Pilot Testing of an Information Extraction (IE) Prototype for Legal Research

Brenda Scholtz
Associate Professor, Faculty of Science, Nelson Mandela University, Port Elizabeth, South Africa
https://orcid.org/0000-0002-0844-1383

Thashen Padayachy
Master's Graduate, Faculty of Science, Nelson Mandela University, Port Elizabeth, South Africa
https://orcid.org/0000-0001-9319-7356

Oluwande Adewoyin
PhD Candidate, Faculty of Science, Nelson Mandela University, Port Elizabeth, South Africa
https://orcid.org/0000-0001-6097-0795

Abstract
This article presents findings from pilot testing of elements of an information extraction (IE) prototype designed to assist legal researchers in engaging with case law databases. The prototype that was piloted seeks to extract, from legal case documents, relevant and accurate information on cases referred to (CRTs) in the source cases. Testing of CRT extraction from 50 source cases resulted in only 38% (n = 19) of the extractions providing an accurate number of CRTs. In respect of the prototype's extraction of CRT attributes (case title, date, journal, and action), none of the 50 extractions produced fully accurate attribute information. The article outlines the prototype, the pilot testing process, and the test findings, and then concludes with a discussion of where the prototype needs to be improved.

Keywords
information retrieval (IR), information extraction (IE), natural language processing (NLP), legal cases, document databases, source cases, cases referred to (CRTs)

Acknowledgement
Financial assistance towards this research from the South African National Research Foundation (NRF) is hereby acknowledged. Opinions expressed and conclusions arrived at are those of the authors and are not to be attributed to the NRF.

DOI: https://doi.org/10.23962/10539/29192

Recommended citation
Scholtz, B., Padayachy, T., & Adewoyin, O. (2020). Pilot testing of an information extraction (IE) prototype for legal research. *The African Journal of Information and Communication (AJIC), 25*, 1-20. https://doi.org/10.23962/10539/29192

1. Introduction

The research we describe in this article—the pilot testing of an information extraction (IE) prototype designed to assist legal researchers—is the continuation of a research project that was outlined in an earlier publication. As outlined in that Padayachy, Scholtz and Wesson (2018) paper, the ultimate aim of the research project is to develop an information extraction (IE) model that, when applied to a database of legal cases, can (1) determine the case most applicable to a point of law; and (2) store the findings in a database.

The Padayachy et al. (2018) paper described the process of designing the IE model. Following a design science research (DSR) framework, the model was based on a literature review, a review of existing systems at a sample firm, and interviews with experts. The 2018 paper also described results from the initial testing of the prototype with small amounts of data and using basic queries in a graph database. In this article, we describe and discuss the results of the next phase of testing of the IE prototype, using a larger pilot sample (50 legal case documents) and testing two of the prototype's four processes: information extraction (IE) and information storage. The legal documents used were provided to us by an organisation in the South African legal domain (which we call LegalCo, for anonymity purposes).

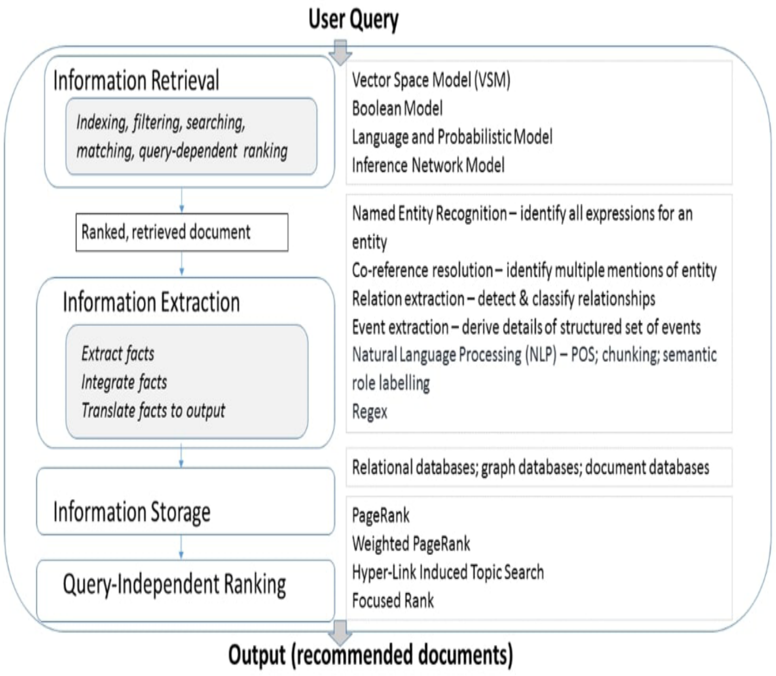
2. The IE prototype

Our model, which we call the IE Model for Legal Cases, consists of the following four main processes:

- information retrieval (IR), with indexing;
- information extraction (IE);
- information storage; and
- query-independent ranking of the most applicable case (MAC).

Figure 1 provides detail on the elements assembled in order to actualise the four processes, and to integrate the processes in a manner designed to produce relevant case information for a legal researcher. The model also incorporates the options of using a document database, a graph database, and/or a relational database for document storage.

Figure 1: IE Model for Legal Cases


User Query
Information Retrieval
Indexing, filtering, searching, matching, query-dependent ranking
Vector Space Model (VSM)
Boolean Model
Language and Probabilistic Model
Inference Network Model
Ranked, retrieved document
Named Entity Recognition – identify all expressions for an entity
Co-reference resolution – identify multiple mentions of entity
Relation extraction – detect & classify relationships
Event extraction – derive details of structured set of events
Natural Language Processing (NLP) – POS; chunking; semantic role labelling
Regex
Information Extraction
Extract facts
Integrate facts
Translate facts to output
Relational databases; graph databases; document databases
Information Storage
PageRank
Weighted PageRank
Hyper-Link Induced Topic Search
Focused Rank
Query-Independent Ranking
Output (recommended documents)

Information retrieval (IR)

The first set of processes that needs to take place in our model, based on a query from the user, is IR, which is the process that deals with the representation, storage, and search of a collection of data in response to a request (Roshdi & Roohparvar, 2015). The data could be in the form of video, audio, or text. For our model, text is the focus, with the assumption that information compilation for legal cases is currently, for the most part, in textual form. An IR system does its job by recovering relevant documents from a compilation of resources made by autonomous modules or a database (Kumar & Sharma, 2018).

As illustrated in Figure 1, *indexing* is the first process in IR. Indexing 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 (Kumar & Sharma, 2018). An index is constructed from a case's keywords in order to act as a pointer to a stored document for rapid, accurate retrieval and storage. *Filtering*, also represented in Figure 1, is done once a query has been entered by a user. Filtering removes all stop words from the user's query. (A stop word is a word that is very commonly used and that is programmatically ignored by a search system during indexing, search, and retrieval of information (Schofield, Magnusson, & Mimno, 2017).) *Searching* is then done on the indexed documents. *Matching* compares the two representations (i.e., the indexed documents and the user's information need) so that *query-dependent ranking* and *ranked retrieval* of documents can take place. The output of this process is a set of ranked, retrieved documents. The user can then provide feedback, if different information is needed, by altering the query.

For text retrieval, four matching techniques are deployed, using the following four models:

- the vector space model (VSM);
- the Boolean model;
- the language and probabilistic model; and
- the inference network model.

The VSM recovers text automatically, through representation of documents and queries as weighted vectors. The VSM was successfully used in the study by Firdhous (2010) to retrieve legal documents based on user queries. The VSM was also used, by Aritomo and Watanabe (2019), to generate a searchable encryption technique that enabled a keyword search for documents through encryption. (However, the technique was applied to request-for-comment documents and not legal case documents.) Irrespective of the domain, the VSM is the most commonly used IR model (Al-Anzi & AbuZeina, 2018). Frequency–inverse document frequency (tf–idf or TFIDF for short) can be used in VSM for weight estimation; it is a numerical statistic that is used to reflect the importance or weighting of a word to a document in a collection. Retrieval is based on the degree of similarity between the term vector and the query vector, while recovered results are ranked using cosine similarity (a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them), which is a major strength of the VSM. However, the VSM suffers from the inability of the vector space to deal with polysemy and synonymy. Also, index terms are assumed to be mutually independent (Kumar & Sharma, 2018).

The Boolean model is the simplest matching technique for text retrieval. The model uses Boolean algebra for exact matching, and represents documents and queries as sets of terms (Liu, 2011). Results of the Boolean model can be true or false, with queries indicated using OR, XOR and AND relators. Application of the Boolean model is characterised by a large assembly of terms, and information is outputted if all the Boolean terms are present in the resource—but the Boolean model is limited by its requirement for strict matching that generates "nothing or too many" problems (Pandey, Mathur, & Joshi, 2019). Also, the model does not deal with frequency and term weights, leading to unranked results. Therefore, users must have a strong knowledge of query-making.

The language and probabilistic model recovers documents or text by placing emphasis on the probabilities of different factors in decision-making, such as documents' relevance, for ranking (Losee, Bookstein, & Yu, 1986; Losee, 2015). However, the model is affected by the difficulties of combining different ranking functions into a single function, ranking speed, the non-convex nature of ranking algorithms, and retrieval of irrelevant information (Pandey et al., 2019).

The inference network model consists of a directed, acyclic graph that contains 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, meaning that true and false are the only two possible outcomes.

Information extraction (IE)

The output of the IR processes (the ranked documents) can then be used as inputs to the IE process. In the IE process, additional filtering is applied to the ranked data and the results are saved to a data repository. From the data repository, the saved results can then be parsed using algorithms such as those that perform clustering or query-independent ranking (e.g., Google PageRank). The main techniques used in IE are:

- named entity recognition (NER) (Abdelmagid et al., 2015; Piskorski & Yangarber, 2013);
- co-reference resolution (Iida, Inui, & Matsumoto, 2003; Piskorski & Yangarber, 2013);
- relation extraction (RE) (Piskorski & Yangarber, 2013);
- event extraction (EE) (Piskorski & Yangarber, 2013);
- natural language processing (NLP) (Piskorski & Yangarber, 2013); and
- regular expressions (regex) usage (Goyvaerts & Levithan, 2009).

NER is an IE technique that processes extracted information from unstructured and structured text (Abdelmagid et al., 2015). When the technique is applied, all the expressions related to an entity are identified. In addition, NER can involve extracting descriptive information from text about an entity, and completing a template based on the extracted information. Two main tasks are involved in NER: identification and classification of predefined entities. Piskorski and Yangarber (2013) identify organisations, persons, temporal expressions, and numerical expressions as examples of predefined entities.

Co-reference resolution requires the identification of multiple mentions of the same entity (Piskorski & Yangarber, 2013). These mentions can be named, pronominal, nominal, or implicit. A named mention refers to an entity by name, e.g., "General Electric", while a pronominal mention refers to an entity by use of a pronoun, e.g., "*he* forgot to buy food". A nominal mention refers to an entity by a noun phrase, e.g., "*The company* unveiled future plans". 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., 2003). 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 is a technique for the detection and classification of predefined relationships between entities identified in a body of text (Piskorski & Yangarber, 2013), e.g., *PersonA is an employee of PersonB.*

EE looks for events in the text and derives detailed and structured sets of information about the events. It is said to be one of the most difficult IE tasks, as it needs to extract the information necessary to answer the questions "who did what to whom, when, where, through what methods?"

NLP can be used to analyse and produce meaning from text that has been extracted from sources such as documents or websites (Singh, 2018). 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 fact extraction process. In the context of our interest in ultimately developing a prototype that can rank and recommending the MAC, NLP can be applied to text that has been extracted from legal cases. Common NLP techniques are part-of-speech (POS) tagging, stop-word removal, parsing, chunking, NER, and semantic role labelling (Chopra, Prashar, & Sain, 2013; Collobert et al., 2011; Vijayarani et al., 2015).

With POS tagging, each word in a set of text is labelled with a unique tag to indicate the word's syntactic role. Words are labelled based on English parts of speech 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). 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. Chunking, also known as shallow parsing, labels segments of a sentence with syntactic constituents such as noun or verb phrases (Collobert et al., 2011). In the context of NLP, NER involves labelling elements in a sentence according to different categories, e.g., "person", "location". Semantic role labelling assigns semantic roles to syntactic constituents of a sentence (Collobert et al., 2011).

In regex usage, specific text patterns are used for searching bodies of text, replacing text, segregating text into smaller bodies, and rearranging pieces of text (Goyvaerts & Levithan, 2009). Regex, if correctly used, can simplify programs and text processing tasks by minimising the amount of code needed for processing. Regex usage differs from NLP as none of the NLP phases need to be applied when using regex with bodies of text, i.e., regex can be used directly on an unprocessed body of text.

Information storage

The information storage element is a crucial part of IE, and the data repository selected impacts the efficiency and performance of the model. For text that is structured or semi-structured, Mooney and Bunescu (2005) recommend that IE is performed first on the text, and then the extracted text is transformed to a relational database. However, relational databases tend to decrease in efficiency and performance over time, especially when the data stored increases. An alternative, to overcome the inefficiencies of relational databases, is to use a graph database (Batra & Tyagi, 2012) or a document database (Roy-Hubara & Sturm, 2020). A graph database uses graphs to store information in nodes and allows for the creation of relationships between nodes using edges (Batra & Tyagi, 2012). Among the benefits of graph databases are performance and flexibility. A document database is a non-relational database that stores data in the form of documents that can be grouped together to form collections (Moniruzzaman & Hossain, 2013). 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 to 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).

Query-independent ranking of selected cases

The information storage process needs to apply techniques and algorithms to save the extracted facts into a data repository so that the last process—query-independent ranking of the selected cases—can be performed. In our model, the query-independent ranking process will return a recommendation of the MAC to the user. For this process, the user of our model will either use an adapted version of Google PageRank or will execute a query on the nodes within the database.

3. Testing of the IE prototype

For our pilot testing of two of our model's components—IE and information storage—the company that we refer to as LegalCo provided the sample data, which consisted of legal case files from the *All South African Law Reports* (LexisNexis South Africa, n.d.).

Legal cases can be separated into two parts:

- *the source case* and its primary attributes (case name, case division, case date); and
- *cases referred to*—which we have given the acronym CRTs—with each CRT having four attributes: title, journal, action, and date.

Each source case can have many CRTs.

The main problem addressed by our pilot testing was how to accurately retrieve CRT information from source case files. The prototype we tested in this pilot study did not include the initial step of IR or the final step of query-independent ranking, i.e., determining the MAC. The requirements for the prototype for this study were therefore only those relating to the IE and information storage steps, as summarised in Table 1. The Natural Language Toolkit (NLTK) was used to implement NLP for IE, since it supports most NLP tasks.

Table 1: Techniques and software used for the prototype's IE and information storage components

Processes	Functionality and techniques used	Software
information extraction (IE)	Extract the XML contents of a .docx formatted legal case	Zipfile, Python library
	Parse legal cases in .docx (MS Word) format (NLP, regular expressions, tokenisation, and stop-word removal)	NLTK RE, LXML, Python library
IE and information storage	Develop the prototype	Pycharm IDE
information storage	Interact with the Neo4j graph database	Neo4j, Python library
	Set up document database	MongoDB
	Manage document database	MongoDB Compass desktop application

Table 2 provides an overview of the iterations/equations and experiments we conducted using the prototype.

Table 2: Iterations and experiments

Iteration	Measures	Experiment Set	Number of Source Cases
1: Difference ratio for a source case	Basic functionality, error checking and effectiveness	1	10
2: Total difference ratio for CRTs	Accuracy and execution time	2	10
		3	50, 102

Iteration 1: Extraction of facts from a legal source case

The aim of iteration 1 (not discussed in detail in this article) was to analyse a typical legal case document from the *All South African Law Reports*, determine what information would be important, and test the accuracy of the processes and techniques applied (as specified in Table 2). In this iteration, the CRTs were not considered, since this was the more complex requirement. In the set of experiments conducted in this iteration, 10 legal cases from the LegalCo database—all cases from the *All South African Law Reports*—were used as a test set (coded as cases U1 to U10). In this iteration the experiments used a graph database first and then a document database. Use of the graph database allowed for the extracted facts to be stored in nodes that were connected to each other by specific relationships. Use of the document database allowed for the extracted facts to be stored as documents with multiple types of data embedded into a single document.

Iteration 2: CRT extraction difference ratio

The aim of iteration 2 (the focus of this article) was to test the accuracy of the prototype's extraction of CRTs from source cases. The equation used to determine the *CRT extraction difference ratio for a single source case* was as follows:

Equation 1

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

Where:
Xi = the CRT extraction difference ratio for source case *i*;
Ai = the CRT output frequency count for source case *i* that differs from the expected CRT output for source case *i*; and
Bi = the expected CRT output frequency count for source case *i*.

The equation used to determine the *total CRT extraction difference ratio for multiple source cases* was as follows:

Equation 2

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

Where:
Y = total CRT extraction difference ratio for multiple source cases;
Ai = the CRT output frequency count for source case *i* that differs from the expected CRT output for source case i; and
Bi = the expected CRT output frequency count for source case *i*.

An ideal value for the difference ratio is 0, since this indicates that there is no difference between the actual output and the expected output (Conroy, 2016). In our experiments, as described in the next section of this article, an error margin of 10% was applied, implying that a difference ratio of 0.1 or less was considered acceptable.

4. Findings

Iteration 1: Experiment set 1

Both a graph database and a document database were investigated and tested during experiment set 1. The graph database implementation 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 dummy data consisted of recipes and drinks associated with each recipe. The insertion of the data was successful. The nodes were the recipes and the relationships were the associated relationships with one or more drinks. However, errors were encountered when trying to create the nodes with relationships. Multiple attempts were made to resolve this error, but no solution could be found. This led to iteration 2, experiment set 2, where a document database was used successfully, as described in the next section.

Iteration 2: Experiment set 2

Experiment set 2 used the document database MongoDB (MongoDB, n.d.) and 10 source cases (coded as cases U1 to U10). In a document database, the data is stored as key-value pairs where both the keys and values are searchable. For experiment set 2, the extracted data was converted to a Python dictionary, whereby data values could be associated with keys, thus allowing for the keys and values to be searchable.

It was found that the primary attributes from the documents were correctly extracted after some cleaning, but the CRTs were not always extracted correctly. U1 to U10 had a total CRT extraction difference ratio of 0.006, implying that 0.6% of the number of observed CRTs in the source cases were different from what was expected. Since a difference ratio of 0.1 or less was considered acceptable, the result of 0.006 was therefore an acceptable value. (The sample size of 10 was small, and thus only appropriate for a pilot study of the kind we were conducting.)

Some of the CRTs that had no actions were in different formats, which could account for why they were not extracted. The use of .docx-formatted documents was found to be better than .pdf-formatted documents, since the paragraph tags that store text in XML could, in the .docx-formatted (MS Word) documents, be accessed and processed. All subsequent testing was thus conducted using MS Word documents, and this was one limitation of this study.

Iteration 2: Experiment set 3

The first part of experiment set 3 used 50 source cases (F1 to F50) from South African case law during the period 1996 to 2018. The aim of this first part was to determine the prototype's extraction accuracy for extracting the number of CRTs. The total CRT extraction difference ratio, determined through the application of equation 2, was used as the measure of accuracy. Table 3 provides the frequency count results for each of the 50 source cases.

Table 3: CRTs extracted

Case	Actual no. of CRTs in the source case	No. of CRTs extracted from the source case	Absolute difference (excess or shortfall in no. of CRTs extracted)	Difference ratio for CRTs (difference between extracted no. and actual no.)
F1	45	46	1	0.02
F2	5	5	0	0.00
F3	42	43	1	0.02
F4	5	4	1	0.20
F5	6	6	0	0.00
F6	2	2	0	0.00
F7	15	11	4	0.27
F8	7	8	1	0.14
F9	5	5	0	0.00
F10	1	2	1	1.00
F11	7	8	1	0.14
F12	3	3	0	0.00
F13	15	18	3	0.20
F14	9	10	1	0.11

F15	33	33	0	0.00
F16	5	4	1	0.20
F17	6	8	2	0.33
F18	23	23	0	0.00
F19	8	4	4	0.50
F20	10	25	15	1.50
F21	10	10	0	0.00
F22	11	27	16	1.45
F23	23	24	1	0.04
F24	10	19	9	0.90
F25	23	19	4	0.17
F26	26	25	1	0.04
F27	12	13	1	0.08
F28	6	6	0	0.00
F29	15	15	0	0.00
F30	15	19	4	0.27
F31	18	19	1	0.06
F32	11	12	1	0.09
F33	6	6	0	0.00
F34	7	9	2	0.29
F35	10	11	1	0.10
F36	17	17	0	0.00
F37	27	21	6	0.22
F38	10	10	0	0.00
F39	75	60	15	0.20
F40	5	5	0	0.00
F41	6	6	0	0.00
F42	8	8	0	0.00
F43	13	13	0	0.00
F44	13	13	0	0.00
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	0.00
Totals	697	731	110	9.46
Average				0.19

Table 4 provides a summary of the accuracy results for the 50 cases according to the frequency, and frequency percentage per range of difference ratios, calculated for the CRT extraction numbers. As can be seen in the table, of the 50 extractions (from the 50 source cases used in the test), 38% (n = 19) had a difference ratio of 0, i.e., in only 38% of the extractions, the number of CRTs extracted was the same as the actual number of CRTs in the source case.

As also shown in Table 4, an overall difference ratio of 0.1578 was observed across the 50 source cases, i.e., the number of CRTs extracted was, on average, 16% different from the actual number in the source case. This difference ratio was higher than the 10% margin of error specified (i.e., in the first three rows in Table 4). Based on the 10% margin of error, 54% (n = 27) of the CRT extraction numbers fell within the margin of error, and 44% (n = 22) of the CRT extraction numbers fell outside the margin of error, thus highlighting the complexities involved in information extraction of this nature.

Table 4: Difference ratio ranges for CRT extractions

Difference ratio range	**Frequency (n = 50)**	**Frequency %**		
0	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		
Totals	**50**	**100**	**Total difference ratio**	**0.157819**

Extraction accuracy for CRT attributes

To analyse the accuracy of the extraction of CRT attributes (title, date, journal, and action), we assigned the following categorisations:

- perfect;
- partial; and
- not (extracted).

For example, if a source case had five CRTs and all five had perfectly extracted attributes, then the extraction would have been classified as "perfect". If only some of the five CRTs were perfectly extracted, and others were only partially extracted, the extraction was categorised as "partial". If none of the five CRTs was extracted, the extraction was categorised as "not" extracted. Table 5 provides a summary of the count of cases that were categorised as being perfect, partial, not, and noisy. Among the extractions of CRT attributes from the 50 source cases, none were found to be "perfect"; 96% (n = 48) were categorised as "partial"; and 4% (n = 2) were categorised as "noisy". We regarded the "partial" extractions as satisfactory for the purposes of this pilot phase, but with further study needed into the causes of incomplete extractions.

Table 5: Extraction accuracy for CRT attributes

Categories	Frequency (n = 50)
perfect	0
partial	48
not	0
noisy	2

For the 48 partial extractions and 2 noisy extractions, further investigation was conducted to determine the cause. Three main causal categories were identified:

- incorrect extraction of extra lines;
- incorrect splitting of CRTs; and
- noisy CRT data.

"Extra lines" errors occurred when non-CRT lines of text, which were similarly formatted to CRT text, were incorrectly detected and extracted as part of a CRT. "Splits" errors occurred when a CRT was incorrectly split into two parts, resulting in a false additional CRT being extracted. "Noise" errors occurred when CRTs had noisy data. (Noisy data is data that negatively affects data processing techniques (Quinlan, 1986).)

Table 6 provides the extraction accuracy results for the CRTs' attributes (title, date, journal, and action). As shown in the table 27% (n = 196) of CRT titles were perfectly extracted, while 48% (n = 353) were partially extracted, and 14% (n = 99) were not extracted. In respect of CRT dates, 83% (n = 604) were perfectly extracted, none was partially extracted, and 6% (n = 44) were not extracted. In respect of CRT journals, 72% (n = 531) were perfectly extracted, fewer than 1% (n = 4) were partially extracted, and 15% (n = 113) were not extracted. In respect of CRT actions, 40% (n = 292) were perfectly extracted, fewer than 1% (n = 3) were partially extracted, and 48% (n = 353) were not extracted.

Table 6: Extraction accuracy for CRT attributes

Attribute	Category	Frequency (n = 731)
CRT title	Perfectly extracted CRT titles	196
	Partially extracted CRT titles	353
	Titles not extracted	99
	Extra line instances	25
	Split instances	19
	Noisy instances	39
CRT date	Perfectly extracted CRT dates	604
	Partially extracted CRT dates	0
	CRT dates not extracted	44
	Extra line instances	25
	Split instances	19
	Noisy instances	39
CRT journal	Perfectly extracted CRT journals	531
	Partially extracted CRT journals	4
	CRT journals not extracted	113
	Extra line instances	25
	Split instances	19
	Noisy instances	39
CRT action	Perfectly extracted CRT actions	292
	Partially extracted CRT actions	3
	CRT actions not extracted	353
	Extra line instances	25
	Split instances	19
	Noise instances	39

None of the 50 extractions could earn an overall "perfect" categorisation (see Table 5 above) because each extraction had at least one CRT attribute error, i.e., an error in extraction of one or more of the CRT's title, date, journal, or action.

Table 7 summarises the difference ratios for the perfectly extracted attributes. For CRT titles, a difference ratio of 0.73 was observed, indicating that 73% of the extracted CRT titles were different from the actual titles. For CRT dates, 17% of the extractions were different from the actual dates; for CRT journals, 27% of the extractions were different from the actual journals; and for CRT actions, 60% were different from the actual actions.

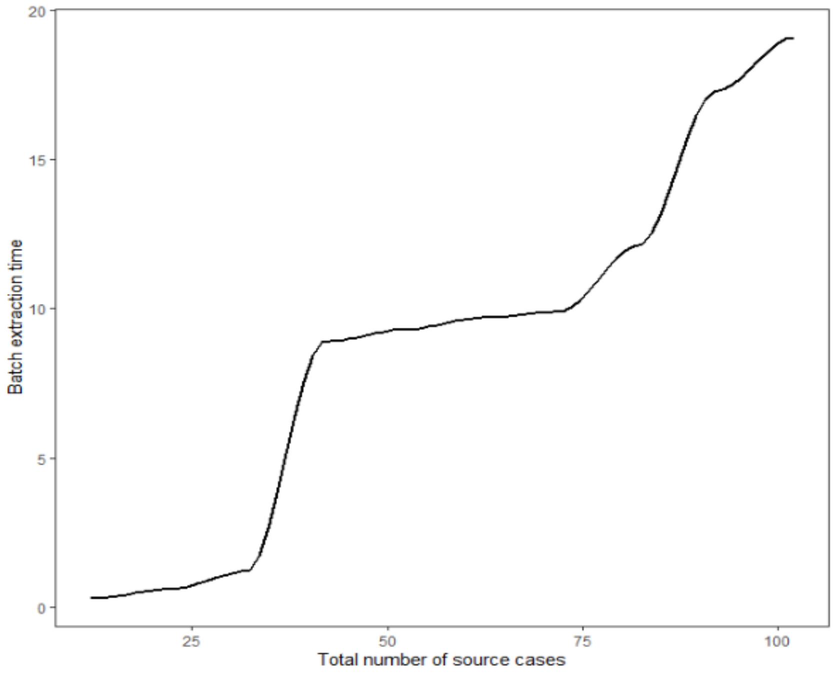
Table 7: Difference ratios for perfectly extracted CRT attributes

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 actions	292	0.60

Execution times

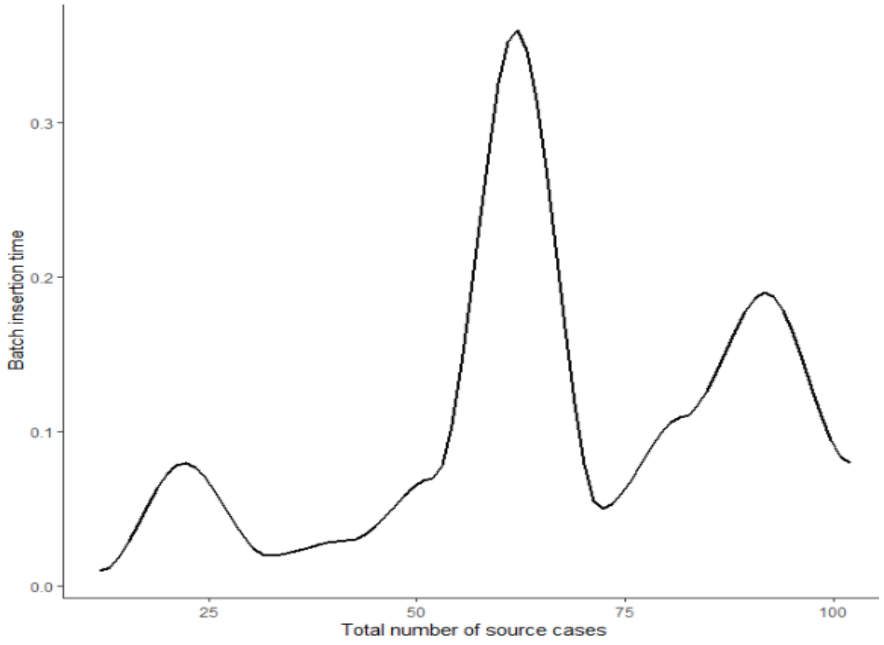
In the second part of experiment set 3, aimed at evaluating the prototype's information storage performance, we evaluated the prototype's execution times. Two metrics for execution time were used, namely extraction time and insertion time (the time taken to insert the source cases in the batch into the database as objects). Ten batches of tests were run, and the extraction and insertion results are illustrated in Figure 2 and Figure 3, respectively. (The extraction process we timed was for the extraction of source cases, not for the extraction of CRTs.) The first test batch had 12 case documents. In each successive test, an additional 10 cases were added, until reaching the last batch of 102 case documents.

Figure 2: Time taken to extract source cases



For the 10 batches, an average time of 8.8 seconds per batch was taken to extract the source cases. The batch extraction time was found to be satisfactory, at 1.2 seconds, for the batch of 32 cases, but the batch extraction time jumped to 8.9 seconds for the 42-case batch. A likely reason for this marked increase in extraction processing time was the increased word counts of the case documents.

Figure 3: Time taken to insert case objects



The insertion time was calculated in terms of the time it took to insert the case objects, as key-value pairs, into the document database. The insertions were found to be quick, taking an average of 0.10 seconds per source case object (n = 102). But no pattern could be found for the insertion times, as the times were not found to increase accumulatively as the size of the batch increased. The reasons for this were not clear, and could be investigated in future research. The longest time taken to insert case objects was for the batch of 62 case objects, which took 0.36 seconds per case. The shortest time taken was 0.01 seconds per case (n = 12).

5. Conclusions

The pilot testing described in this article, of a prototype for two components—the IE process and the information storage process—in our proposed IE Model for Legal Cases, highlights the challenges that need to be addressed in order to accurately and efficiently extract and store CRTs. Testing of our prototype's CRT extraction from 50 source cases resulted in only 38% (n = 19) of the extractions providing an accurate number of CRTs. None of the 50 extractions resulted in fully accurate extractions of the CRT title, date, journal, and action attributes. It is thus clear that the IE prototype needs to be improved. A key area requiring improvement is the prototype's ability to extract CRT information from a wider variety of case formats, in order to reduce data noise and, in turn, the number of errors caused by the splitting of a single CRT's information into more than one CRT extraction, and the number of errors caused by the addition of non-CRT-related text into a CRT's extraction.

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.

Al-Anzi, F. S., & AbuZeina, D. (2018). Beyond vector space model for hierarchical Arabic text classification: A Markov chain approach. *Information Processing & Management, 54*(1), 105–115. https://doi.org/10.1016/j.ipm.2017.10.003

Aritomo, D., & Watanabe, C. (2019). Achieving efficient similar document search over encrypted data on the cloud. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)* (pp. 1–6). https://doi.org/10.1109/smartcomp.2019.00020

Batra, S., & Tyagi, C. (2012). Comparative analysis of relational and graph databases. *International Journal of Soft Computing and Engineering (IJSCE)*, *2*(2), 509–512.

Chopra, A., Prashar, A., & Sain, C. (2013). Natural language processing. *International Journal of Technology Enhancements and Emerging Engineering Research*, *1*(4), 131–134.

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

Conroy, R. (2016). *Sample size: A rough guide.* Dublin: Royal College of Surgeons in Ireland. http://doi.org/10.1080/08897077.2011.640215

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

Croft, W. B., Metzler, D., & Strohman, T. (2015). *Information retrieval in practice*. New York: Pearson.

Firdhous, M. (2010). Automating legal research through data mining. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *1*(6), 9–16.

Goyvaerts, J., & Levithan, S. (2009). *Regular expressions cookbook*. Boston: O'Reilly Media.

Iida, R., Inui, K., Takamura, H., & Matsumoto, Y. (2003). Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the 2003 EACL Workshop on the Computational Treatment of Anaphora* (pp. 23–30).

Indurkhya, N., & Damerau, F. J. (Eds.).(2010). *Handbook of natural language processing (Vol. 2)*. Boca Raton, FL: CRC Press.

Kumar, R., & Sharma, S. C. (2018). Information retrieval system: An overview, issues, and challenges. *International Journal of Technology Diffusion (IJTD)*, *9*(1), 1–10.

LexisNexis South Africa. (n.d.). *All South African law reports*. Retrieved from https://store.lexisnexis.co.za/products/all-south-african-law-reports-2020-skuZASKUPG1994

Liddy, E. D. (2001). Natural language processing. In M. A. Drake (Ed.), *Encyclopedia of library and information science* (2nd ed.). New York: Marcel Dekker. https://doi.org/10.1017/S0267190500001446

Liu, B. (2011). *Web data mining: Exploring hyperlinks, contents, and usage data*. New York: Springer-Verlag.

Losee, R. M. (2015). Validating a model predicting retrieval ordering performance with statistically dependent binary features. *International Journal of Information Retrieval Research (IJIRR)*, *5*(1), 1–18. https://doi.org/10.4018/ijirr.2015010101

Losee, R. M., Bookstein, A., & Yu, C. T. (1986). Probabilistic models for document retrieval: A comparison of performance on experimental and synthetic databases. In *SIGIR '86: Proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 258–264). https://doi.org/10.1145/253168.253222

MongoDB. (n.d.). Introduction to MongoDB: Documents. Retrieved from https://docs.mongodb.com/manual/core/document/

Moniruzzaman, A., & Hossain, S. (2013). NoSQL database: New era of databases for big data analytics – Classification, characteristics and comparison. *International Journal of Database Theory and Application*, *6*(4), 43–45. https://doi.org/10.1016/S0262-4079(12)63205-9

Mooney, R. J., & Bunescu, R. (2005). Mining knowledge from text using information extraction. *ACM SIGKDD Explorations Newsletter*, *7*(1), 3–10. https://doi.org/10.1145/1089815.1089817

Padayachy, T., Scholtz, B., & Wesson, J. (2018). An information extraction model using a graph database to recommend the most applied case. In *Proceedings of the 2018 International Conference on Computing, Electronics and Communications Engineering (ICCECE)* (pp. 89–94). doi: 10.1109/iCCECOME.2018.8658659

Pandey, S., Mathur, I., & Joshi, N. (2019). Information retrieval ranking using machine learning techniques. In *2019 Amity International Conference on Artificial Intelligence (AICAI)* (pp. 86–92). https://doi: 10.1109/AICAI.2019.8701391

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*. Berlin: Springer. https://doi.org/10.1007/978-3-642-28569-1_2

Quinlan, J. R. (1986). The effect of noise on concept learning. In R. S. I. Michalski, J. G. Carboneel, & T. M. Mitchell (Eds.), *Machine learning*. Burlington, MA: Morgan Kaufmann Publishers.

Roshdi, A., & Roohparvar, A. (2015). Review: Information retrieval techniques and applications. *International Journal of Computer Networks and Communications Security*, *3*(9), 373–377.

Roy-Hubara, N., & Sturm, A. (2020). Design methods for the new database era: A systematic literature review. *Software & Systems Modeling*, *19*, 297–312. https://doi.org/10.1007/s10270-019-00739-8

Schofield, A., Magnusson, M., & Mimno, D. (2017). Pulling out the stops: Rethinking stopword removal for topic models. In *15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (pp. 432–436). https://doi.org/10.18653/v1/e17-2069

Singh, S. (2018). Natural language processing for information extraction. arXiv preprint arXiv:1807.02383.

Vijayarani, S., Ilamathi, M. J., & Nithya, M. (2015). Preprocessing techniques for text mining – An overview. *International Journal of Computer Science & Communication Networks*, *5*(1), 7–16.