web scraping. The second problem encountered was the inconsistent formatting of PDF documents. The inconsistency prevented PDF formatted legal cases from being completely processed during the experiments. The third problem encountered was technical issues with the graph database vendor used. This prevented the successful implementation of a graph database.

Five limitations were experienced during the research. The MAC Model was not evaluated to determine how well it met its requirements. The MAC Model's architecture was not tested to determine how well it fit into LexisNexis' technical architecture. The IE Prototype only processes legal

cases that are in .docx format. LexisNexis was unable to provide the legal cases in .docx format which resulted in the researcher having to convert legal cases from PDF format to .docx format. The focus of the research was on creating the MAC Model and implementing the IE process. Therefore, not all processes of the model were implemented.

7.5 Future Research

The results from the evaluation of the IE Prototype provide areas for future work. The IE Prototype can be adapted to cater for different formats of legal cases, not only .docx formats. This can be useful as not all legal cases would be formatted to a particular document type. An adaption and implementation of a query-independent ranking algorithm can be explored to perform the ranking of legal cases. Lastly, The IE Prototype can be extended to include machine learning to perform the IE process. This can result in an IE Prototype that learns to recognise different formats of legal citations and brings about better accuracy for extraction. This would require a training set of legal cases to be created and used on a machine learning model.

7.6 Summary

This research produced two solution artefacts based on the DSR methodology and DSR guidelines (Table 2-1) namely:

- The MAC Model (theoretical artefact); and
- The IE Prototype (practical artefact).

The MAC Model was developed as a prescriptive model to provide a possible solution to the problem of recommending the MAC for a field of law. The MAC Model uses three-tier architecture and consists of four processes to recommend the MAC. The techniques recommended to implement the MAC Model are the Vector Space Model for IR, a combination of regular expressions and NLP for IE, a graph or document database for Information Storage, and the PageRank algorithm for query-independent ranking.

The IE Prototype was developed as a proof-of-concept implementation of the MAC Model's IE process. The IE Prototype was evaluated to determine how effective, accurate, and scalable it was. The results of the evaluation revealed that 96% of the extractions from the IE Prototype were classified as partial extractions. Improvements should be made to the IE process to increase the effectiveness so that there are only perfect extractions. The IE Prototype was able to process up to 102 legal cases at a given time which was deemed satisfactory considering that documents can contain any amount of text.

--The end--

References

Abdelmagid, M., Ahmed, A., & Himmat, M. (2015). Information Extraction Methods and Extraction Techniques in the Chemical Document's Contents: Survey. *ARPN Journal of Engineering and Applied Sciences*, *10*(3), 1068–1073. Retrieved from www.arpnjournals.com/jeas/research_papers/rp_2015/jeas_0215_1562.pdf

Al-Anzi, F. S., & AbuZeina, D. (2018). Beyond Vector Space Model for Hierarchical Arabic Text Classification: A Markov Chain Approach. *Information Processing and Management*, *54*(1), 105–115. http://doi.org/10.1016/j.ipm.2017.10.003

Barrett-Grant, K., & Heywood, M. (2003). Introduction to the Legal System. In K. Barrett-Grant, D. Fine, M. Heywood, & A. Strode (Eds.), *HIV/AIDS and the Law: A Resource Manual* (3rd ed., pp. 45–62). Cape Town, South Africa: AIDS Law Project and AIDS Legal Network. Retrieved from https://section27.org.za/wp-content/uploads/2010/04/03Manual.pdf

Beagle Inc. (2018). Beagle: We sniff out the fine print so you don't have to. Retrieved April 10, 2018, from https://www.beagle.ai/

Binkley, D., & Lawrie, D. (2009). Information retrieval applications in software maintenance and evolution. *Encyclopedia of Software Engineering*, 10. http://doi.org/10.1081/E-ESE-120044704

Bird, S., & Loper, E. (2002). NLTK: The Natural Language Toolkit. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics* (pp. 1–4). Philadelphia, United States of America. http://doi.org/10.3115/1118108.1118117

Black, H. C., Nolan, J. R., Nolan-Haley, J. M., Hicks, S. C., & Brandi, M. N. (1990). *Black's Law Dictionary*. (B. A. Garner, Ed.) (6th ed.). Saint Paul: West Publishing Co.

Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. *Proceedings of the Second International Workshop on Software and Performance - WOSP '00*, 195–203. http://doi.org/10.1145/350391.350432

Buschmann, F., Maunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture Volume 1: A System of Patterns* (1st ed.). New York, United States: John Wiley & Sons.

Carr, M., & Verner, J. (1997). Prototyping and Software Development Approaches. Retrieved from https://www.google.com.my/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCkQFjAA&url=http://www.cb.cityu.edu.hk/is/getFileWorkingPaper.cfm?id=55&ei=73eDU6aCBo7PlAXNooGQBg&usg=AFQjCNFCEFbDyv9tNk_YuH0VpPfavJPs2A&sig2=wimyHPVpHpp

Ceri, S., Bozzon, A., Brambilla, M., Della Valle, E., Fraternali, P., & Quarteroni, S. (2013). The Information Retrieval Process. In *Web Information Retrieval* (pp. 13–27). Berlin, Germany: Springer-Verlag Berlin Heidelberg. http://doi.org/10.1007/978-3-642-39314-3

Chen, S., Gulati, S., Hamid, S., Huang, X., Luo, L., Morisseau-Leroy, N., … Zhang, C. (2003). A Three-tier System Architecture Design and Development for Hurricane Occurrence Simulation. In *Proceedings, ITRE 2003 - International Conference on Information Technology: Research and Education* (pp. 113–117). Newark, New Jersey, United States of America. http://doi.org/10.1109/ITRE.2003.1270584

Chen, Y. (2016). *Comparison of Graph Databases and Relational Databases When Handling Large-Scale Social Data*. University of Saskatchewan.

Chopra, A., Prashar, A., & Chandresh, S. (2013). Natural Language Processing. *International Journal of Technology Enhancements and Emerging Engineering Research*, *1*(4), 131–134.

Choudhary, L., & Burdak, B. S. (2012). Role of Ranking Algorithms for Information Retrieval. *International Journal of Artificial Intelligence & Applications*, *3*(4), 203–220. http://doi.org/10.5121/ijaia.2012.3415

Chowdhary, K. . (2012). Natural Language Processing. Jodhpur,India: MBM Engineering College. Retrieved from http://www.krchowdhary.com/me-nlp12/nlp-01.pdf

Chowdhury, G. (2003). Natural Language Processing. *Annual Review of Information Science and Technology*, *37*(1), 51–89. http://doi.org/10.1002/aris.1440370103

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. *The Journal of Machine Learning Research*, *12*, 2493–2537. http://doi.org/10.1.1.231.4614

Conroy, R. (2016). Sample Size: A Rough Guide. Dublin, Ireland: Royal College of Surgeons in Ireland. http://doi.org/10.1080/08897077.2011.640215

Cornuéjols, A., Wemmert, C., Gançarski, P., & Bennani, Y. (2018). Collaborative Clustering: Why, When, What and How. *Information Fusion*, *39*, 81–95. http://doi.org/10.1016/j.inffus.2017.04.008

Croft, W. B., Metzler, D., & Strohman, T. (2015). *Information retrieval in practice*. New York, United States: Pearson. Retrieved from ciir.cs.umass.edu/downloads/SEIRiP.pdf

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C., Dimitrov, M., … Aswani, N. (2017). Developing language processing components with GATE (a user guide). *University of Sheffield*. Sheffield, South Yorkshire, England: University of Sheffield, England. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.122.1949&rep=rep1&type=pdf

Daly, L. (2011). High-performance XML parsing in Python with lxml. Retrieved October 25, 2017, from https://www.ibm.com/developerworks/library/x-hiperfparse/index.html

De Marzi, M. (2012). Neo4j Internals. Retrieved November 16, 2018, from https://dzone.com/articles/neo4j-internals

de Villiers, M. R. (2005). Interpretive Research Models for Informatics: Action Research, Grounded Theory, and the Family of Design-and-Development Research. *Alternation*, *12*(2), 10–52. Retrieved from http://alternation.ukzn.ac.za/Files/docs/12.2/02 deV.pdf

Dessau, L., & Wodak, T. (2003). Seven Steps To Clearer Judgment Writing. Sydney, Australia: Education Monograph. Retrieved from mja.gov.in/Site/Upload/GR/7Steps_2ClearerJudgmentWriting.pdf

eBravia. (2018). eBravia: Our Solutions. Retrieved April 10, 2018, from https://ebrevia.com/#our-solutions

EliteDataScience. (2017). 5 Tasty Python Web Scraping Libraries. Retrieved October 25, 2017, from https://elitedatascience.com/python-web-scraping-libraries

Equivo. (2012). Equivio Zoom: The E-Discovery Platform for Predictive Coding and Analytics. Kensington, United States: Equivo. Retrieved from http://www.equivio.com/files/files/Product Brief - Equivio Zoom.pdf

Firdhous, M. (2010). Automating Legal Research through Data Mining. *International Journal of Advanced Computer Science and Applications*, *1*(6), 9–16.

Frekjmr, E., Hertzum, M., & Hornbaek, K. (2000). Measuring Usability: Are Effectivness, Efficiency, and Satisfaction Really Correlated? *CHI Letters*, *2*(1), 345–352.

Galgani, F., & Hoffmann, A. (2010). LEXA : Towards Automatic Legal Citation Classification. In J. Li (Ed.), *Advances in Artificial Intelligence 23rd Australasian Joint Conference Adelaide Australia, December 2010 Proceedings*. Adelaide: Springer-Verlag New York Inc. http://doi.org/10.1007/978-3-642-17432-2_45

Garlan, D., & Shaw, M. (1993). An Introduction to Software Architecture. *Advances in Software Engineering and Knowledge Engineering*, *1*(January), 1–39. http://doi.org/10.1142/9789812798039_0001

Gleich, D. F. (2014). PageRank beyond the Web. *SIAM Review*, *57*(3), 321–363. http://doi.org/10.1137/140976649

Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., & Fdez-Riverola, F. (2013). Web Scraping Technologies in an API World. *Briefings in Bioinformatics*, *15*(5), 788–797. http://doi.org/10.1093/bib/bbt026

Gormley, C., & Tong, Z. (2015). Elasticsearch: The Definitive Guide [2.x]. Retrieved April 28, 2017, from https://www.elastic.co/guide/en/elasticsearch/guide/current/intro.html

Goyvaerts, J., & Levithan, S. (2009). *Regular Expressions Cookbook*. (A. Oram, Ed.) (1st ed.). Sebastopol, United States: O'Reilly Media, Inc.

Gregor, S., & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly*, *37*(2), 337–355. http://doi.org/10.2753/MIS0742-1222240302

Gupta, V., & Lehal, G. S. (2009). A Survey of Text Mining Techniques and Applications. *Journal of Emerging Technologies in Web Intelligence*, *1*(1), 17. http://doi.org/10.4304/jetwi.1.1.60-76

Gurusamy, V., & Kannan, S. (2014). Preprocessing Techniques for Text Mining. In *RTRICS*. Retrieved from https://www.researchgate.net/profile/Vairaprakash_Gurusamy/publication/273127322_Preprocessing_Techniques_for_Text_Mining/links/54f8319e0cf210398e949292.pdf?inViewer=0&pdfJsDownload=0&origin=publication_detail%5Cnhttps://www.researchgate.net/publication/2

Hachey, B., & Grover, C. (2005). Automatic Legal Text Summarisation: Experiments with Summary Structuring. In *Proceedings of the 10th international conference on Artificial intelligence and Law* (pp. 75–84). Bologna, Italy: ACM New York. http://doi.org/10.1145/1165485.1165498

Han, Jiawei, Kamber, Micheline, Pei, J. (2012). *Data Mining Concepts and Techniques*. *PhD Proposal* (Third Edit, Vol. 3). http://doi.org/10.1017/CBO9781107415324.004

Hevner, A., March, S., Park, J., & Ram, S. (2004). Design Science Research in Information Systems. *MIS Quarterly*, *28*(1), 75–105. Retrieved from https://pdfs.semanticscholar.org/fa72/91f2073cb6fdbdd7c2213bf6d776d0ab411c.pdf

Hevner, A. R. (2007). A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, *19*(2), 87–92. http://doi.org/http://aisel.aisnet.org/sjis/vol19/iss2/4

Hopcroft, J., Motwani, R., & Ullman, J. (2006). *Introduction To Automata Theory, Languages, and Computation* (3rd ed.). Boston: Addison-Wesley Longman Publishing Co., Inc. Retrieved from http://ieeexplore.ieee.org/document/5392601/

Houlihan, D. (2017). *ROSS Intelligence and Artificial Intelligence in Legal Research*. Boston. Retrieved from http://bluehillresearch.com/ross-intelligence-and-artificial-intelligence-in-legal-research/

Ifijeh, G. I. (2010). Information explosion and University Libraries: Current Trends and Strategies for Intervention. *Chinese Librarianship: An International Electronic Journal*, *Dec. 2010*(30), 1–15. Retrieved from http://www.iclc.us/cliej/cl30doraswamy.pdf

Iida, R., Inui, K., & Matsumoto, Y. (2006). Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL ACL 06* (pp. 625–632). Sydney, Australia. http://doi.org/10.3115/1220175.1220254

Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS Transactions on Computers*, *4*(8), 966–974. Retrieved from

http://www.math.upatras.gr/~esdlab/oldEsdlab/en/members/kotsiantis/Text Classification final journal.pdf%5Cnhttp://www.scopus.com/inward/record.url?eid=2-s2.0-23444448953&partnerID=40&md5=11a5f24b7ee05d580eccaf940e3499e4

Indurkya, N., & Damerau, F. (2010). *Handbook of Natural Language Processing* (2nd ed.). Chapman & Hall/CRC.

Ingersoll, G. (2015). 5 open source natural language processing tools. Retrieved September 18, 2017, from https://opensource.com/business/15/7/five-open-source-nlp-tools

Jain, A., Sharma, R., Dixit, G., & Tomar, V. (2013). Page Ranking Algorithms in Web Mining, Limitations of Existing methods and a New Method for Indexing Web Pages. In *Proceedings - 2013 International Conference on Communication Systems and Network Technologies, CSNT 2013* (pp. 640–645). Gwalior, India: IEEE. http://doi.org/10.1109/CSNT.2013.137

jARVEST. (2017). jARVEST. Retrieved October 25, 2017, from https://sing.ei.uvigo.es/jarvest/manual.html

Jemini, M. (2018). Creation of Adjacency Matrix. Retrieved November 17, 2018, from https://www.includehelp.com/ds/creation-of-adjacency-matrix.aspx

Jiang, J. (2012). Information Extraction from Text. In C. Aggarwal & C. Zhai (Eds.), *Mining Text Data* (pp. 11–41). Boston: Springer,Boston, MA. http://doi.org/10.1007/978-1-4614-3223-4

Johannesson, P., & Perjons, E. (2012). *A Design Science Primer* (1st ed.). Lexington: CreateSpace Independent Publishing Platform.

Jouili, S., & Vansteenberghe, V. (2013). An empirical comparison of graph databases. In *Social Computing (SocialCom)* (pp. 708–715). Alexandria, United States: IEEE. http://doi.org/10.1109/SocialCom.2013.106

Kabakus, A. T., & Kara, R. (2017). A performance evaluation of in-memory databases. *Journal of King Saud University - Computer and Information Sciences*, *29*(4), 520–525. http://doi.org/10.1016/j.jksuci.2016.06.007

Katz, W. (2002). *Introduction to Reference Work* (1st ed.). New York, United States: McGraw-Hill.

Kaza, S., Hu, P. J., & Chen, H. (2011). Designing , Implementing , and Evaluating Information Systems for Law Enforcement — A Long-Term Design-Science Research Project. *Communications of the Association for Information Systems*, *29*(28).

Khaso, M. (2016). How Much Data is Produced Every Day? Retrieved May 16, 2017, from http://www.northeastern.edu/levelblog/2016/05/13/how-much-data-produced-every-day/

Kolosovskiy, M. (2009). Data Structure for Representing a Graph: Combination of Linked List and Hash table. *ArXiv.Org*. Retrieved from http://arxiv.org/abs/0908.3089

Krapivin, M., & Marchese, M. (2008). Focused page rank in scientific papers ranking. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5362 LNCS*, 144–153. http://doi.org/10.1007/978-3-540-89533-6-15

Kuchling, A. . (2018). Regular Expression HOWTO. Retrieved September 28, 2018, from https://docs.python.org/2/howto/regex.html

Kumari, T., Gupta, A., & Dixit, A. (2014). Comparative Study of Page Rank and Weighted Page Rank Algorithm. *International Journal of Innovative Research in Computer and Communication Engineering*, *2*(2), 2929–2937.

Lamkanfi, A., Demeyer, S., Soetens, Q. D., & Verdonck, T. (2011). Comparing Mining Algorithms for Predicting the Severity of a Reported Bug. *Proceedings of the European Conference on Software Maintenance and*

Reengineering, CSMR, 249–258. http://doi.org/10.1109/CSMR.2011.31

LAW.gov. (2016). Sentencing Guidelines. Retrieved March 25, 2017, from https://www.loc.gov/law/help/sentencing-guidelines/southafrica.php#Guidelines

LexisNexis. (2016). Legal Citator. Retrieved April 28, 2017, from https://www.mylexisnexis.co.za/Index.aspx#

LexisNexis. (2017a). About Us - LexisNexis. Retrieved May 11, 2017, from https://www.lexisnexis.com/en-us/about-us/about-us.page

LexisNexis. (2017b). Casebase Case Citator Online - Legal Reference | LexisNexis. Retrieved May 13, 2017, from http://www.lexisnexis.com.au/en-AU/products/CaseBase-Case-Citator-online.page

Liddy, E. (2001). *Natural Language Processing*. *Encyclopedia of Library and Information Science* (2nd ed.). New York, United States: Marcel Decker, Inc. http://doi.org/10.1017/S0267190500001446

Liu, B. (2011). *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. (M. Carey & S. Ceri, Eds.) (2nd ed.). New York, United States: Springer-Verlag New York Inc.

LXML. (2017). lxml - Processing XML and HTML with Python. Retrieved October 25, 2017, from http://lxml.de/index.html

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 55–60). Baltimore, Maryland, United States of America. http://doi.org/10.3115/v1/P14-5010

March, S. T., & Storey, V. C. (2008). Design Science in the Information Sytems Discipline: An Introduction to the Special Issue on Design Science Research. *MIS Quarterly*, *32*(4), 725–730. http://doi.org/192.96.15.27

Marr, B. (2016). How Big Data Is Disrupting Law Firms And The Legal Profession. Retrieved May 15, 2017, from https://www.forbes.com/sites/bernardmarr/2016/01/20/how-big-data-is-disrupting-law-firms-and-the-legal-profession/#1c90a1d97c23

Martin, P. W. (2013). *Basic Legal Citation*. Ithaca: Legal Information Institute.

Marzagão, T. (2013). Web Scraping with Selenium - part 1. Retrieved October 25, 2017, from http://thiagomarzagao.com/2013/11/12/webscraping-with-selenium-part-1/

Mason, R. (1986). Four Ethical Issues of the Information Age. *MIS Quarterly1*, *10*(1), 5–12. http://doi.org/10.2307/248873

Menditto, A., Patriarca, M., & Magnusson, B. (2007). Understanding the meaning of accuracy, trueness and precision. *Accreditation and Quality Assurance*, *12*(1), 45–47. http://doi.org/10.1007/s00769-006-0191-z

Meyer, M., Helfert, M., Donnellan, B., & Kenneally, J. (2012). Applying Design Science Research for Enterprise Architecture Business Value Assessments. In *Design Science Research in Information Systems. Advances in Theory and Practice* (pp. 108–121). Las Vegas, Navada: Springer, Berlin, Heidelberg. http://doi.org/10.1007/978-3-642-29863-9_9

Molloy Librarian. (2017). Databases & Finding Articles: Boolean Searching. Retrieved October 31, 2017, from https://molloy.libguides.com/c.php?g=58070&p=373272

MongoDB. (2018a). Introduction to MongoDB. Retrieved November 21, 2018, from https://docs.mongodb.com/manual/core/document/

MongoDB. (2018b). MongoDB Architecture Guide. New York, United States: MongoDB, Inc. Retrieved from

https://jira.mongodb.org/secure/attachment/.../MongoDB_Architecture_Guide.pdf

MongoDB. (2018c). Top 5 Considerations When Evaluating NoSQL Databases. New York, United States: MongoDB, Inc. Retrieved from https://webassets.mongodb.com/_com_assets/collateral/10gen_Top_5_NoSQL_Considerations.pdf?_ga=2.206650074.837077773.1542003309-1760698149.1541612457

Moniruzzaman, A., & Hossain, S. (2013). NoSQL Database: New Era of Datbases for Big Data Analytics - Classification, Characteristics and Comparison. *International Journal of Database Theory and Application*, *6*(4), 43–45. http://doi.org/10.1016/S0262-4079(12)63205-9

Myers, D., & McGuffee, J. W. (2015). Choosing Scrapy. *Journal of Computing Sciences in Colleges*, *31*(1), 83–89. Retrieved from https://www.researchgate.net/publication/314179276_Choosing_Scrapy

Myers, M., & Venable, J. (2014). A set of ethical principles for design science research in information systems. *Information Management*, *51*(6), 801–809. http://doi.org/10.1016/j.im.2014.01.002

NLTK Project. (2017). Natural Language Toolkit — NLTK 3.2.5 documentation. Retrieved October 9, 2017, from http://www.nltk.org/

Parmar, R., & Roy, S. (2018). MongoDB as an Efficient Graph Database: An Application of Document Oriented NOSQL Database. *Advances in Parallel Computing*, *29*(February), 331–358. http://doi.org/10.3233/978-1-61499-814-3-331

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. http://doi.org/10.2753/MIS0742-1222240302

Piskorski, J., & Yangarber, R. (2013). Information Extraction: Past, Present and Future. In T. Poibeau, H. Saggion, J. Piskorski, & R. Yangarber (Eds.), *Multi-source, Multilingual Information Extraction and Summarization* (pp. 23–50). Berlin: Springer-Verlag Berlin Heidelberg. http://doi.org/10.1007/978-3-642-28569-1

Pokorný, J., Valenta, M., & Kovačič, J. (2017). Integrity constraints in graph databases. *Procedia Computer Science*, *109*(2016), 975–981. http://doi.org/10.1016/j.procs.2017.05.456

Prakash, J., & Kumar, R. (2015). Web Crawling through Shark-Search using PageRank. In *International Conference on Intelligent Computing, Communication & Convergence* (Vol. 48, pp. 210–216). Elsevier Masson SAS. http://doi.org/10.1016/j.procs.2015.04.172

Prasse, P., Sawade, C., Landwehr, N., & Scheffer, T. (2015). Learning to Identify Regular Expressions that Describe Email Campaigns. *Journal of Machine Learning Research*, *16*, 3687–3720.

Programiz. (n.d.). Adjacency List. Retrieved November 17, 2018, from https://www.programiz.com/dsa/graph-adjacency-list

Rabin, M. O., & Scott, D. (1959). Finite Automata and Their Decision Problems. *IBM Journal of Research and Development*, *3*(2), 114–125. http://doi.org/10.1147/rd.32.0114

RAVN Systems. (2016). The Power of Understanding Artificial Intelligence in The Legal World (White Paper). London, England: RAVN Systems. Retrieved from https://www.ravn.co.uk/ravn-publishes-white-paper-power-understanding-ai-legal-world/

Richardson, L. (2015). Beautiful Soup Documentation. Retrieved October 25, 2017, from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Robinson, I., Webber, J., & Eifrem, E. (2015). *Graph Databases*. (M. Beaugureau, Ed.)*Joe Celko's Complete Guide to NoSQL* (2nd ed.). Sebastopol, United States: O'Reilly Media, Inc. http://doi.org/10.1016/B978-

0-12-407192-6.00003-0

Roshdi, A., & Roohparvar, A. (2015). Review : Information Retrieval Techniques and Applications. *International Journal of Computer Networks and Communications Security*, *3*(9), 373–377.

Sarker, I. H., Faruque, F., Hossen, U., & Rahman, A. (2015). A survey of software development process models in software engineering. *International Journal of Software Engineering and Its Applications*, *9*(11), 55–70. http://doi.org/10.14257/ijseia.2015.9.11.05

Scrapy. (2016). Scrapy at a glance — Scrapy 1.4.0 documentation. Retrieved October 25, 2017, from https://doc.scrapy.org/en/latest/intro/overview.html

Singh, H., & Sharma, R. (2012). Role of Adjacency Matrix & Adjacency List in Graph Theory. *International Journal of Computers & Technology*, *3*(August 2012), 179–183. http://doi.org/10.24297/ijct.v3i1c.2775

Singh, S. (2018). Natural Language Processing for Information Extraction. *CoRR*, 1–24. Retrieved from http://arxiv.org/abs/1807.02383

Soanes, C., & Stevenson, A. (2004). *The Concise Oxford English Dictionary*. Oxford University Press.

Sumathy, K., & Chidambaram, M. (2013). Text Mining: Concepts, Applications, Tools and Issues–An Overview. *International Journal of Computer Applications*, *80*(4), 29–32. http://doi.org/10.5120/13851-1685

The Apache OpenNLP Development Community. (2011). Apache OpenNLP Developer Community. Retrieved October 9, 2017, from http://opennlp.apache.org/docs/1.8.2/manual/opennlp.html#intro.cli.toolslist

Ulrich, K. T., & Eppinger, S. D. (2012). *Product Development and Computer Aided Design*. (L. H. Spell, Ed.)*Manual of Engineering Drawing* (5th ed.). New York, United States: McGraw-Hill. http://doi.org/10.1016/B978-0-7506-8985-4.00002-4

University of Massachusetts. (2002). *Challenges in Information Retrieval and Language Modeling*. *Report of a Workshop held at the Center for Intelligent Information Retrieval* (Vol. 37). Amherst, Massachusetts. Retrieved from http://doc.utwente.nl/66226/

Vargiu, E., & Urru, M. (2012). Exploiting Web Scraping in a Collaborative Filtering-Based Approach to Web Advertising. *Artificial Intelligence Research*, *2*(1), 44–54. http://doi.org/10.5430/air.v2n1p44

Venable, J., Pries-Heje, J., & Baskerville, R. (2012). A Comprehensive Framework for Evaluation in Design Science Research. *Design Science Research in Information Systems. Advances in Theory and Practice*, *7286*(2012), 423–438. http://doi.org/10.1007/978-3-642-29863-9_31

Venable, J., Pries-Heje, J., & Baskerville, R. (2016). FEDS: A Framework for Evaluation in Design Science Research. *European Journal of Information Systems*, *25*(1), 77–89. http://doi.org/10.1057/ejis.2014.36

Vijayarani, S., Ilamathi, J., Nithya, M., Ilamathi, M. J., Nithya, M., Professor, A., & Research Scholar, M. P. (2015). Preprocessing Techniques for Text Mining - An Overview. *International Journal of Computer Science & Communication Networks*, *5*(1), 7–16. http://doi.org/10.1016/j.procs.2013.05.286

Vogel, L. (2016). Regular Expressions in Java-Tutorial. Retrieved September 28, 2018, from http://www.vogella.com/tutorials/JavaRegularExpressions/article.html

Wagh, R. S. (2014). Exploratory Analysis of Legal Documents using Unsupervised Text Mining Techniques. *International Journal of Engineering Research & Technology*, *3*(2), 2264–2267. Retrieved from http://www.ijert.org/view-pdf/8360/exploratory-analysis-of-legal-documents-using-unsupervised-text-mining-techniques

Web-Harvest. (2017). Web-Harvest Project Home Page. Retrieved October 25, 2017, from http://web-harvest.sourceforge.net/index.php

Webbers Attoryneys, Notaries, C. (2017). Special Courts. Retrieved September 18, 2017, from http://www.webberslaw.com/special-courts/

White, J. P. (2009). Effects of the Information Explosion on Information Literacy. Retrieved May 15, 2017, from http://jacquelinepwhite.wordpres.com/2009/09/28/effects-of-the-information-explosion-on-information-literacy/

Yambu. (2018). Tribunals in South Africa. Retrieved September 25, 2018, from https://www.yambu.co.za/tribunals/

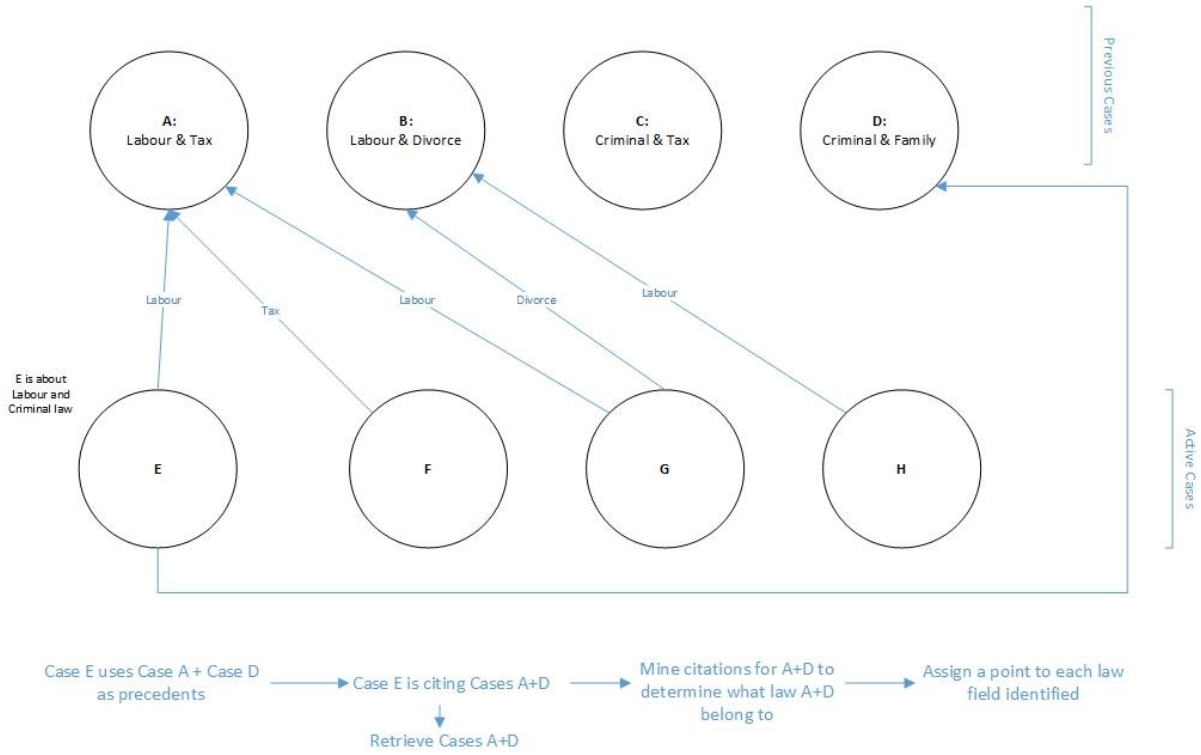
Yin, R. K. (2014). *Case Study Research: Design and Methods 5th Edition* (5th ed.). London, England: SAGE Publications, Inc.

Young, B. (2010). The Role of Stakeholder Perceptions during IT- Enabled Change : An Investigation of Technology Frames of Reference in a Sales Process Innovation Project. Atlanta, Georgia: Georgia State University. Retrieved from http://scholarworks.gsu.edu/cis_diss/40

Zhang, Z. (2017). Graph Databases for Knowledge Management. *IT Professional*, *19*(6), 26–32. http://doi.org/10.1109/MITP.2017.4241463

Appendices

Appendix A: Visualisation of Research Problem



Repeating the above process will result in a table resembling the following:

Law field	**Case**	**Points**
Labour	A	2
Tax	A	1
Labour	B	1
Divorce	B	1
Criminal	C	0
Tax	C	0
Family	D	0
Criminal	D	0

Return the Most Applied Case. In this example, the Most Applied Case for Labour Law is from Case A. Return the rest of the cases followed by each case's ranking.

Appendix B: Responses from LexisNexis

First Questionnaire

Based on research conducted, a preliminary set of requirements for the proposed system are:

- Use of text mining in conjunction with (Gupta & Lehal, 2009):
  - Information retrieval;
  - Topic tracking;
  - Summarisation;
  - Categorisation; and
  - Machine learning.
- Text mining algorithms that could be implemented include (Lamkanfi, Demeyer, Soetens, & Verdonck, 2011):
  - Naïve Bayes;
  - K-nearest neighbour; and
  - Support vector machines.
- Documents should be pre-processed by means of Document Representation (Ikonomakis et al., 2005).

Questions

1. **Experts**:
   a. Who are the ultimate experts of this project and what are their roles?

Technical Development: Lee Adriaanse (Technical Research and Compliance Manager) and Leon Rajindrapersad (Online and Mobile Solutions Architect)

Law Reports and Content Experts: Cindy Naidoo (Senior Editor – Judgments Online) and Rene' Devprasad (Managing Editor – Law Reports)

LegalCitator (Part of Law Reports): Adv Christopher Rodel (LegalCitator Editor) and Marcus Jones (Editor & Key Background Knowledge on development of the LegalCitator)

   b. Who will be the end-users of the system that uses my algorithms?
2. **Existing systems**:
   a. What existing systems does LexisNexis have in place that relate to searching for citations?

We currently use ElasticSearch and also have a specialised product called the LegalCitator which provides an analysis of judgments reported by LexisNexis.

   b. What algorithms does LexisNexis' existing system use?

   LexisNexis whatever algorithms are offered by COTS packages. It doesn't currently use any particular algorithm for data mining, entity extraction or dark data research. We are currently embarking with our search providers on an entity extraction exercise using Stanford NLP, NLTK and OPENNLP

   c. Does LexisNexis' existing system make use of any artificial intelligence techniques such as text mining?

   See above. We've experimented a bit with KNIME for prototyping.

   d. What functions does the existing system currently perform?

   Basic Search

e. What functions does the existing system not perform that you would like it to do?

Entity Extraction, Sentiment Analysis with Machine Learning capabilities.

f. Is there any documentation related to the existing system in LexisNexis that I can read through?

Standard ElasticSearch documentation available online.

3. What related systems are available on the market?

Watson, Beagle, ROSS Intelligence, Equivio, Premonition, eBravia, Cognitiv+ - the list is extensive

4. **Data**:
    a. What data do you have available?

    Case Law and Legislation – xml format

    b. How will I access the data?

    We'll organise through sFTP site.

    c. What format is the data in?

    XML

    d. What processes are followed to collect the data?

    Will discuss on your visit.

5. Who should I contact if I have any queries? Cindy Naidoo or Lee Adriaanse will be able to assist or help you take your queries forward to the necessary parties.

Second Questionnaire

Follow-up questions to Round One

1. What processes are followed to **collect** the data?
    a. Who enters/submits the data? Various Editors who are responsible for the different Law Reports publications, followed by the designated LegalCitator Editor.
    b. How is the data captured?
        i. For LegalCitator – inside of a desktop LegalCitator Editor application
        ii. For other series – in a word document that is styled and eventually converted to xml
    c. How often is the data entered/submitted? The time frames vary, depending on the publications. Some data is updated daily and some monthly, by the Law Reports Editors and our Electronic Publishing Team. Thereafter, the LegalCitator Editor does his updates on a monthly basis.
2. How will I access the data? Access to our live site has been created for you. We will also be able to send you copies of the relevant files in word or pdf format if necessary.
3. Who will be the end-users of the updated LegalCitator system? Internal (staff) and external customers.

Round Two Questions

LegalCitator questions:

1. What is your ultimate end-goal for the system? The goal is to make this product as user friendly as possible, saving the customer as much research time as possible. – while drawing as much value out of our current content set for our customers.
2. How do users of the LegalCitator use the system? i.e. do they just search for legal documents, find what they need and present it as part of their defence? The information found on the LegalCitator would need to be worked into a legal practitioner's argument before presenting.

Information Retrieval and Data questions:

Information Retrieval (IR) is a process that deals with the representation, storage, and searching of a collection of data in response to a request from a user (Roshdi & Roohparvar, 2015).

1. What IR processes does LexisNexis follow?
    a. No formal process. Visual pattern identification and document meta-data markup
2. Where does LexisNexis store the Case Law and Legislation data?
    a. Reference to the content in SQL DB. The content itself is an xml file on a file share locally and this is replicated to the production environment at our Vodacom data centre
3. How does LexisNexis store the Case Law and Legislation data?
    a. XML File format
4. If Case Law and Legislation data is converted from XML format, what processes are followed to do the conversion?
    a. Its converted to XML from a word document, using Word Styles which are mapped to XML elements in a tool called LMT – Link Management Tool

Text Mining questions:

Text mining follows the process of extracting information from unstructured pieces of text and converting the information into knowledge. Pieces of text include, amongst others, emails and full-text documents (Gupta & Lehal, 2009).

1. Are the Case Law and Legislation data structured, unstructured, or semi-structured?
    a. Semi-structured
2. How do you think text mining will benefit the LegalCitator system?
    a. It will get more untreated product to the client more quickly
3. Why do you require the system to support entity extraction and sentiment analysis?
    a. For the process of supporting the LegalCitator product and to assist in treating the vast amounts of untreated case law series.

General information questions:

1. What are the key factors that will be used to determine if the project is a success?
    a. Process and extract ALL SA legal cases
    b. Save extracted legal cases for future use
    c. Help in recommending the MAC
2. What are the types of data collected? i.e. documents, images, keywords etc.
    a. Information, keywords, phrases
    b. ???? uncertain if you mean collected from your process or the input to your process
3. How much Case Law and Legislation data do you have?
    a. 100 000 Law Reports
4. From what time-period does the data start?
    a. 1994 onwards

5. Have you looked at using a system that is on the market? i.e. Watson or Beagle
    a. We've investigated it.

Email Responses – August 2017

1. What process is followed to categorise a legal case? i.e. categorising a case as being Criminal Law. Here, the person summarising the judgment for us decides which category of law it would fit into after reading through and summarising the case. Key words pertaining to that particular category are allocated to each judgment.
    - Is this process done manually? If yes, what steps are followed to do the categorisation? At the moment, yes, it is done manually. The summariser, with a legal background and legal knowledge, would read through the judgment in order to determine what area of law it would fit under.
    - If the process is not done manually, what system is used to do the categorisation?
2. What process is followed to enter a case's data into the database? i.e. does a document first have to be created containing the relevant data, then added to a database? Yes. Each judgment is "styled" into a specific LexisNexis Law Reports structure, using a template. The document is then proofread, keywords and summaries are added, additional work such as adding parallel citations is done. Once this is completed, the hard copy version of for example, the All SA Law Reports is sent out to the printers and then using the same word files that were worked on to create this, the Electronic Product Team builds each judgment onto our live site.

To add a case to the BLC database we do use a manual process of entering the data from the Word documents into the BLC editor, which happens after the process Cindy has explained.

3. With regards to the databases used for the BLC, what is the difference between the BLC Database and Gracie Database? The Gracies Database is basically a subject-index database. The editor would refer to permanent headings in this database (areas or categories of law) and capture and break down key words from a judgment even further, thereafter attaching these keywords to the case in question as listed in the Gracies database – on the BLC database however, the Editor adds in a lot more info, such as the case details, case history etc. All of this info is ultimately pulled together and displayed in BLC search results.
4. Would it be possible to obtain a diagram of the tables and their respective fields within the BLC and Gracie Database? For this, Christopher Rodel would be able to assist you. (Hi Chris – please add to this for me)

I've attached the BLC schema for you, which should assist.

Follow up Questions

1. When reading through a case, how does the summariser determine the field of law? i.e. looks for key words? Does a word count of certain words play any role? Are any special techniques used?
2. Which database is the final version of a case stored on? Is it the BLC database or Gracies database?
3. Which database is a case accessed from when LexisNexis staff and external users request a case?
4. With regards to the second question, where is a case taken from before you start the process of 'entering a case into the database' -which database is the case sitting on?
5. With regards to the second question, what data is added to a case apart from the keywords, summaries, and parallel citations?
6. Is a case stored on a relational database or is a case stored as one 'document object' in a database?

7. In the second question's answer you refer to a "live site", is this the mylexisnexis.com site?

Additional Questions

Architecture

I'd like to get an understanding of the architecture used for the LegalCitator. I've done a rough diagram of my current understanding (See attached file).

1. Is my diagram correct?
    a. If no, what information am I missing?
2. Does LexisNexis use any Application Servers, Web Servers, and Database Servers for the LegalCitator?
    a. If yes, what are the names of these servers?
    b. What role do these servers play in returning information to a user who is:
        i. using the LegalCitator?
        ii. Using the Mylexisnexis.com website?
3. On which database are the judgements stored?

Legal Citations

1. I am not sure if there is a 'world standard' that all law organisations must use with regards to legal citations. Does LexisNexis use a specific style of legal citations?
    a. Would you be able to point me to any resources that explain how to interpret/use the citation principles?
2. What are parallel citations?
    a. Are parallel citations different from 'normal' legal citations?

LexisNexis Abroad

Do LexisNexis branches overseas use the same legal software/systems like the LexisNexis branch in South Africa. i.e. LexisNexis in South Africa uses the LegalCitator, would LexisNexis in USA use a different product?

Appendix C: Ethics Clearance

NELSON MANDELA
UNIVERSITY

FACULTY OF SCIENCE RTI COMMITTEE

To: Dr B. Scholtz/ T. Padayachy
From: Lynette Roodt
Date: 27 November 2017
Ref: H17-SCI-CSS-009

Dear Dr B. Scholtz/ T. Padayachy

TITLE OF PROJECT: TEXT MINING TECHNIQUES FOR LEGAL CITATION CLASSIFICATION

Your above-entitled application was considered and approved by the Sub-Committee for Ethics in the Faculty of Science on 20 November 2017.

The Ethics clearance reference number is H17-SCI-CSS-009 and is valid for three years. Please inform the Committee, via your faculty officer, if any changes (particularly in the methodology) occur during this time.

An annual affirmation to the effect that the protocols in use are still those, for which approval was granted, will be required from you. You will be reminded timeously of this responsibility, and will receive the necessary documentation well in advance of any deadline.

We wish you well with the project. Please inform your co-investigators of the outcome, and convey our best wishes.

Yours sincerely

Lynette Roodt
Manager: Faculty Administrator
Faculty of Science

Appendix D: Parts of a Legal Case

Minister of Basic Education and others v Basic Education for All and others
[2016] 1 All SA 369 (SCA)

Division:	SUPREME COURT OF APPEAL
Date:	2 December 2015
Case No:	20793/2014
Before:	A CACHALIA, N DAMBUZA, CH LEWIS, MS NAVSA and XM PETSE JJA
Sourced by:	E Thantsa
Summarised by:	DPC Harris

Editor's Summary

The Minister of Basic Education brought an appeal against the High Court's finding that the failure to provide each learner at public schools in Limpopo with a textbook for each subject, prior to the commencement of the 2014 school year, was an infringement of their right to a basic education in terms of section 29(1)(a), the right to equality in terms of section 9, and to dignity in terms of section 10 of the Constitution. The Department of Basic Education ("DBE") had adopted a clear national policy that each learner was to be provided with a textbook for each subject before commencement of the academic year. However, the Department failed to do so in respect of some learners in Limpopo. The second to fifth appellants were members of the Limpopo department which had failed to comply with the policy.

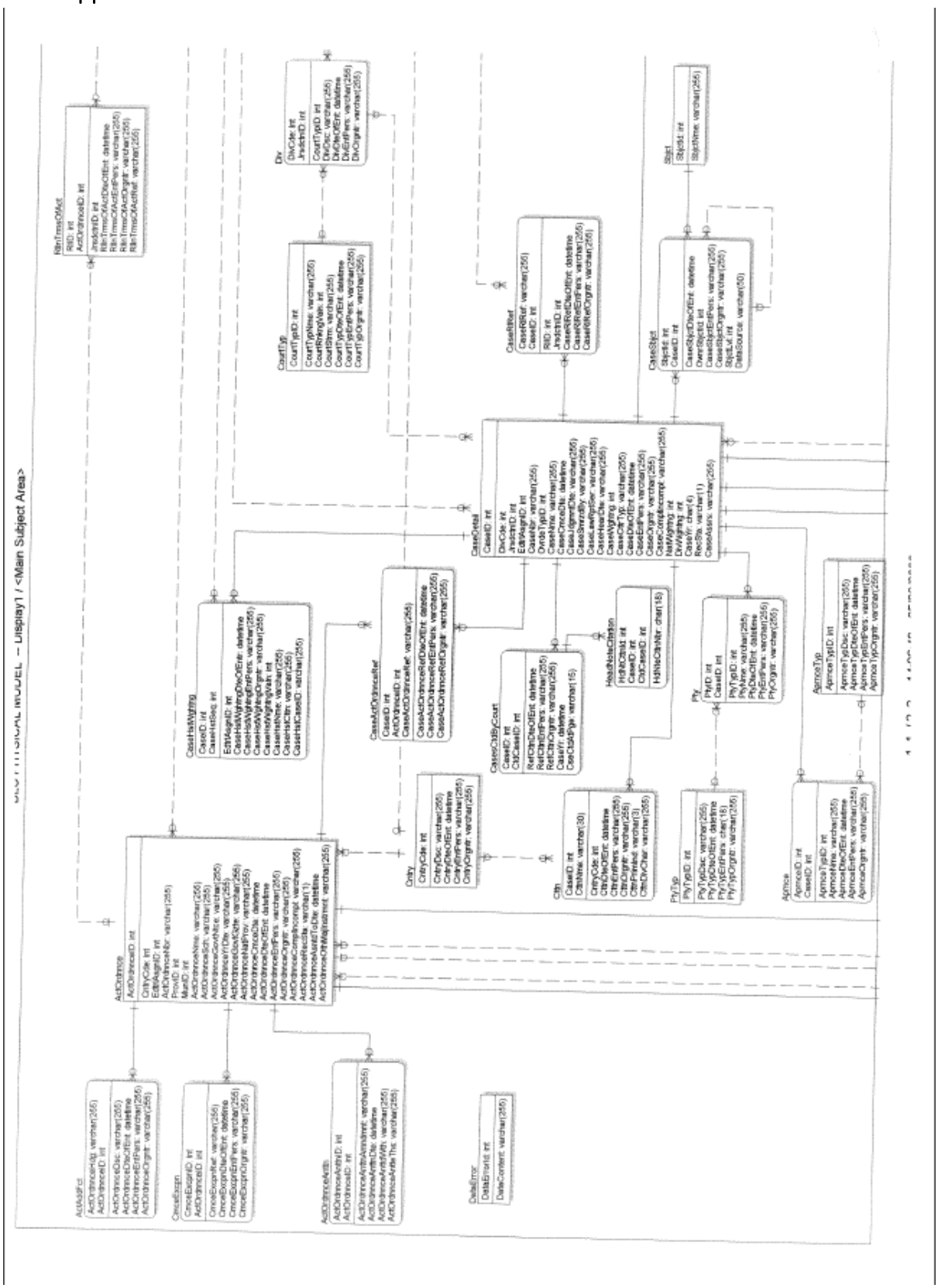
Notes

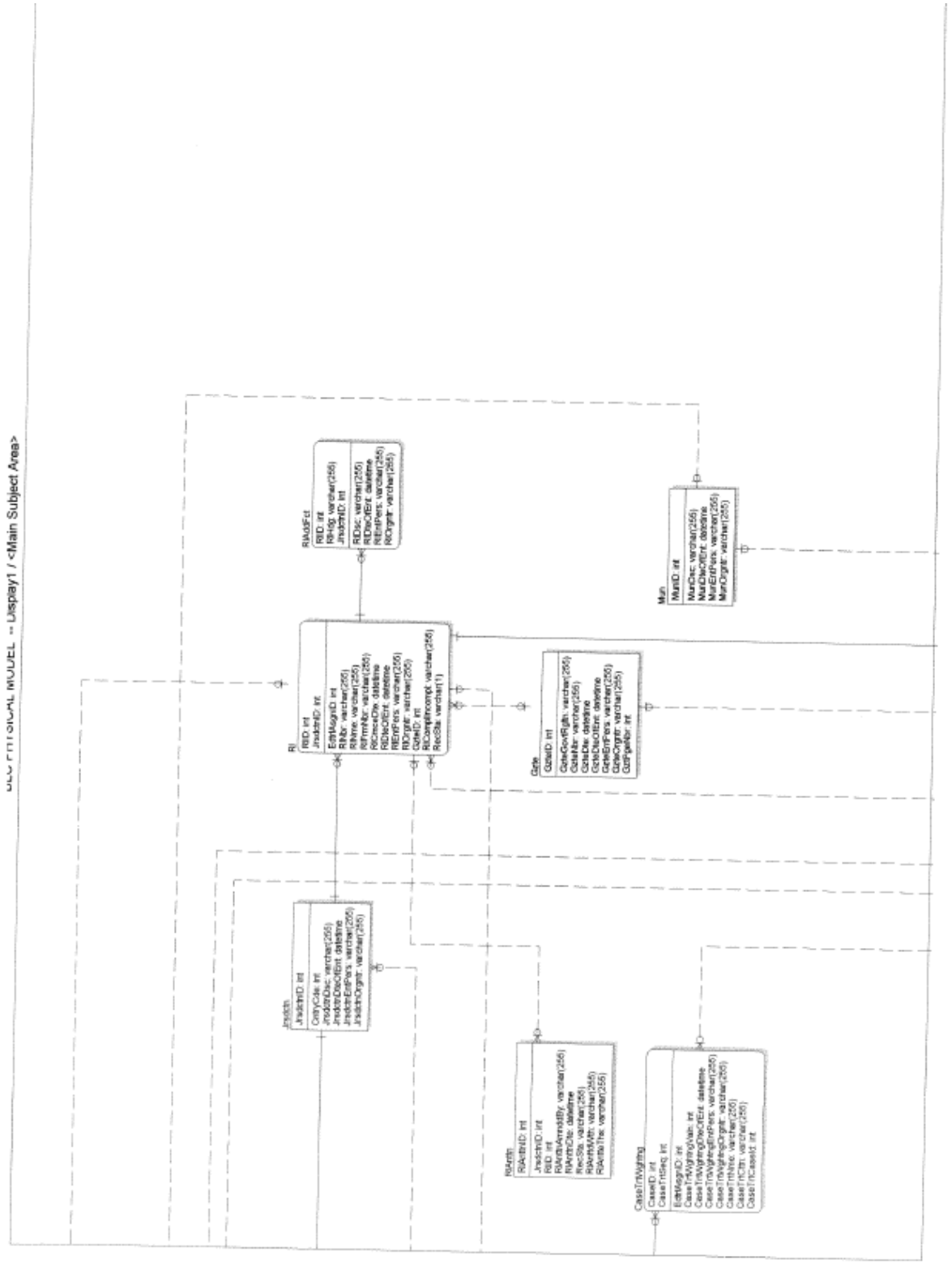
For Constitutional law see:
- *LAWSA* Second Edition Replacement Volume (Vol 5(3), paras 1-334)
- Cheadle MH; Davis DM and Haysom NRL *South African Constitutional Law: The Bill of Rights* (2ed) Durban, LexisNexis 2005 Service Issue 18 (last updated in May 2015)

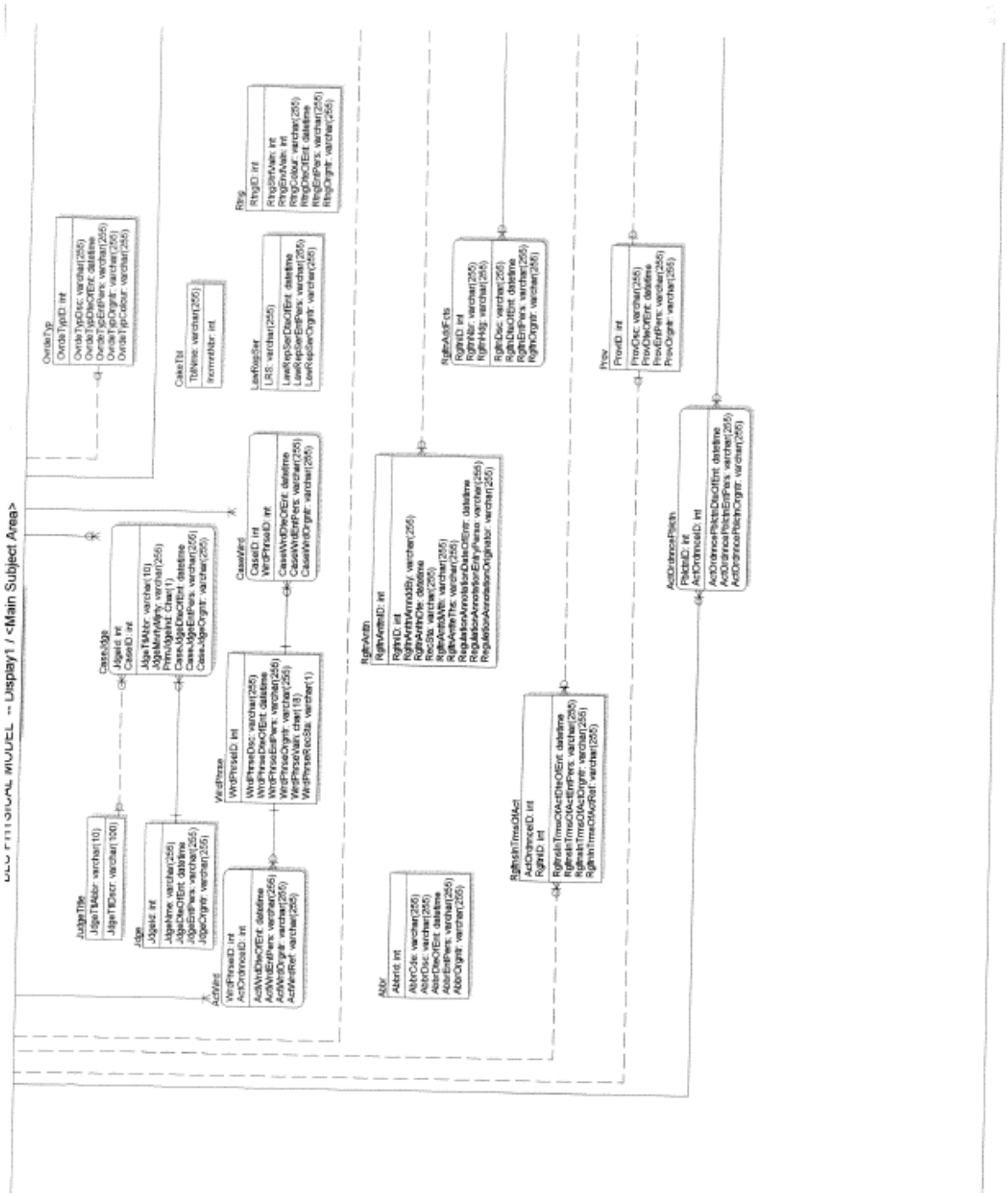
Cases referred to in judgment

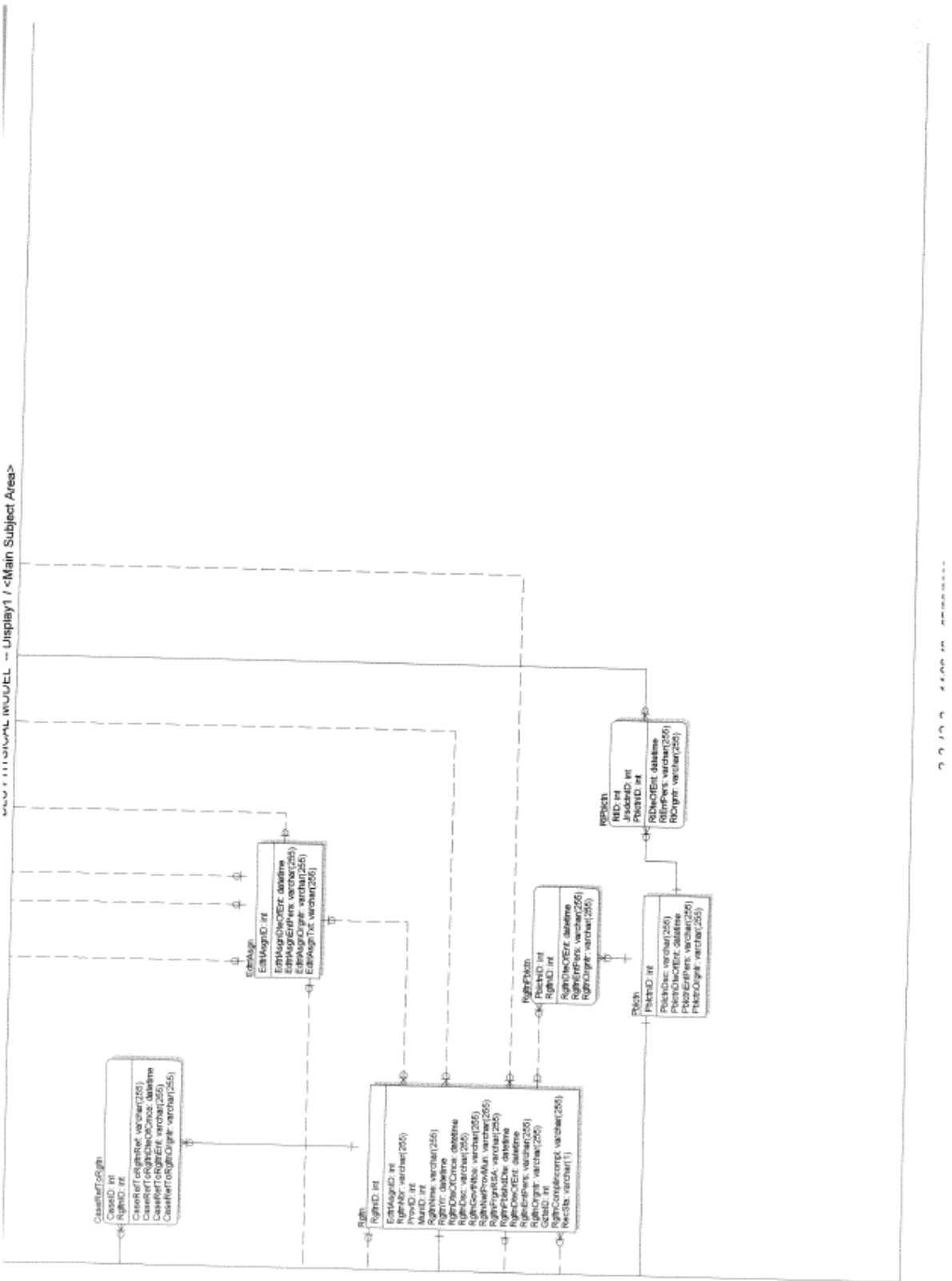
Governing Body of the Juma Musjid Primary School and others v Essay NO and others (*Centre for Child Law and another as amici curiae*) 2011 (8) BCLR 761 ([2011] ZACC 13) (CC) - **Approved**	380
Government of the Republic of South Africa and others v Grootboom and others 2000 (11) BCLR 1169 ([2000] ZACC 19; 2001 (1) SA 46) (CC) - **Referred to**	385
Harksen v Lane NO and others 1997 (11) BCLR 1489 ([1997] ZACC 12; 1998 (1) SA 300) (CC) - **Referred to**	386

Appendix E: The BLC Schema




Logical Model -- Display1 / <Main Subject Area>
RAddFct
RID: int
RHdg: varchar(256)
JnsdctnID: int
RDsc: varchar(255)
RDscOfEnt: datetime
REntPers: varchar(255)
ROrgnlr: varchar(255)
Mun
MunID: int
MunDsc: varchar(256)
MunDscOfEnt: datetime
MunEntPers: varchar(255)
MunOrgnlr: varchar(255)
RI
RID: int
JnsdctnID: int
EdtMagnID: int
RMdr: varchar(255)
RMme: varchar(255)
RPrmMdr: varchar(255)
RDscOfCre: datetime
RDscOfEnt: datetime
REntPers: varchar(255)
ROrgnlr: varchar(255)
GateID: int
RDoesItIncompl: varchar(255)
RecSta: varchar(1)
Gate
GateID: int
GateGovRgtn: varchar(255)
GateMdr: varchar(255)
GateDte: datetime
GateDscOfEnt: datetime
GateEntPers: varchar(255)
GateOrgnlr: varchar(255)
GatePgeNbr: int
Jrsdctn
JnsdctnID: int
CntyCde: int
JnsdctnDsc: varchar(255)
JnsdctnDscOfEnt: datetime
JnsdctnEntPers: varchar(255)
JnsdctnOrgnlr: varchar(255)
RAdmn
RAdmID: int
JnsdctnID: int
RID: int
RAdmAmndBy: varchar(256)
RAdmDte: datetime
RecSta: varchar(255)
RecordMen: varchar(1048)
RAdmTitle: varchar(255)
CaseTrkNgtng
CaseID: int
CaseEntSeq: int
EdtMagnID: int
CaseTrkNgtngVale: int
CaseTrkNgtngDscOfEnt: datetime
CaseTrkNgtngEntPers: varchar(255)
CaseTrkNgtngOrgnlr: varchar(255)
CaseTrkMnc: varchar(256)
CaseTrkTtn: varchar(256)
CaseTrkCaseId: int

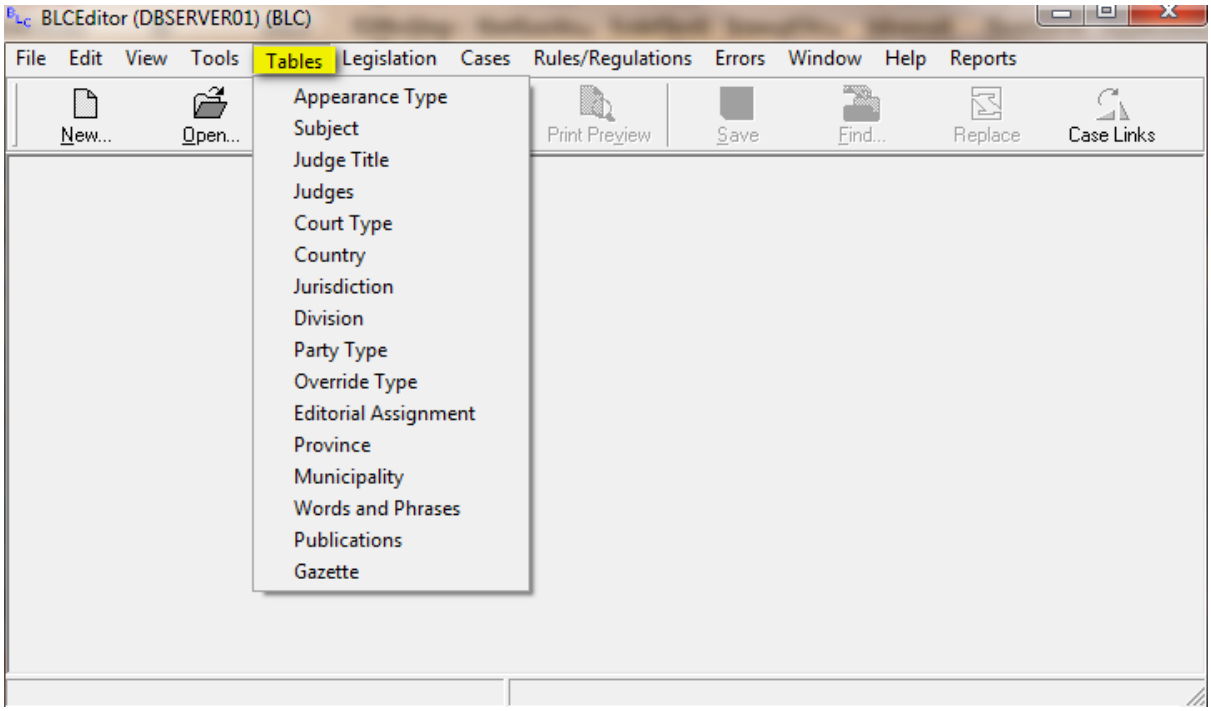

Display1 / <Main Subject Area>
OrdreTyp
OrdreTypID: int
OrdreTypDesc: varchar(255)
OrdreTypDtOfEnt: datetime
OrdreTypEntPers: varchar(255)
OrdreTypOrgnr: varchar(255)
OrdreTypColour: varchar(255)
CaseTil
TilNme: varchar(255)
IncmnNbr: int
Ring
RingID: int
RingEntVain: int
RingColour: varchar(255)
RingDtOfEnt: datetime
RingEntPers: varchar(255)
RingOrgnr: varchar(255)
LawRepSet
LRS: varchar(255)
LawRepSetDtOfEnt: datetime
LawRepSetEntPers: varchar(255)
LawRepSetOrgnr: varchar(255)
CaseVert
CaseID: int
WrtPhraseID: int
CaseVertDtOfEnt: datetime
CaseVertEntPers: varchar(255)
CaseVertOrgnr: varchar(255)
CaseJdge
JdgeID: int
CaseID: int
JdgeTitANbr: varchar(10)
JdgeMnrMajPty: varchar(255)
PrmrJdgeInd: Char(1)
CaseJdgeDtOfEnt: datetime
CaseJdgeEntPers: varchar(255)
CaseJdgeOrgnr: varchar(255)
WrtPhrase
WrtPhraseID: int
WrtPhraseDesc: varchar(255)
WrtPhraseDtOfEnt: datetime
WrtPhraseEntPers: varchar(255)
WrtPhraseOrgnr: varchar(255)
WrtPhraseMain: char(10)
WrtPhraseRecSta: varchar(1)
RghtArtIn
RghtArtInID: int
RghtID: int
RghtArtInAnnotBy: varchar(255)
RghtArtInDte: datetime
RecSta: varchar(255)
RghtArtInAnnotDtOfEnt: datetime
RghtArtInTxt: varchar(255)
RepublicanAnnotatorDateOfEnt: datetime
RepublicanAnnotatorEntryPerson: varchar(255)
RepublicanAnnotationOriginator: varchar(255)
RghtAddFcts
RghtID: int
RghtNbr: varchar(255)
RghtHdg: varchar(255)
RghtDesc: varchar(255)
RghtDtOfEnt: datetime
RghtEntPers: varchar(255)
RghtOrgnr: varchar(255)
Prov
ProvID: int
ProvDesc: varchar(255)
ProvDtOfEnt: datetime
ProvEntPers: varchar(255)
ProvOrgnr: varchar(255)
JdgeTitle
JdgeTitANbr: varchar(10)
JdgeTitDescr: varchar(100)
Jdge
JdgeID: int
JdgeName: varchar(255)
JdgeDtOfEnt: datetime
JdgeEntPers: varchar(255)
JdgeOrgnr: varchar(255)
ActVert
WrtPhraseID: int
ActOrdnnceID: int
ActVertDtOfEnt: datetime
ActVertEntPers: varchar(255)
ActVertOrgnr: varchar(255)
ActVertRef: varchar(255)
Abbr
AbbrID: int
AbbrCde: varchar(255)
AbbrDesc: varchar(255)
AbbrDtOfEnt: datetime
AbbrEntPers: varchar(255)
AbbrOrgnr: varchar(255)
RghtsInTmsOfAct
ActOrdnnceID: int
RghtID: int
RghtsInTmsOfActDtOfEnt: datetime
RghtsInTmsOfActEntPers: varchar(255)
RghtsInTmsOfActOrgnr: varchar(255)
RghtsInTmsOfActRef: varchar(255)
ActOrdnnceRghts
RghtsID: int
ActOrdnnceID: int
ActOrdnnceRghtsDtOfEnt: datetime
ActOrdnnceRghtsEntPers: varchar(255)
ActOrdnnceRghtsOrgnr: varchar(255)

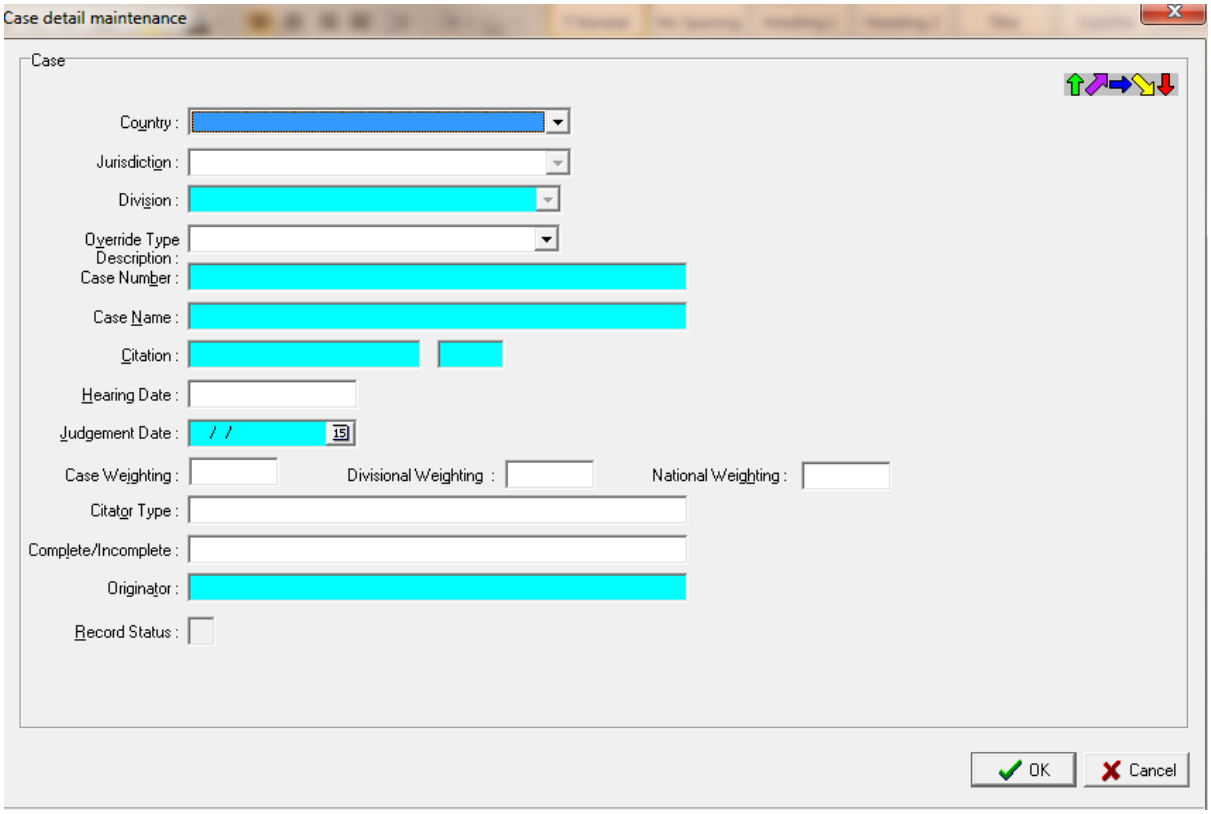

Model -- Display1 / <Main Subject Area>
CaseRefToRight
CaseID: int
RightID: int
CaseRefToRightRef: varchar(255)
CaseRefToRightDecOfOmcs: datetime
CaseRefToRightEnt: varchar(255)
CaseRefToRightOrgnlr: varchar(255)
EditAlign
EditAlignID: int
EditAlignDecOfEnt: datetime
EditAlignEntPers: varchar(255)
EditAlignOrgnlr: varchar(255)
EditAlignTxt: varchar(255)
Right
RightID: int
EditAlignID: int
RightAttr: varchar(255)
PlntID: int
MainID: int
RightName: varchar(255)
RightYr: datetime
RightDecOfOmcs: datetime
RightDoc: varchar(255)
RightGovtNtfcs: varchar(255)
RightNatProvMun: varchar(255)
RightPrgmSA: varchar(255)
RightPblshdDte: datetime
RightDecOfEnt: datetime
RightEntPers: varchar(255)
RightOrgnlr: varchar(255)
GrpID: int
RightCodeRecord: varchar(255)
RecSts: varchar(1)
RightPlntn
PlntnID: int
RightID: int
RightDecOfEnt: datetime
RightEntPers: varchar(255)
RightOrgnlr: varchar(255)
RBPlntn
RBID: int
JrsdctnID: int
PlntnID: int
RBDecOfEnt: datetime
RBEntPers: varchar(255)
ROrgnlr: varchar(255)
Plntn
PlntnID: int
PlntnDec: varchar(255)
PlntnDecOfEnt: datetime
PlntnEntPers: varchar(255)
PlntnOrgnlr: varchar(255)

Appendix F: Project Plan

Task Description	**Deliverable**	**Estimated Deadline**
Introduction	Chapter 1	June 2017 – July 2017
Literature Study	Chapter 2,3,4	August 2017 – October 2017
Complete Chapters 1,2,3	Chapter 1,2,3	End of 2017
Start draft of Chapter 5	Chapter 5 draft	December 2017
Design and Development	Chapter 5	October 2017 – January 2018
Evaluation	Chapter 6	January 2018 – February 2018
Conclusion	Chapter 7	March 2018
Completion of Chapters	First Draft Submission	May 2018
Amendments to Chapters	Second Draft Submission	October 2018
Amendments to Chapters	Final Submission	November 2018

Appendix G: Screenshots from LegalCitator





Appendix H: Test Document Used for Regular Expression Testing

Date: 2018-03-17

Title: Testing Apache Tika Library

Body:

This is a test to see how well text can be extracted using Apache Tika and regular expressions. This will be used as a starting step to extracting text from a legal case. The main Apache Tika tutorial by https://cbrownley.wordpress.com/2016 was followed as guide to get this test working. The main tutorial uses 8 libraries. This test will not use all the libraries.

The following libraries will be used for this test:

tika

re

pandas

For further information contact me on 0123456789 or testingone@mandela.ac.za

This test made use of the Python libraries RE and Apache Tika to extract data from a PDF document. The following data was extracted:

- “Date: 2018-03-17;
- “Title: Testing Apache Tika Library;
- Website address “https://cbrownley.wordpress.com/2016;
- Telephone number “0123456789”; and
- Email address testingone@mandela.ac.za.

Appendix I: Complete Details of Test Documents

Test Documents Used in Experiments			
Document Name	Full Name	Used in	Details
TesterDoc1	TestV2	Experiment 2 Part 1	TesterDoc1 was created by the researcher to test extracting facts. TesterDoc1 was in PDF format. Facts related to date, title, mobile number, email address, and web address were to extracted.
		Experiment 3 Part 2	TesterDoc1 was converted to MS Word .docx format to perform IE on.
T1	Minister of Basic Education and others v Basic Education for All and others [2016] 1 All SA 369 (SCA	Experiment 2 Part 2	T1 was a legal case obtained from LexisNexis. T1 is in PDF format and was used as a base to build the MAC System to extract facts.
		Experiment 3 Part 2	T1 was converted to a MS Word .docx file to perform IE on.
T2	Azisa (Pty) Ltd v Azisa Media CC and another [2002] 2 All SA 488 (C)	Experiment 4	T2 was a legal case obtained from LexisNexis. T2 was used to further build the MAC System as it provided a different structure than T1.
T3	Cliff v Electronic Media Network (Pty) Ltd and another [2016] 2 All SA 102 (GJ)	Experiment 4	T3 was a legal case obtained from LexisNexis. T3 was used to further build the MAC System as it provided a different structure than T1 and T2.
U1	Equal Education v Minister of Basic Education [2018] 3 All SA 705 (ECB)	Experiment 4	U1-U10 were used as unseen cases to test the MAC System.
U2	Food and Allied Workers Union and others v Scandia Delicatessen CC and another [2001] 3 All SA 342 (		
U3	Hoho v S [2009] 1 All SA 103 (SCA)		
U4	Kate%u2019s Hope Game Farm (Pty) Limited v Terblanchehoek Game Farm (Pty) Limited [1997] 4 All SA 185 (A)		
U5	Ketler Investments CC t_a Ketler Presentations v Internet Service Providers%u2019		

	Association [2014] 1 Al		
U6	Minister of Home Affairs and others v Somali Association of South Africa, Eastern Cape (SASA EC) and		
U7	Movie Camera Company (Pty) Ltd v Van Wyk and another [2003] 2 All SA 291 (C)		
U8	Pioneer Foods (Pty) Ltd v Bothaville Milling (Pty) Ltd [2014] 2 All SA 282 (SCA)		
U9	Standard Bank of South Africa Limited v Harris and others [2002] 4 All SA 164 (SCA)		
U10	National Director of Public Prosecutions v Mohunram and others [2007] 4 All SA 704 (SCA)		

Appendix J: 50 Cases CRT

	Results	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	F33	F34	F35	F36	F37	F38	F39	F40	F41	F42	F43	F44	F45	F46	F47	F48	F49	F50
CRT Title	Number of CRT Titles extracted perfectly	2	1	2	2	4	0	10	3	4	0	4	1	6	3	1	2	1	2	2	9	0	3	1	2	14	3	8	1	5	8	5	1	1	1	9	14	0	8	9	3	4	1	0	12	0	3	2	2	1	0
	Number of partialy extracted CRT Titles	27	4	34	1	1	2	1	2	1	1	2	2	4	5	28	1	3	18	1	1	6	4	20	7	1	21	1	3	5	10	4	8	12	9	5	6	0	0	2	2	32	7	1	4	12	0	1	14	16	10	2
	Number of CRT Titles not extracted	15	0	6	1	1	0	0	1	0	0	1	0	4	1	3	0	2	2	1	0	4	1	3	2	3	1	3	0	4	5	6	14	5	6	0	1	3	1	0	14	0	3	1	0	0	0	0	0	2	2	1
	Number of Extra instances	0	0	1	0	0	0	0	1	0	1	1	0	4	0	1	1	0	1	0	0	0	1	0	2	1	0	1	0	0	0	1	0	0	0	0	1	0	0	3	0	1	0	0	1	0	0	0	0	1	0
	Number of Split instances	2	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	2	0	0	0	2	0	0	0	0	0	0	0	1	1	2	3	0
	Number of Junk instances	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	18	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	0	0
CRT Date	Number of CRT Dates extracted perfectly	31	5	38	4	6	2	11	4	5	1	7	3	14	8	32	1	6	22	4	9	10	6	23	10	16	24	12	6	13	14	17	11	6	7	10	16	20	10	51	5	5	8	13	12	11	4	16	19	13	3
	Number of partialy extracted CRT Dates	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	1	1	0	4	0	0	0	0	0	0	0	0	0	1	0	0
	Number of CRT Dates not extracted	13	0	4	0	0	0	0	2	0	0	0	0	0	1	0	2	0	0	0	1	0	2	1	1	2	1	0	0	0	4	1	0	0	0	0	0	1	0	3	0	1	0	0	1	0	0	0	1	0	0	0
	number of Extra instances	0	0	1	0	0	0	0	1	0	1	1	0	4	0	1	1	0	1	0	0	0	1	0	2	1	0	1	0	0	0	1	0	0	0	1	0	0	0	3	0	1	0	0	1	0	0	0	1	1	0	0
	Number of Split instances	2	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	2	0	0	0	2	0	0	0	0	0	0	1	1	2	3	0	0
	Number of Junk instances	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	18	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CRT Journa	Number of CRT Journals extracted perfectly	26	5	30	3	5	2	11	3	5	1	7	3	10	8	29	2	4	20	3	10	7	6	21	8	15	23	9	6	11	13	10	11	6	7	9	12	19	10	41	5	5	5	12	12	11	4	16	18	11	1
	Number of partialy extracted CRT Journals	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	Number of CRT Journals not extracted	17	0	11	1	1	0	0	3	0	0	0	0	4	1	3	1	2	2	1	0	3	2	3	2	3	2	3	0	4	5	8	0	0	0	1	5	2	0	13	0	0	3	1	0	0	0	0	2	2	2	2
	Number of Extra instances	0	0	1	0	0	0	0	1	0	1	1	0	4	0	1	1	0	1	0	0	0	1	0	2	1	0	1	0	0	0	1	0	0	0	1	0	0	0	3	0	1	0	0	1	0	0	0	1	1	0	0
	Number of Split instances	2	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	2	0	0	0	0	2	0	0	0	0	0	0	1	1	2	3	0	0
	Number of Junk instances	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	18	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CRT Action	Number of CRT Actions extracted perfectly	18	5	20	3	6	2	11	2	5	0	3	0	13	1	5	2	2	7	1	9	4	1	2	0	15	7	3	0	4	4	10	0	0	1	10	17	1	9	54	1	0	3	4	11	10	0	2	1	1	2
	Number of partialy extracted CRT Actions	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Number of CRT Actions not extracted	25	0	21	1	0	0	0	4	0	1	4	3	1	8	27	1	4	15	3	1	6	7	22	9	3	18	9	6	11	14	8	11	6	6	0	0	20	1	1	4	5	5	9	1	1	4	14	19	12	1
	Number of Extra instances	0	0	1	0	0	0	0	1	0	1	1	0	4	0	1	1	0	1	0	0	0	1	0	2	1	0	1	0	0	0	1	0	0	0	1	0	0	0	3	0	1	0	0	1	0	0	0	1	1	0	0
	Number of Split instances	2	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	2	0	0	0	0	2	0	0	0	0	0	0	1	1	2	3	0	0
	Number of Junk instances	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	18	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	0	0

Appendix K: ICCECE'18 Conference Paper

An Information Extraction Model Using A Graph Database To Recommend The Most Applied Case

Thashen Padayachy
Department of Computing Sciences
Nelson Mandela University
Port Elizabeth, South Africa
s211228729@mandela.ac.za

Brenda Scholtz
Department of Computing Sciences
Nelson Mandela University
Port Elizabeth, South Africa
brenda.scholtz@mandela.ac.za

Janet Wesson
Department of Computing Sciences
Nelson Mandela University
Port Elizabeth, South Africa
janet.wesson@mandela.ac.za

Abstract— **Legal researchers spend too much time on locating the most applied case. The purpose of this paper is to present a comprehensive model to assist legal researchers in accessing legal data, specifically the most applied case for a field of law. An extensive critical review of the literature was conducted to identify the status of research in this field. The findings revealed that existing studies did not present models to specifically recommend the most applied case, nor did they indicate database methods to store and query legal documents. The model was derived based on the findings from the literature review and extended to include the adoption of a graph database for information storage. A case study approach was used to empirically test the concepts. Testing was limited to a small number of documents as proof of concept and future research will test for scalability. The model can contribute to the research community in the fields of information retrieval, information extraction, database retrieval methods, as well as the legal domain. For legal researchers, the findings can be used to assist them with the process of accessing the most applied case quickly and reducing time spent searching.**

Keywords: Information Retrieval, Information Extraction, Graph Databases, Ranking, Legal Research

I. INTRODUCTION

Throughout the years the value and dependency of information have become critically important, thus resulting in the information explosion [1]. Information can be found in different forms of media and is made up of data, facts, and ideas. Media, containing information, includes printed documents and documents in electronic format. One domain that is particularly affected by the information explosion is the legal domain. Each new case that is heard in court increases the body of knowledge that legal researchers use [2]. A common use for this knowledge is to find precedents. However, finding this knowledge consumes large amounts of time. Specifically, a large amount of time is taken when searching for the most applied case (MAC). The MAC refers to a case that is the most useful and commonly used case for a field of law. The legal domain is struggling with challenges related to information asset management [3], which incorporates all processes for deploying information assets to obtain meaningful insights [4]. Increased client sophistication and rapid development of technology are two challenges facing this domain. Clients have become more informed and have better access to information. However, finding the MAC efficiently and effectively is still a challenge and organisations are relying on human knowledge and manual processes in order to this.

This paper reports on trends in the legal domain related to information and document retrieval, particularly related to studies on retrieving the MAC. The main purpose of the paper is to propose a comprehensive model for recommending the MAC for a field of law. Whilst a model to produce relevant documents based on a user's query was proposed by [5], it does not discuss an efficient way to store processed information. Other studies identified in the literature review of information retrieval (IR) in the legal domain identified approaches such as IR [6], rule-based systems [7], expert systems [8], data visualisation [9], machine learning [10], and text mining [11], [5]. While much research has been conducted within the legal domain, none of the proposed models were comprehensive and did not incorporate database approaches for the storage and retrieval of legal documents. At the time of writing this paper no formal, comprehensive, theoretical model could be found for recommending the MAC for a field of law.

The Design Science Research (DSR) methodology was adapted for this study and this paper reports on the design cycle of [12] [13]. DSR produces one or more artefacts, which is this case is the theoretical model for recommending the MAC and the system prototypes. The model was derived from a critical review of the literature, an extant systems analysis, and interviews with experts from an organisation in the legal domain (for anonymity purposes called LegalCo). Knowledge gained from interviews with legal researchers was used to clarify the processes used for recommending the MAC. This knowledge related to the manual, human process followed by these researchers to find the MAC. The proposed model extends the study of [5] and makes use of Information Retrieval (IR), Information Extraction (IE), and query-independent ranking techniques that must be integrated to produce relevant information for finding the MAC to a legal researcher. The model also incorporates the adoption of a graph database for document storage as an alternative to a relational database.

The structure of the paper is as follows: Section II reports on a critical review of the literature in the legal domain and the automated models and techniques used. Section III will discuss the requirements and design of the proposed model, followed by an implementation of the design in Section IV. The findings of the study will be presented in Section V, followed by the conclusions and limitations of the study in Section VI.

II. RELATED WORK

A. Literature Reviewed

Several studies were identified within the legal domain during the literature review. Each study contributed to a certain field within computer science. However, none of the studies addressed recommending the MAC or saving

978-1-5386-4904-6/18/$31.00 ©2018 IEEE

processed information to an efficient database. Table 1 summarises the studies reviewed.

Title	Field and Context	Aim	Author/s
ROSS Intelligence and Artificial Intelligence in Legal Research	Artificial Intelligence and IR in the legal domain.	To assess the impact of legal research with ROSS AI.	[6]
Towards a Legal Rule-Based System Grounded on the Integration of Criminal Domain Ontology and Rules	Rule-based systems in the legal domain.	To create a legal rule-based decision support system.	[7]
Expert Sytstems with Applications in the Legal Domain	IR and expert systems in the legal domain, specifically legislative acts.	To create an IR system that provides easy access to legislative acts.	[8]
Approaches for information retireval in legal documents	IR and data visualisation in the legal domain.	To create an IR system that provides users with search options based on the semantics of a word, and visualise retrieved documents using semantic networks.	[9]
The Uncertain Promise of Predictive Coding	Machine Learning in the legal domain.	To provide a review on predictive coding technologies.	[10]
LEXA : Towards Automatic Legal Citation Classification	Text mining and machine learning in the legal domain.	To automatically classify legal citations.	[11]
Automating Legal Research through Data Mining	Text mining, IE, and IR in the legal domain.	To create model that uses text mining to automate the legal research process.	[5]

Table 1 Literature reviewed of studies in the legal domain

A study conducted by [6] focused on using the Artificial Intelligence tool called ROSS to facilitate IR in the legal research process. Multiple experiments were conducted using different searching tools such as Boolean searching, natural language searching, searching using ROSS with Boolean searching, and searching using ROSS with natural language searching. Boolean searching was used with keywords to identify documents with Boolean operators to further narrow down the results returned. Natural language searching was referred to as searching by means of a query that is in plain language. Recommending the MAC would require a search tool. The search tools discussed by [6] could be used, but for the purpose of returning documents to a user a natural language searching tool could be better. A legal rule-based decision support system with the aim of assisting with legal reason was proposed [7] and was built using an ontology from the criminal law domain. Similarly, [8] conducted a study to create an expert system. However, the aim was to make accessing legislative acts easier for citizens within Romania.

The study conducted by [9] focused on obtaining appropriate documents and reducing the time spent on reading retrieved documents. To retrieve relevant documents, the authors provided search options that looked at the semantics of words in a search query. To reduce the amount of time spent on reading through documents, the authors visualised the information of the documents by means of semantic networks. Whist this study addressed IR, none of the processes used are applicable for recommending the MAC. The proposed model for recommending the MAC is aimed at processing the documents retrieved by IR, not analysing the semantics of the words used in the search query.

A review on the use of machine learning within the legal domain by [10] revealed that while machine learning can bring about advantages, legal researchers should not completely rely on the results produced by systems that use machine learning. This is because results either overlook important documents or return too many irrelevant documents. Machine learning has also been used to classify legal citations by means of a knowledge base of rules [11]. The rules were used to annotate text within citations at different levels. However, the study by [11] was limited to the amount of data used for testing. Further testing is required on larger datasets. The study conducted by [5] proved to be the most relevant to recommending the MAC, since it facilitates the legal research process by returning relevant legal documents based on a search query. The model uses data repositories, and a vector space IR model to return relevant documents. A model based on [5] can be used as a foundation to recommend the MAC, and be extended to show the adoption of an efficient database. These techniques are discussed in the next section.

B. Techniques for Proposed Model

1) Information Retrieval

IR is a process that deals with the representation, storage, and searching of a collection of data in response to a request from a user [14]. The main goal of IR is to return relevant information that satisfies a user's request. Relevance of the information can be determined by applying two measurements, called *precision* and *recall*. Precision refers to the percentage of retrieved documents that are relevant to the user's query, whilst recall refers to the percentage of documents that are relevant to a query and are retrieved. To help ensure relevance, all IR systems must support three basic processes, namely [14]:

- Representation of a document's content;
- Representation of the user's information need; and
- A comparison of the two above mentioned representations.

IR processes are completed by following five intermediate stages as shown in Fig 1 , namely indexing, filtering, searching, matching, and query-dependent ranking and retrieving [14]. Indexing is the first stage of IR and is the process of representing a document's content by creating a logical view of the document in a collection by means of keywords or terms [15]. To represent a document's content, indexing occurs offline, resulting in an indexed representation of documents. Filtering is done once a query has been entered by a user. During the second stage, filtering removes all stop words and common words from the user's query. Searching is done during the process of representing a user's information need. In the third stage, searching is done through the indexed documents. The fourth stage compares the two representations by matching the documents to the user's query to obtain retrieved

documents. The result of the comparison is a set of ranked retrieved documents in response to the user's query. This form of ranking is known as query-dependent ranking as ranking is done based on the occurrence of a query's terms in documents.

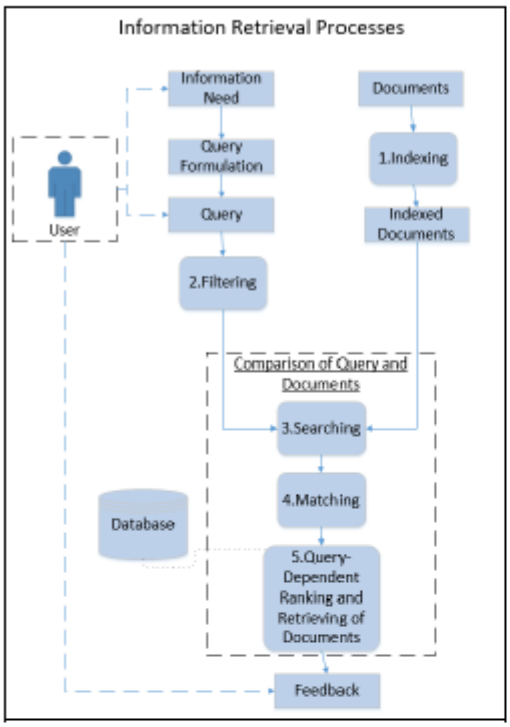


Fig. 1. IR Processes (Author's own construction)

The user then provides feedback if different information is needed by altering the query. Models such as the Boolean and Vector Space Model (VSM) can be used to support IR. The Boolean model is the simplest IR model, is binary, uses Boolean algebra for exact matching and represents documents and queries as sets of terms [16]. The VSM is commonly used, not binary, and represents documents as weighted vectors [16]. Table 2 summarises the two IR models.

IR Model	Advantages	Disadvantages
Boolean Model	Simple; uses Boolean algebra.	No query-dependent ranking; binary.
VSM	Commonly used; not binary.	None.

Table 2 Comparison of IR Models

If an IR system supports the three processes stated by [14], the system can then be customised to a user's specific domain. An IR system can be used in conjunction with other techniques such as IE, which would apply additional filtering to the data that will be saved to databases or parsed using algorithms.

2) Information Extraction

IE is the process of obtaining structured information from unstructured or semi-structured text [17]. Structured text can also be used for IE [18]. Unstructured text contains a variety of text (for example news, blogs or stories) and thus makes extraction difficult. Semi-structured text is presented and formatted in a specific manner for a domain, whilst structured text is highly formatted, structured, and organised. IE consists of three processes, namely [18]:

- Extracting facts;
- Integrating facts; and
- Translating the facts to output.

Extracting facts requires text from a document to be analysed and extracted. Once facts are extracted, the facts are integrated to create a larger set of facts or infer new facts. Facts can then be saved to a database or put through algorithms related to natural language processing (NLP) or ranking to produce output. To support the processes of IE, specific tasks can be performed, depending on the aim of extraction. Fig 2 illustrates the IE processes along with the different IE tasks. Tasks that can be applied are as follows [19]: Named entity recognition (NER); Co-reference resolution (CO); Relation extraction (RE); and Event extraction (EE). NER processes extracted information from unstructured and structured text [18]. When NER is applied, all expressions related to an entity such as organisations, names of people and places, temporal events, and numerical expressions are identified. Once information has been extracted from text, a template can be completed based on the extracted information. The CO task requires identification of multiple mentions of the same entity. At the time of research not much information could be found on CO. RE identifies entities in a body of text and then proceeds to find and classify predefined relationships between the entities. EE parses bodies of text to identify events. Once events are identified, a detailed and structured set of information about the events are derived. Ideally, information such as '*Who did what to whom, when, where, through what methods, and why*' is derived. Two popular techniques namely, NLP and Regular expressions (Regex), can be applied for IE of documents. NLP can be applied to the tasks described by [19] and analyses language to produce meaningful representations [20]. NLP can use different NLP methods such as part-of-speech tagging (POS), chunking, and semantic role labelling [21]. POS labels every word in a set of text with a unique tag to indicate a word's syntactic role. Chunking labels segments of a sentence with syntactic constituents. Semantic role labelling provides a semantic role to a syntactic constituent. Regex are specific text patterns that can be applied to text for IE [22]. Regex can simplify text processing tasks, especially if the text is semi-structured or structured. However, Regex can become hard to understand due to its short syntax and ability to grow in size [23]. Table 3 summarises the two IE techniques.

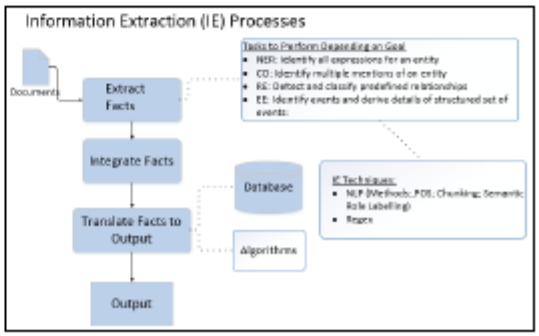


Fig. 2. IE Processes (Author's own construction)

After information has been extracted, the information can be saved to a database and used by relevant algorithms. Algorithms that could be used are machine learning algorithms such as linear regression, clustering, or query-independent algorithms such as the PageRank. The database is a crucial part of IE and the database selected will impact the efficiency and performance of the model. For text that is structured or semi-structured it is recommend to first perform IE on the text then transform the extracted text to a relational database [24]. However, relational databases tend to decrease in efficiency and performance over time especially when the data stored increases. A database that constantly increases in size can cause difficulties when trying to change the database for an organisation's needs due to the multiple joins that are used [25]. An alternative to overcome the issues of relational databases is to use a graph database [26].

IE Technique	Advantages	Disadvantages
NLP	Obtains meaningful representations from text.	Requires different processing steps.
Regex	Identifies patterns; simplifies text processing.	Can become difficult to decipher.

Table 3 Comparison of IE Techniques

3) *Graph Databases*

Various fields such as science, government, and business can be modelled using graphs to understand datasets produced by these fields [27]. The graph space is divided into two parts, namely: *graph compute engines*; and *graph databases*. Graph compute engines are used to analyse large datasets primarily for offline graph analytics. On the other hand, graph databases are graph-orientated databases that consist of one or more graphs to manage and perform complex queries over data [28] and are primarily used for transactional online graph persistence [27]. A graph consists of a set of vertices and edges. Vertices are known as nodes and are connected by edges that define the relationship between nodes [27]. Relationships are a key feature of graph databases as they eliminate the need to infer connections between entities by the use of foreign keys.

Several characteristics make graph databases preferable to traditional relational databases [27] and to outperform them [26] These are: simplicity [27]; performance; flexibility; and agility [29]. Graph databases allow for increased performance due to the data being connected unlike relational databases [27]. Improved performance is seen when executing queries on the graph database. Performance remains constant even when the graph database increases as opposed to a relational database whose join-intensive query performance becomes poor as the database increases in size. In terms of flexibility graphs allow for new data to easily be added whether it be relationships, nodes, labels, or subgraphs without interrupting existing query and application functionality. Elements that have complicated relationships can be easily handled by connecting with edges, eliminating the need for foreign keys. This also results in less migration, overhead, and risk. Relational databases on the other hand require more work. In terms of agility graph databases allow for easy development and maintenance of systems which can be evolved in a controlled manner due to their schema-free nature.

III. THE PROPOSED MAC MODEL

A model was derived from interviews with experts from LegalCo, a literature review, and extant systems analysis. Findings from LegalCo indicated that no formal processes were followed by LegalCo employees when searching for information, particularly for the MAC. Existing systems used by LegalCo only allow employees to search for information and perform brief analysis on legal cases, but no extraction or ranking functions are supported. Furthermore, the experts indicated that large amounts of time are spent searching for the MAC which could delay their overall research time. LegalCo also made data regarding the ALL SA legal journal available for testing purposes [30]. The use of a database is essential to store information and avoid reprocessing of existing data. To recommend the MAC requires information to be ranked. As such, different query-independent ranking algorithms were analysed in an extant systems analysis. These were RAVN ACE [31], Equivo [32], Beagle [33], and eBravia [34], which all performed analysis on legal documents by means of natural language processing. However, none of the systems performed any ranking on the information returned to the user. The MAC Model consists of four components (Fig 3), namely:

1. IR;
2. IE;
3. A graph database; and
4. Query-Independent Ranking.

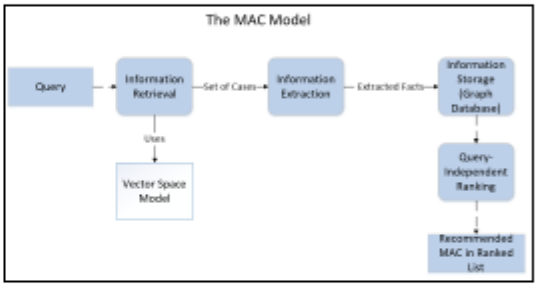


Fig. 3. The MAC Model

In DSR a model uses constructs and methods to present a solution to a problem [35]. In the MAC model the first component, IR, deals with the retrieval of information based on a query received from a user. The query is parsed through an IR model from which a set of ranked results is returned for a user to select. The ranking is known as query-dependent ranking and is based on the retrieval model chosen. In this case a Vector Space Model was chosen as it is the most commonly used IR model [36]. The second component, IE, deals with the extraction of facts from the results that a user selected. Appropriate IE processes and tasks must be applied to legal cases to extract a set of facts. Tasks such as NER can be applied to extract facts. Once facts are extracted the facts can be saved to the third component, a graph database. The fourth component, ranking, performs query-independent ranking of the selected cases and returns a recommendation of the MAC to the user. This round of ranking differs from the ranking performed by the Vector Space Model as it is performed by a query-dependent ranking algorithm such as the PageRank algorithm. The PageRank can be adapted to recommend the MAC or a query could be executed on the nodes within the graph database.

IV. DESIGN AND IMPLEMENTATION

The design and implementation of the graph database formed part of the design cycle of DSR. A simple labelled property graph was used to model the database. When

creating the graph, it is best to think of what questions relating to the data must be asked [27]. In our study, questions such as *'What case is referred to in Case A?' or 'How many cases does Case A refer to?'*. A typical legal case contains basic information about the case as well as a list of all the cases that were referred to. For both iterations nodes were used to represent a case and the referred-to-cases. The aim of the first iteration was to analyse a typical legal case document from the ALL SA legal journal, determine what information would be important and design a conceptual graph model for recommending the MAC. During the second iteration experts from LegalCo were consulted to verify the model and improvements were made based on their feedback. The final labelled property graph is illustrated in Fig 4. The properties for a case node are the case's title, division of the case, date of when the case was heard, and the case's unique number. The properties for referred-to-case (Ref-To-Case) nodes are the referred to case's title, the date on when the case was heard and the legal journal in which the case can be found. Edges are used to represent the decision made by a judge on a referred-to-case node.

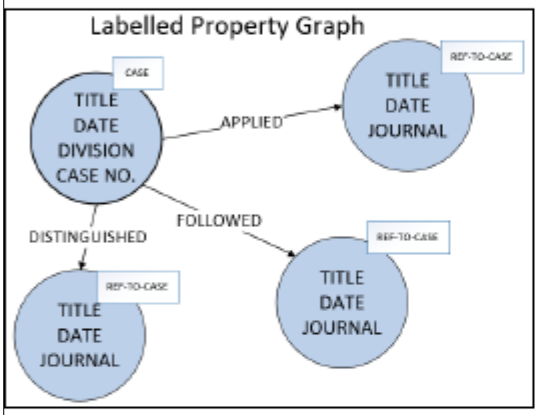


Fig. 4. Labelled Property Graph

Implementation and testing of the graph database was done using Neo4j's library for Python, which is commonly used for graph databases [25][37]. Neo4j is a NoSQL database whose data model is a labelled property graph.

Architectural decisions regarding the graph database such as the type of server to use, clustering, load balancing, testing, capacity planning, and importing and bulk data were considered [27]. However, only two of the architectural decisions proposed by [27] were made, namely, choosing where to host the database (embedded or external) and testing,. For testing purposes, the database was embedded. Unit testing was conducted on parts of the database to ensure that all functions of the graph work accordingly. An embedded database offers various advantages such as lower latency and explicit transactions. Lower latency is experienced as the application communicates directly with the database, therefore eliminating any network overhead. Explicit transactions allow for control of the transactional life cycle by allowing for the execution of complex sequences of commands within the context of a single transaction.

V. FINDINGS

The MAC model was successfully used to guide the requirements analysis, design and implementation of the first prototype of a MAC system. The initial testing of the graph database indicated that it could be used to efficiently extract the MAC. While Neo4j offers a stand-alone tool to directly work within the graph database, the Neo4j library provided for Python was used to implement the graph database. The Python library was adopted because the other components of the model will also be implemented in Python so one development environment is needed. Minor bugs of the Neo4j query language were encountered and not always quick to correct. A useful feature of Neo4j was the immediate visualisation of data inserted into the graph database, which allowed the researcher to see if all the nodes' properties, and relationships were inserted correctly. Insertion of data was quick and no delays were encountered.

VI. CONCLUSION

The purpose of this paper was to present a comprehensive model to aid legal researchers in accessing the MAC for a field of law. The envisaged contribution is a model that addresses all processes required to recommend the MAC. The model can provide a guide to researchers in the field of document retrieval and information asset management, particularly in the legal domain. Various studies within the legal domain were analysed and experts at a legal organisation in South Africa consulted. None of the earlier studies presented a comprehensive model that could recommend the MAC and store data efficiently. Furthermore, the legal experts indicated that there was only a human resource intensive process to find the MAC. The different components and processes required to recommend the MAC using IE were identified from literature. The graph database component of the proposed model was iteratively designed and tested. The study was limited to initial testing by inserting small amounts of data and performing basic queries on the graph database. Additional testing using larger datasets is required as well as testing the different architectural decisions effect on the performance of the graph database. Whilst the study was limited to the legal domain, future research could test the model in other IE domains.

ACKNOWLEDGMENT

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the authors and are not necessarily to be attributed to the NRF.

REFERENCES

[1] G. I. Ifijeh, "Information explosion and University Libraries: Current Trends and Strategies for Intervention," *Chinese Librarianship. An Int. Electron. J.*, vol. Dec. 2010, no. 30, pp. 1–15, 2010.

[2] B. Marr, "How Big Data Is Disrupting Law Firms And The Legal Profession," *Forbes*, 2016. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2016/01/20/how-big-data-is-disrupting-law-firms-and-the-legal-profession/#1c90a1d97c23. [Accessed: 15-May-2017].

[3] N. Evans and J. Price, "Managing information in law firms: Changes and challenges," *Inf. Res.*, vol. 22, no. 1, 2017.

[4] Y. Bhatt and A. Thirunavukkarasu, "Information Management: A Key for Creating Business Value| TDAN.com," 2010. [Online]. Available: tdan.com/information-management-a-key-for-creating-business-value/12829. [Accessed: 09-Apr-2018].

[5] M. Firdhous, "Automating Legal Research through Data Mining," *Int. J. Adv. Comput. Sci. Appl.*, vol. 1, no. 6, pp. 9–16, 2010.

[6] D. Houlihan, "ROSS Intelligence and Artificial Intelligence in Legal Research," Boston, 2017.

[7] M. El Ghosh, H. Naja, H. Abdulrab, and M. Khalil, "Towards a Legal Rule-Based System Grounded on the Integration of Criminal Domain Ontology and Rules," *Procedia Comput. Sci.*, vol. 112, pp. 632–642, 2017.

[8] A.-M. Cornelia, C. I. Murzea, B. Alexandrescu, and A. Repanovici, "Expert Systems with Applications in the Legal Domain," *Procedia Technol.*, vol. 19, pp. 1123–1129, 2015.

[9] R. Giri, Y. Porwal, V. Shukla, P. Chadha, and R. Kaushal, "Approaches for information retrieval in legal documents," in *Tenth International Conference on Contemporary Computing (IC3)*, 2017, pp. 1–6.

[10] D. A. Remus, "The Uncertain Promise of Predictive Coding," *Iowa Law Rev.*, vol. 99, no. 4, pp. 1691–1724, 2014.

[11] F. Galgani and A. Hoffmann, "LEXA : Towards Automatic Legal Citation Classification," in *Advances in Artificial Intelligence 23rd Australasian Joint Conference Adelaide Australia, December 2010 Proceedings*, 2010.

[12] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007.

[13] A. R. Hevner, "A Three Cycle View of Design Science Research," *Scand. J. Inf. Syst.*, vol. 19, no. 2, pp. 87–92, 2007.

[14] A. Roshdi and A. Roohparvar, "Review : Information Retrieval Techniques and Applications," *Int. J. Comput. Networks Commun. Secur.*, vol. 3, no. 9, pp. 373–377, 2015.

[15] S. Ceri, A. Bozzon, M. Brambilla, E. Della Valle, P. Fraternali, and S. Quarteroni, "The Information Retrieval Process," in *Web Information Retrieval*, Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2013, pp. 13–27.

[16] B. Liu, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, 2nd ed. New York, United States: Springer-Verlag New York Inc, 2011.

[17] J. Jiang, "Information Extraction from Text," in *Mining Text Data*, C. Aggarwal and C. Zhai, Eds. Boston: Springer,Boston, MA, 2012, pp. 11–41.

[18] M. Abdelmagid, A. Ahmed, and M. Himmat, "Information Extraction Methods and Extraction Techniques in the Chemical Document â€™ S Contents : Survey," *ARPN J. Eng. Appl. Sci.*, vol. 10, no. 3, pp. 1068–1073, 2015.

[19] J. Piskorski and R. Yangarber, "Information Extraction: Past, Present and Future," in *Multi-source, Multilingual Information Extraction and Summarization*, T. Poibeau, H. Saggion, J. Piskorski, and R. Yangarber, Eds. Berlin: Springer-Verlag Berlin Heidelberg, 2013, pp. 23–50.

[20] G. Chowdhury, "Natural Language Processing," *Annu. Rev. Inf. Sci. Technol.*, vol. 37, no. 1, pp. 51–89, 2003.

[21] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (almost) from Scratch," 2011.

[22] J. Goyvaerts and S. Levithan, *Regular Expressions Cookbook*, 1st ed. Sebastopol, United States: O'Reilly Media, Inc., 2009.

[23] M. Erwig and R. Gopinath, "Explanations for Regular Expressions," in *Proceedings of the 15th international conference on Fundamental Approaches to Software Engineering*, 2012, vol. 7212, pp. 394–408.

[24] R. J. Mooney and R. Bunescu, "Mining knowledge from text using information extraction," *ACM SIGKDD Explor. Newsl.*, vol. 7, no. 1, pp. 3–10, 2005.

[25] A. Perçuku, D. Minkovska, and L. Stoyanova, "Modeling and Processing Big Data of Power Transmission Grid Substation Using Neo4j," *Procedia Comput. Sci.*, vol. 113, pp. 9–16, 2017.

[26] S. Batra and C. Tyagi, "Comparative analysis of Relational and graph databases," *IJSCE Int. J. Soft Comput. Eng.*, vol. 2, no. 2, pp. 509–512, 2012.

[27] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*, 2nd ed. Sebastopol, United States: O'Reilly Media, Inc., 2015.

[28] J. Pokorný, M. Valenta, and J. Kovačič, "Integrity constraints in graph databases," *Procedia Comput. Sci.*, vol. 109, no. 2016, pp. 975–981, 2017.

[29] Z. Zhang, "Graph Databases for Knowledge Management," *IT Prof.*, vol. 19, no. 6, pp. 26–32, 2017.

[30] LexisNexis Editorial Staff, "All South African Law Reports 2017," *Butterworths Law Reports*, 2017. [Online]. Available: https://store.lexisnexis.co.za/products/all-south-african-law-reports-2017-skuZASKUPG1994. [Accessed: 15-Feb-2018].

[31] RAVN Systems, "The Power of Understanding Artificial Intelligence in The Legal World (White Paper)." RAVN Systems, London, England, 2016.

[32] Equivo, "Equivio Zoom: The E-Discovery Platform for Predictive Coding and Analytics." Equivo, Kensington, United States, pp. 1–5, 2012.

[33] Beagle Inc., "Beagle: We sniff out the fine print so you don't have to," 2018. [Online]. Available: https://www.beagle.ai/. [Accessed: 10-Apr-2018].

[34] eBravia, "eBravia: Our Solutions," 2018. [Online]. Available: https://ebrevia.com/#our-solutions. [Accessed: 10-Apr-2018].

[35] S. T. March and V. C. Storey, "Design Science in the Information Sytems Discipline: An Introduction to the Special Issue on Desing Science Research," *MIS Q.*, vol. 32, no. 4, pp. 725–730, 2008.

[36] F. S. Al-Anzi and D. AbuZeina, "Beyond Vector Space Model for Hierarchical Arabic Text Classification: A Markov Chain Approach," *Inf. Process. Manag.*, vol. 54, no. 1, pp. 105–115, 2018.

[37] Neo4j, "Developer: Language Guides - Neo4j Graph Database," 2018. [Online]. Available: https://neo4j.com/developer/language-guides/. [Accessed: 26-May-2018].