Universiteit Leiden

ICT in Business and the Public Sector

The Creation and Integration of a Legal Information Retrieval System without Manual Query Construction

Name: Wilco Draijer

Date: January 31, 2019

1st supervisor: S. Verberne
2nd supervisor: K.C.M. Nuijten

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgements

First, I would like to express my gratitude to my first supervisor Dr. Suzan Verberne for her guidance, insights and expertise. Her knowledge of the text mining field helped enormously with building a valid system and writing a solid thesis.
I am grateful to Koos Nuijten, my second supervisor, for his insight in the business component of my research. This was very valuable to create an actual ICT in Business thesis (and not just ICT).
I also want to thank Ivo Fugers, my manager at Ortec where I did my research. He has been a great advisor during my research, on both the theoretical and practical side.
Special thanks to the organizers of COLIEE (Ken Satoh, Randy Goebel, Mi-Young Kim, Yoshinobu Kano, Masaharu Yoshioka, Juliano Rabelo) and JURISIN (Katsumi Nitta and all commitee members) for the data used in this research and the opportunity to present my research in Yokohama, Japan. The trip and conference was an amazing experience.
Thanks to all lawyers and legal employees I could interview for this research.
Above all, I want to thank God for the strength and guidance He has given me.

Abstract

Searching precedence is an error-prone and time-intensive task in the legal field. The query construction is difficult since too few search terms result in an abundance of results, but too many increase the chance of missing an important piece. This research follows the question: How should a precedent retrieval system without manual query construction be developed?
A queryless system is researched using entire documents as an input. First the best performing scoring functions are tested, followed by research on the best preprocessing steps using the scoring functions from the first step.
The More Like This query from Elasticsearch and TF–IDF perform better as scoring functions than doc2vec and LDA. The preprocessing research shows that basic preprocessing steps perform well and that data-specific steps could increase performance. These results are a good starting point for creating a legal search system.
To research the integration of a legal information retrieval into the legal field lawyers are interviewed to discuss legal search currently, to propose a queryless system with visualizations and retrieve responses to this proposal, and to retrieve suggestions for improvement. The interviews resulted in a clear view on how necessary a well functioning legal search system is, and in many suggestions for development and design choices to integrate the system more easily into the field.
To finalize the research described in this thesis, suggestions are proposed on how to improve a legal search engine both in usability and functional terms.

Contents

Chapter 1

Introduction

A precedent is "a case or issue decided by a court that can be used to help answer future legal questions[1], which can hold or be used as a valuable argument in a new case". For example in the case ECLI:NL:RBAMS:2016:2573[2], a Dutch case in the court of Amsterdam, where an appeal on disproportion fails since the court decided in case ECLI:NL:RBAMS:2013:BZ3203[3] that an appeal like that can only succeed under very exceptional circumstances, which there were not. Or in case ECLI:NL:RBAMS:2018:5226[4] where there is referred to two other cases[5][6] to show that when one of the components of the offense is missing in the indictment, the proven facts are not punishable and therefore the case should be dismissed.

Cases like this where previous judgments influence or decide current rulings are not unusual. As current rulings are regularly influenced or decided by previous judgments, it is important for lawyers and other employees in the legal field to find prior cases with similar characteristics when a new case arrives. With the big increase in available digital legal documents it becomes more difficult and time consuming to find (enough) relevant cases[7]. Currently there are two main legal search engines used in the Netherlands: Legal Intelligence and Rechtsorde[8]. These engines search through judgments, case law and and other published legal documents. The search engines match a query given by the user and rank all documents according to similarity of documents to query. To use these engines a lawyer will think of search terms based on the information of a new case and use these to create a query. When the results of this query are not to his/her liking, he/she will alter the query, either by adding or removing words, or by entering synonyms.
The downside of this query-driven process is that the query construction is very complex and therefore error prone. This query construction can be avoided by using information from the new case as input. This information can be some facts about the new case or a document related to the new case the lawyer has already found. There are multiple techniques available to find out if documents are similar.

1.1 Research

In this study a precedent retrieval system is made and researched. The main research question is: How should a precedent retrieval system without manual query construction be developed?
The research is split in three parts, each with a sub-question:

- Scoring function selection: What scoring functions can be used best in legal case retrieval?
- Preprocessing steps: What is the effect of preprocessing choices?
- Integration in the legal field: What is the usability and desirability of a system like this?

These questions will be answered by first creating a system with a set of basic preprocessing steps and multiple different scoring functions. When the best scoring functions are determined the research will continue by researching different preprocessing steps and applying the best scoring functions to them. The research will be finalized by interviewing multiple jurists and investigating how the search for precedence is done currently. Appendix A will be used as a proposal for an interface for the system described in this research, the lawyers will be asked for feedback on this interface and the system behind the interface.

Chapter 2

Background

2.1 Legal Information Retrieval

The field of legal information retrieval exists for 35 to 40 years. Since then there has been a lot of research on this topic. The International Conference of Artificial Intelligence in Law (ICAIL) has been first held in 1987. Bench-Capon et al. showed that in the 25 years after there have been seven papers discussed on Legal Information Retrieval at these conferences since then[9]. Since there has been a big increase in available digital legal cases and other legal documentation, the urgency of an effective retrieval system has not reduced since 1987. Retrieving relevant precedence from an overflow of documents where no ranking or importance is given is an important but difficult task.
Van Opijnen et al.[10] discusses relevancy based on work of Saracevic[11][12], noting his definition of 'Relevance involves an interactive, dynamic establishment of a relation by inference, with intentions toward a context' and inferring it is a comparative concept that can change over time. Relevance in domain specific retrieval systems is hard since they are 'designed by retrieval specialists without comprehensive domain knowledge, sometimes assisted by domain specialists with too little knowledge of retrieval technology'[10]. This shows that when creating a retrieval system it is necessary to dive into the corresponding domain to grasp what is important or relevant.

2.2 Related Work

Information retrieval can be done by query-based retrieval or document similarity. Query-based retrieval matches keywords against all the documents in the database. Legal Intelligence and Rechtsorde use this kind of search with multiple advanced search options so users can find very specific results if needed. The downside of this is that too few search terms or filters can result in a surplus of results, but too many search terms or filters can result in missing important precedence. The alternative to manual query construction is document similarity, which takes a document as an input and compares it to all documents and ranks them on similarity. It returns a list sorted by similarity score. The quality of this list is dependent on the similarity measures taken to compare.
Similarity can be measured with a language model. A language model describes the probabilities of a word or word sequence occurrence within a document. The model tries to capture representative words of a document; these words will have a high probability in the model. These probabilities are based on frequency counts of words in a document. To handle unseen words or word sequences a type of smoothing is applied to give all words a probability. Longer documents have a higher sample size and are therefore more reliable, they will give the same result on successive trials. If words occur more often in longer documents they should therefore have a higher weight than words that occur often in shorter documents. Dirichlet Prior Smoothing does this by taking document length into account when calculating word probabilities[13]. Tian et al. found that a language model using Dirichlet Prior Smoothing works better than BM25 or the Vector Space Model in legal information retrieval[14]. They have retrieved the highest score in the FIRE 2017 IRLeD Track without taking the extra knowledge of citation context (which is unknown in real searches) into account[15].
In COLIEE 2018 the best performing group (called JNLP with Vu Duc Tran et al.) represented documents in a continuous vector space where the summary of the document is embedded. They enriched the summaries

with catchphrases, which were retrieved using catchphrase extraction[16]. Additionally they extracted lexical features of different parts of the documents. Using Rouge formulas they matched unigrams, bigrams, longest common subsequence, weighted longest common subsequence, skip-bigram, and skipped bi-gram + unigram. They found that using both lexical matching and summary encoding gave best results.

2.3 Scoring Functions

The similarity measures mentioned in section 2.2 are not the only option. Multiple scoring functions can be applied to either increase the information–richness of the text or reduce the dimensionality. The vectors or scores from these scoring functions can be used to measure similarity.

2.3.1 Distance

When text is translated into vectors (for example with the techniques described in the following subsections) there are two common ways to measure distance between them. Euclidean distance is a straight line distance measurement. Cosine Similarity uses the angle between two vectors. These distances are shown in figure 2.1, where d is the euclidean distance and θ the cosine similarity. When two document vectors have similar directions (for example topics), but not similar lengths (for example how much a document is about a topic; occurrences of certain words), the euclidean distance measure will be large while the cosine similarity not. If direction is a better indicator than length cosine similarity should be used. Euclidean distance is highly affected by length of the text since the length of the vector will be higher when the amount of words is increased. Whether this is a downside is dependent on the task. When the task is about grouping similar documents the length of the document is not that relevant and should therefore not impact the outcome. When the task is information retrieval the length could be relevant, since longer documents could contain more information. When document length impacts the outcome, the euclidean distance could be normalized.

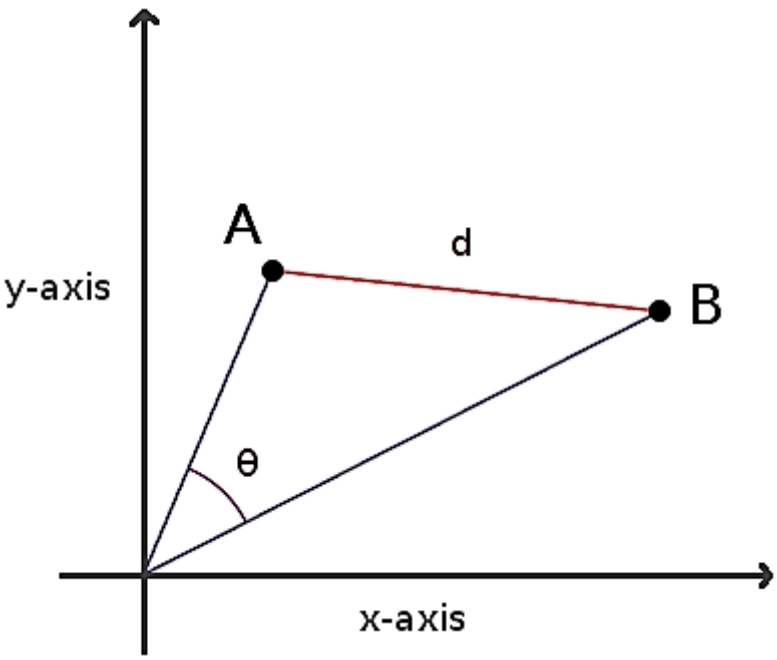


Figure 2.1: Euclidean distance d and Cosine Similarity θ

2.3.2 TF–IDF

To find distinct characteristics of a document just counting words in a single document will not necessarily give a fair insight. Even after removing stop-words specific words appear very often in a document. If such words appear often in all documents it will not be a distinctive term. To account for this Term

Frequency–Inverse Document Frequency (TF–IDF) can be used[17]. This metric indicates how often a word occurs in a document, weighted by the number of documents the word occurs in.

2.3.3 Topic Modeling

A topic model is a statistical model, which can be used to discover abstract topics that are hidden in a collection of documents based on co-occurrence of words. This can be used to group documents based on topic distribution. Latent Dirichlet Allocation (LDA) is a topic model researched in 2003 by Blei et al. and has since then gained popularity[18]. This models topics as a probability distribution over words and documents as a probability distribution of topics. This can be used in document similarity following the reasoning "If two documents have a similar topic distribution, then they will hold similar content".

2.3.4 Word and Paragraph Embedding

Modeling words or documents independent of the exact words used can have a big influence on matching documents with similar content. This was researched by Mikolov et al. in 2013, and was later nicknamed 'word2vec'[19]. There are two types of word2vec models: continuous bag-of-words (CBOW) and continuous skip-gram. CBOW uses a probabilistic feed forward neural network language model, earlier proposed by Bengio et al[20]. This works especially well to predict a word based on the context. However, like the traditional bag-of-words model it does not take sentence structure or word order into consideration. The skip-gram model on the other hand predicts words before and after the current word based on this current word. Both models are shown in figure 2.2.

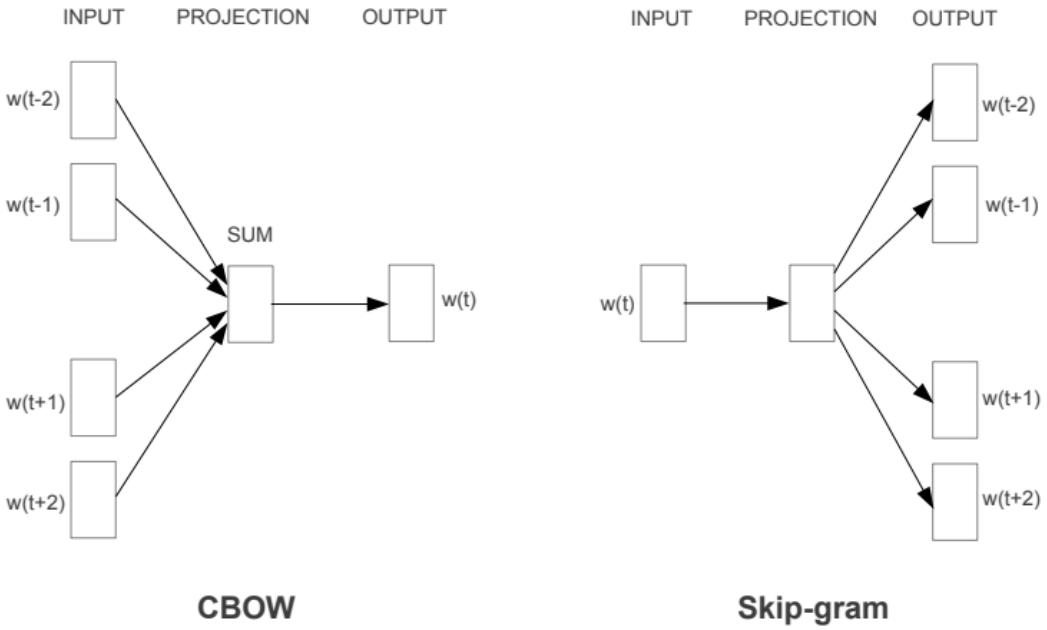


Figure 2.2: Continuous Bag-of-Words and Skip-gram[19]

After word2vec Le and Mikolov developed an extension to this by the nickname of 'doc2vec'[21]. This adds a paragraph id (a unique feature for each document) to the word vectors of word2vec. This results in a dense vector for each document which can be used to calculate similarity between documents.

2.3.5 Elasticsearch

Elasticsearch is an efficient search and analytics system[22]. One of the functionalities in their search is the More Like This-query. It takes a text field, searches for similar texts and gives a score to all results. It

utilizes the Lucene Scoring Formula, which uses TF–IDF[23].
The Lucene Scoring formula is calculated as follows:

$$score(q,d) = coord\text{-}factor(q,d) \cdot query\text{-}boost(q) \cdot \frac{V(q) \cdot V(d)}{|V(q)|} \cdot doc\text{-}len\text{-}norm(d) \cdot doc\text{-}boost(d) \quad (2.1)$$

With:

- *q* as the query
- *d* as a document
- *coord-factor(q,d)*: When multiple items are queried, this factor may reward documents extra for every additional term matched
- *query-boost(q)*: the user can specify boosts to each query, sub-query, and each query term which will receive extra weight
- *V(q)* · *V(d)*: the dot product of the weighted vectors (TF–IDF)
- |*V(q)*|: Euclidean norm
- *doc-len-norm(d)*: document length normalization factor
- *doc-boost(d)*: the user can specify important documents which will receive extra weight

The biggest difference between More Like This and the regular TF–IDF is that More Like This always normalizes on document length. Though relevancy classification tends to favor longer documents, since they contain more words and therefore could be unfairly advantaged, documents holding absolute more relevant information could be more interesting than those which hold relatively more relevant information.

2.4 Random Forest Classifier

A Random Forest Classifier (RFC) is an ensemble learning method using multiple different decision trees[24][25]. A decision tree is created using features of observations as input and target values (e.g. relevancy) as output. A tree groups candidates together based on target value and splits the tree based on similarities the features. Figure 2.3 shows an example of a decision tree that may appear in a random forest.

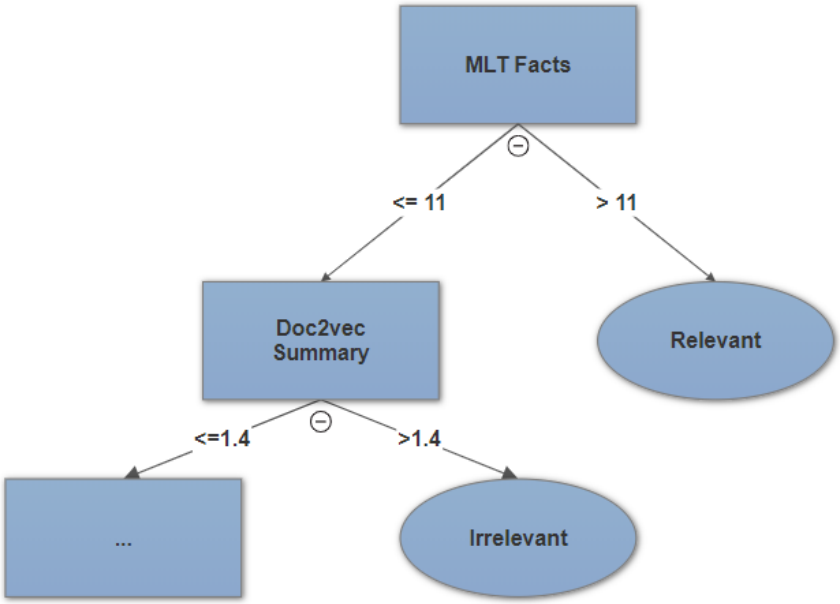


Figure 2.3: Decision Tree

A random forest exists out of many decision trees. All trees vote for the target value. The target value with the most votes is the output of the random forest. An RFC is a solid choice when trying to find interaction between features and keeping a robust system when adding noise features. In research where different features are researched an RFC is a good option to retrieve good results, while also researching which features have a high feature importance and therefore cause these results. Additionally, an RFC is computationally cheaper than alternatives like a Support Vector Machine[25].

2.5 Preprocessing

Processing text is difficult when the text is in the unstructured (almost chaotic) format it normally is in. Non-ASCII characters, missing or double spaces, an overload of punctuation and words that do not add any information are very common in all kinds of text documents, but makes information retrieval more difficult. To handle these problems the text can be preprocessed. The task of preprocessing can exist out of multiple techniques to make the text more manageable and find similarities between texts that otherwise would have been missed. This is always an essential step before extracting scoring functions, because if chaotic text is used as an input, unusable scoring functions and results will be found.

Stop-words are words that do not distinguish a document from others. Words like 'a', 'this', 'be' or 'further' do not add any information to a single document and can therefore be removed. That too common words can add noise to text was already found in 1958 by Luhn[26].

2.5.1 Stemming and Lemmatization

It is difficult to find words with different suffixes, but similar meanings in different documents. Two techniques can be used to normalize words. Stemming is a simpler technique which changes words like 'computing' or 'computer' to its shortened form 'comput'. This helps with different types of words with easy suffixes. It has trouble with forms like 'worse' and 'worst', which the stemmer reduces to 'wors' and 'worst' respectively[27]. Lemmatization on the other hand reduces form to their lemma. It would change 'worse' and 'worst' to 'bad'. It uses a part-of-speech tagger that marks every word as its grammatical type. Depending on the type this changes how it should be lemmatized.
Both stemming and lemmatization have been proven successful for natural language processing and document retrieval. The study by Balakrishnan and Lloyd-Yemoh comparing stemming and lemmatization found only a slight performance increase when using lemmatization, but no significant difference[28].

Noun Phrase Chunking

A noun phrase often has a noun as its head and can have additional quantifiers, determiners or adjectives. Examples of a noun phrase are 'people', 'my first car' and 'the two black chairs I have in my room'. This technique can be used to match word groups instead of just singular words.

Chapter 3

Methods

I first researched the different possible scoring functions to find the most indicative scoring functions using a base set of preprocessing steps. Next I researched different preprocessing steps using the best scoring functions from the first part. I finished by researching the usability and desirability of a queryless retrieval system, and how to integrate it into the legal field. In this chapter these steps are explained along with the data used in this research.

3.1 Data

The data used in this research is given by the Competition on Legal Information Extraction/Entailment (COLIEE). COLIEE is an annual competition held to gather legal information retrieval techniques and knowledge[29]. This competition has close ties to the Juris-Informatics workshop (JURISIN) and at the workshop there is a special session to present the results[30]. COLIEE 2018 existed of four different tasks. The first task involves legal case retrieval. A number of supporting cases (on average ten) from 200 candidates are to be extracted for each of 285 new cases. The new cases have a short summary and corresponding facts. The candidates are not the same for every new case. The second task is about entailment. A decision in a new case is based on a relevant old case. The aim of this task is to find the paragraph of the old case on which the decision is based. The third task is finding relevant Japanese Civil Code Articles for a legal bar exam question. The answer is a number of articles for every question. The fourth and final task is to identify relevant articles for questions.
I participated in this competition to be able to compare the results of competition to my own submission and the research described in this thesis. I only participated in the first of the four tasks.

A visual of the used data is shown in figure 3.1.
The training data exists out of 285 new cases. Each new case has a textual summary describing the case, and a list of facts about the case. An example of a summary and facts of a new case is given in appendix B and C.
Each of the new cases has 200 prior cases as candidates(57,000 total for the training set). The task at hand is to select for each new case which of these 200 candidates are relevant for the new case. Every new case has a different set of candidate cases, there is no overlap between the candidates. The distribution of word counts of the candidate cases, the facts and the summary are somewhat skewed to the right side. This is shown in table 3.1, where the average, median and maximum word count, and standard deviation are given.

Table 3.1: Documents word count

Type	Median	Average	Std. Deviation	Maximum word count
Candidates	24,654	35,533	37,390.98	516,321
Facts	5,675	6,772	4,549.85	33,113
Summary	426	538	439.12	5,247

This shows that the candidates are significantly longer and presumably hold more information than the provided facts and summaries for the new cases. The test data exists of 59 new cases, also each with 200 candidate cases (11,800 total).

Figure 3.1: COLIEE Data

3.2 System Overview

The system exists of three steps: preprocessing, scoring function extraction and classification. The preprocessing cleans the data without losing information. Scoring function extraction is the usage of different techniques to represent a document by either a score or a vector. Vectors can be used to find the distance between new cases (summary and facts) and a candidate. These numeric characteristics (score and distance) can then be used as input for a classifier to find patterns in characteristics impacting the relevancy of the candidate, as shown in figure 3.2.

3.2.1 Distance

To find the distance between two vectors I decided to use both euclidean distance and cosine similarity. Cosine similarity is a good indicator for similar topics, but euclidean can find the difference in the intensity of a topics. Figure 3.3 shows 4 hypothetical vectors plotting both topic distribution (angle) and intensity of present topics (length).
When trying to find the most relevant document for document B (represented by vector B), cosine similarity shows that document A is less relevant, because they are not about similar topics, though their euclidean distance is small. When trying to find the most relevant document for document C, document D may be more relevant than document B, since not only is the direction very similar, but intensity of the topics are

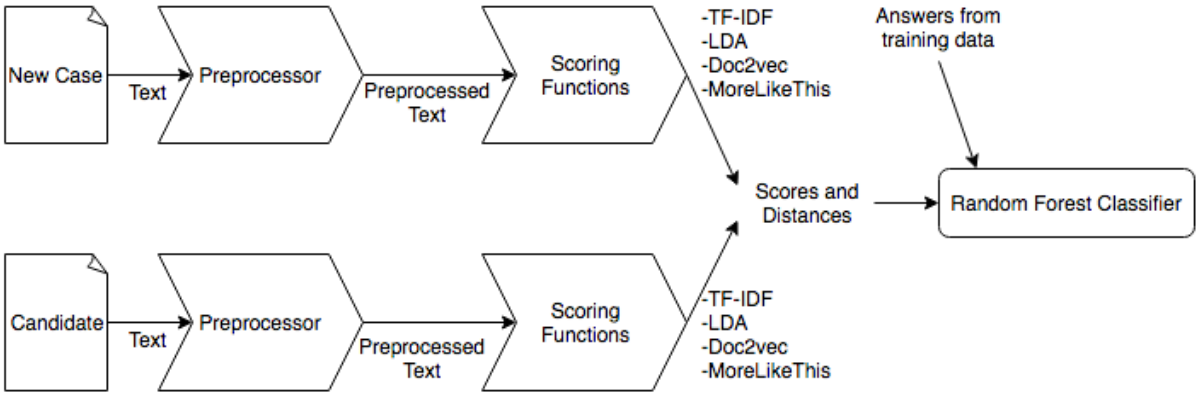


Figure 3.2: Pipeline of the system

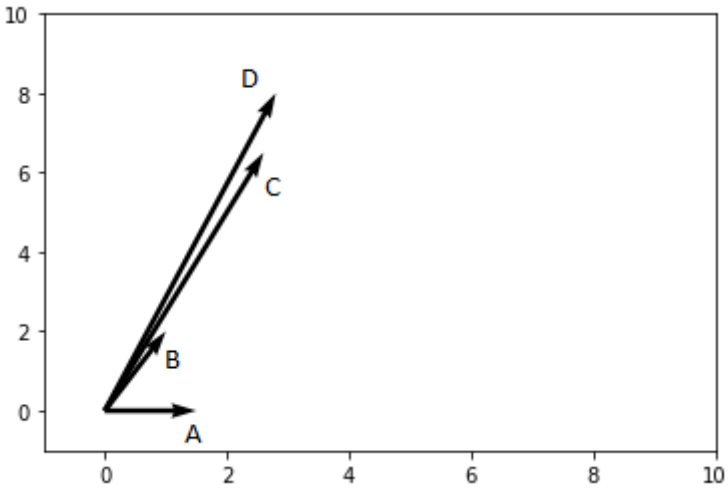


Figure 3.3: Vector distance comparison

also almost equal. Cosine similarity would indicate these were equally similar. Using both distance functions will give the information about the topic similarity as well as the intensity.

3.3 Scoring Functions

I researched TF–IDF, doc2vec and LDA vectors, and a 'More Like This' score as scoring functions for this information retrieval system.
The basic preprocessing steps prior to scoring function extraction are the following:

1. Lowercased entire text
2. Removing all characters that are not a digit, letter or space
3. Removed additional info like the numbered tags for the paragraphs
4. Removed all punctuation marks
5. Splitting adjacent words and numbers (e.g. 'number2' to 'number 2')
6. Replaced 'subject act' with 'subject_act'
7. Replaced 's 4', 'sec 4' or 'section 4' with 'section_4' (for all numbers)
8. Step 7 but with 'art' or 'article'
9. Step 7 but with 'topic'
10. Removed all words with two or less characters
11. Removed loose digits

12. Removed all stop words (using the nltk stop words package)
13. Lemmatized all words (using the nltk WordNetLemmatizer)

Doc2vec is implemented with Doc2vec gensim model, using a vector size of 50 and 100, a minimum count of 4, 80 epochs and the other settings on default. It splits the training and test set and does not include the new case.

I use the implementation of TF–IDF in the sklearn package in Python for unigrams, with different minimum document frequencies (4, 40, 250, 400). There are two choices to be made regarding the TF–IDF corpus. The amount of candidates in the corpus can exist of:

- All candidate cases (68,800 in the used dataset)
- Split the training set and test set candidates (train: 57,000; test: 11,800)
- Only the possible candidate cases for a single new case (200)

Additionally the new cases can be part of the corpus or not. This results in six different options. These are all researched.

LDA is implemented with the gensim LdaMulticore model. It uses 40 topics, the top 4000 words and the other settings on default.

The More Like This query from Elasticsearch uses two text fields and gives a score based on the similarity between them. The system compared each candidate to both the facts and the summary of the new case. It is implemented with the default setting.

3.3.1 Random Forest Classifier

The outputs of the scoring functions described in 3.3 are used as an input for a Random Forest Classifier (RFC). The parameters are researched with a grid search, tested with a 3 and 5 cross-validation folds. The refitting is done on F–measure. The RFC is tuned with the cross validation on the training set, with a test size of 0.3 or 30%. The train and test set split is stratified. To avoid skewed results I execute the RFC three times and I average the results.
Additionally I researched the use of a Decision Tree instead of an RFC. The Decision Tree is researched in the same way as the RFC.

3.4 Preprocessing

Using the information from the scoring function research I continue by researching the impact of individual preprocessing steps. I research the impact of the following steps:

- Using stemming instead of lemmatization
- Removing parentheses and the text in between
- Noun Phrase Chunking for acts
- Removing all words with two or less characters
- Replacing 'subject act' with 'subject_act'
- Replaced 's 4', 'sec 4' or 'section 4' with 'section_4' (for all numbers)
- Previous step but with 'art' or 'article'

To find the value of individual steps I compare the score of using all steps with the score of using all but one step. I take the best result and continue in the same fashion for the other steps. This greedy way of working could miss the best option (since not all possible combinations are tried), but is the most time-efficient way.

3.4.1 Noun Phrase Chunking

A lot of acts in the data do not exist out of two words, but often multiple, for example 'the immigration and refugee protection act'. With step 5 of the preprocessing described above, this would change to 'the immigration and refugee protection_act'. Since the terms would then still appear loose this would lessen the usability of TF–IDF. To change an act to one term connected by underscores I used Noun Phrase Chunking. The act would then change to 'the_immigration_and_refugee_protection_act'. I tested preprocessing

both with and without Noun Phrase Chunking to see the impact. If an act would only appear in a document as one term with underscores, it would lose some information. In the example mentioned above the words 'immigration' and 'refugee' would no longer be individually matched if it is connected with underscores. Therefore it may be better to leave the original form intact and put a noun phrase chunked version after it, and that is how I execute it.
Noun Phrase Chunking is implemented using the 'make_doc_from_text_chunks' function from the textacy package[31]. This performs slower than using the RegexpParser from nltk with a grammar. However, the textacy function is able to handle huge amounts of text, which is necessary in this research.

Chapter 4

Results

During the research for this thesis I competed in the COLIEE competition. My submission retrieved second place. After writing a paper about the research, I was invited to present my findings at the twelfth JURISIN conference held at Keio University on 12 and 13 November 2018. During my participation in COLIEE I obtained the results showcased in section 4.2. After this competition I continued my research and obtained the results showcased in the subsequent sections.

4.1 Result Metrics

The results are measured with three different metrics: Precision, Recall and F-measure. These three are calculated as follows:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{4.1}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4.2}$$

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.3}$$

This means that Precision looks how often the system was correct out of all records classified as relevant, while Recall looks at how often the system could have been right. In the field of case law retrieval the recall metric is especially important, since the amount of relevant precedence lawyers and judges miss should be as low as possible. However, when the system classifies everything as relevant it has a recall of 1; when the system classifies everything as irrelevant, except when it is absolutely sure it has a precision of 1. This both gives a skewed insight in how the system is performing. To avoid this the F-measure is also calculated, taking both Recall and Precision into account, giving better insight in the overall performance.

The ranked retrieval task described in this thesis is executed like a classification task. The Random Forest Classifier uses a threshold to determine relevancy of a candidate based on the features. There is no lower or upper limit to the amount of cases that could be classified as relevant. Out of the 200 candidates 9.9 cases are classified as relevant on average in the training data; for the test set this is 10.7 cases. On average 4.5 relevant candidates for a new case are actually retrieved. This means that on average 6.2 cases are missed in the retrieval of relevant cases in the test set. The result of the lower retrieval is a maximum recall of of 45%.

4.2 COLIEE Workshop

During my participation in the COLIEE competition I obtained the following results. For my submission I used the base set of preprocessing steps described in section 3.3. The scoring functions used are shown in table 4.1.

Table 4.1: Scoring Functions for COLIEE submission

Scoring Functions
More Like This Score on Facts
More Like This Score on Summary
Doc2vec Cosine Similarity distance to Facts
Doc2vec Cosine Similarity distance to Summary
TF–IDF Euclidean distance to Facts
TF–IDF Euclidean distance to Summary
TF–IDF Cosine similarity distance to Facts
TF–IDF Cosine similarity distance to Summary

Table 4.2: Random Forest Classifier Parameters.

Parameter	*Value*	*Researched Range*
Number of trees	45	15–85 in steps of 5
Criterion	Gini (impurity)	Gini and Entropy
Minimum samples to split a node	2	2–12 in steps of 1
Maximum depth	35 nodes	10–80 in steps of 5
Minimum samples in a leaf	2	2–10 in steps of 1
Maximum scoring functions in a tree	Square root of number of scoring functions	1–8 in steps of 1 and square root

Tuning the hyperparameters of the RFC resulted in the values shown in table 4.2.
All scoring functions are attributed a feature importance by the Random Forest Classifier. An example is shown in table 4.3.

Table 4.3: Feature Importance

Scoring function	Importance
More Like This Score on Facts	0.1868
TF–IDF Euclidean distance to Facts	0.1459
TF–IDF Cosine similarity distance to Summary	0.1452
TF–IDF Cosine similarity distance to Facts	0.1335
TF–IDF Euclidean distance to Summary	0.1116
More Like This Score on Summary	0.1103
Doc2vec Cosine similarity distance to Summary	0.0838
Doc2vec Cosine similarity distance to Facts	0.0829

There were eleven other submissions by five other teams. The results are shown in table 4.4. UL is the tag I used for the competition. The table is sorted by F-measure.
After my submission I tried to improve my results by decreasing the TF–IDf Minimum Document Frequency, grouping TF–IDF corpora, decreasing the Doc2vec vector length and removing less important features. The result to this continued research is shown in table 4.5.

These results served as a base for the rest of my research.

4.3 Scoring Functions

In addition to TF–IDF, Doc2vec and MLT, I researched the use of LDA as a scoring function for the Random Forest Classifier. Different permutations of the scoring functions are tested with different distance measurements (if applicable). The results are shown in table 4.6. TF–IDF to facts and MLT are combined

Table 4.4: Results COLIEE

id	*F-measure*	*Precision*	*Recall*
JNLP-k=10	0.654635	0.676271	0.634340
JNLP-r=2.5	0.595806	0.546419	0.655008
UL	**0.393375**	**0.563798**	**0.302067**
HUKB1	0.380765	0.497436	0.308426
UA-postproc	0.374080	0.348422	0.403816
UA-smote	0.372268	0.353868	0.392687
HUKB2	0.346957	0.404661	0.303657
UA	0.345826	0.372477	0.322734
Smartlaw	0.344565	0.287076	0.430843
UBIRLED-2	0.307536	0.195511	0.720191
UBIRLED-1	0.219056	0.132881	0.623211
UBIRLED-3	0.172275	0.561404	0.101749

Table 4.5: Continued Research

Improvement	*Change/Value*	*F-measure*	*Precision*	*Recall*
Base line	-	0.3934	0.5638	0.3021
Minimum Doc. Freq.	4	0.3337	0.5736	0.2353
	40	0.3411	0.5620	0.2448
	250	0.3690	0.5415	0.2798
TF–IDF	Grouped corpora	0.3520	**0.5970**	0.2496
Doc2vec length	50	0.3797	0.5253	0.2973
Removing less important features	No Doc2vec	**0.4052**	0.5537	**0.3196**
	No eucl. and cosine to summary	0.3562	0.4554	0.2925

together since TF–IDF to the summary document has the next lowest Feature Importance after removing Doc2vec.

The TF–IDF corpora splitting choices are tested with the TF–IDF and MLT scoring functions. The results of this research is shown in 4.7. Here 'possible candidates' means only the 200 candidates a new case has; 'all candidates' means also the candidates of the other new cases. Training set and test set split means that the candidates of the training set form a corpora for training and those of the test set for testing.

The Decision Tree Classifier and preprocessing steps are tested with the TF–IDF and MLT scoring functions, since these have given the best results so far. Using all candidates and not including the new case is the situation most similar to real life (where there is a database with thousands of documents and it takes too much time to change the system when a new case arrives), therefore this is used in the subsequent research. To find the amount of variation in the F-measure (for the TF–IDF corpus including all candidates and excluding the new cases), the F-measure is calculated over for every single new case. With this list of F-measures is the standard deviation calculated. When none of the 200 candidates is predicted as relevant the precision, and therefore the f-measure, is set as 0. Therefore the average over the F-measures for all new cases (the macro-average) can be lower than the overall F-measure (the micro-average). This is shown in table 4.8.

4.3.1 Decision Tree

The research into the use of a Decision Tree is shown in table 4.9. The parameters that performed best for the Random Forest also performed best for the Decision Tree (without the number of trees, since that is not a parameter in the Decision Tree).
This shows that the Precision of the Random Forest Classifier is significantly higher compared to the Decision Tree. However, the Recall is higher for the Decision Tree, this results in a close F-measure.

4.4 Preprocessing

The result of testing different permutations of preprocessing steps is shown in table 4.10.

Some acts had an abbreviation after them for example: "He was ordered to be removed for reasons of serious criminality under s. 36(1)(a) of the Immigration Refugee and Protection Act (IRPA)". Noun Phrase Chunking would presumably perform better when the abbreviated version between parentheses is removed, therefore this is tested with and without that step.

Table 4.10 shows the following steps have a negative influence on the performance.

- Stemming instead of lemmatization
- Removing the parentheses and text in between
- Noun Chunk Phrasing to replace 'subject act' to 'subject_act'
- Replacing 's 4', 'sec 4' or 'section 4' with 'section_4' (for all numbers)

The next steps have a positive influence on the performance.

- Removing words with two or less characters
- Replacing 'art 2' or 'article 4' with 'article_4' (for all numbers)
- Replacing 'subject act' with subject_act

To test the significance of the difference between different preprocessing steps I executed the Wilcoxon signed-rank test. This indicated that for this sample size (n=59) the differences between the settings are too small to be significant. The lowest P-value obtained – for the difference between the highest *(Base - replacing 'sec 4' with 'section_4')* and the lowest *(Base + removing parentheses and words in between)* performing setting – is p=0.081 (Z=280.0).

Table 4.10 shows that using the basic preprocessing steps without replacing 's 4', 'sec 4' or 'section 4' with 'section_4', and using TF–IDF and More Like This as scoring functions to represent documents provides the best result (when using all candidates for the creation of the TF–IDF corpus).
The entire preprocessing that provides the best results exists out of the following steps.

1. Lowercased entire text
2. Removing all characters that are not a digit, letter or space
3. Removed additional info like the numbered tags for the paragraphs
4. Removed all punctuation marks
5. Splitting adjacent words and numbers (e.g. 'number2' to 'number 2')
6. Replaced 'subject act' with 'subject_act'
7. Replacing 'art 2' or 'article 4' with 'article_4' (for all numbers)
8. Replacing 'topic 3'with 'topic_3'
9. Removed loose digits
10. Removed all words with two or less characters
11. Removed all stop words (using the nltk stop words package)
12. Lemmatized all words (using the nltk WordNetLemmatizer)

Table 4.6: Scoring function subset results

Summary functions	*Scoring func.*	F-Measure	Precision	Recall
TF–IDF, MLT and Doc2vec	TF–IDF Eucl. to Facts TF–IDF Eucl. to Sum. TF–IDF Cos. to Facts TF–IDF Cos. to Sum. MLT Score Facts MLT Score Sum. Doc2vec Cos. Facts Doc2vec Cos. Sum.	0.3934	**0.5638**	0.3021
TF–IDF and MLT	TF–IDF Eucl. to Facts TF–IDF Eucl. to Sum. TF–IDF Cos. to Facts TF–IDF Cos. to Sum. MLT Score Facts MLT Score Sum.	**0.4052**	0.5537	0.3196
TF–IDF, MLT and LDA	TF–IDF Eucl. to Facts TF–IDF Eucl. to Sum. TF–IDF Cos. to Facts TF–IDF Cos. to Sum. MLT Score Facts MLT Score Sum. LDA Cos. Facts LDA Cos. Sum.	0.3699	0.5400	0.2814
Only TF–IDF	TF–IDF Eucl. to Facts TF–IDF Eucl. to Sum. TF–IDF Cos. to Facts TF–IDF Cos. to Sum.	0.3483	0.3741	0.3259
TF–IDF Euc. and MLT	TF–IDF Eucl. to Facts TF–IDF Eucl. to Sum. MLT Score Facts MLT Score Sum.	0.3770	0.4886	0.3068
TF–IDF Cos. and MLT	TF–IDF Cos. to Facts TF–IDF Cos. to Sum. MLT Score Facts MLT Score Sum.	0.3858	0.5105	0.3100
TF–IDF to Facts and MLT	TF–IDF Eucl. to Facts TF–IDF Cos. to Facts MLT Score Facts MLT Score Sum.	0.3562	0.4554	0.2925
Only TF–IDF Cosine	TF–IDF Cos. to Facts TF–IDF Cos. to Sum.	0.3761	0.4361	**0.3307**
Only TF–IDF Euclidean	TF–IDF Eucl. to Facts TF–IDF Eucl. to Sum.	0.3478	0.3839	0.3180
Only MLT	MLT Score Facts MLT Score Sum.	0.2669	0.4000	0.2003

Table 4.7: TF–IDF Corpora F-scores, Precision and Recall

TF–IDF Corpora	F-measure	Precision	Recall
Without new cases			
All candidates	0.3392	0.5576	0.2438
Training set and test set split	0.3695	0.5005	0.2931
Possible candidates	0.3366	0.5281	0.2401
With new cases			
All candidates	0.3520	**0.5970**	0.2496
Training set and test set split	**0.3871**	0.5082	**0.3127**
Possible candidates	0.3146	0.5234	0.2250

Table 4.8: F-measure and Standard Deviation

Measurement	Value
Micro-average F-measure	0.3392
Macro-average F-measure	0.2930
Standard Deviation	0.2173

Table 4.9: Decision Tree vs Random Forest results

Classifier	F-measure	Precision	Recall
Random Forest	**0.3392**	**0.5576**	0.2438
Decision Tree	0.3158	0.3749	**0.2729**

Table 4.10: Preprocessing results

Preprocessing steps	F-measure	Precision	Recall
Base	0.3392	0.5576	0.2438
Base + Stemming instead of Lemmatization	0.3342	0.5556	0.2390
Base + removing parentheses and words in between	0.3224	0.5318	0.2313
Base + removing parentheses and words in between + NPC instead of replacing 'subject act' with subject_act	0.3357	0.5303	0.2456
Base + NPC instead of replacing 'subject act' with 'subject_act'	0.3254	0.5390	0.2332
Base - removing words with 2 or less char	0.3383	**0.5613**	0.2422
Base - replacing 'sec 4' with 'section_4'	**0.3452**	0.5567	**0.2464**
Base - replacing 'sec 4' with 'section_4' - replacing 'art 2' with 'article_2'	0.3419	0.5536	0.2475
Base - replacing 'sec 4' with 'section_4' - replacing 'subject act' with 'subject_act'	0.3370	0.5394	0.2448

Chapter 5

Discussion

In this chapter the results of the research in the scoring function selection and preprocessing steps will be discussed.

5.1 Scoring Functions

The feature importances of table 4.3 and the results of table 4.6 give insight into what scoring functions are actually representative of a candidate case. It shows that a combination of TF–IDF and More Like This performs best as scoring functions. Since the doc2vec scoring functions do not improve the system, they are apparently not able to capture the document accurately in a single vector. Doc2vec is trained using all documents, causing it to be unable to capture documents accurately, possibly due to too few documents. The performance of doc2vec could go up when using more data or using a pre-trained model[32].

LDA does also not improve the system. This could be caused by the selected parameters, or that the topic distribution is not an actual representation of the content of a document.
If the value for minimum samples to split a node and minimum samples in a leaf in the Random Forest Classifier (table 4.2) would have been high then the data set would have a lot of similar examples and would therefore not been trustworthy (since there are only 57,000 candidate cases). Since these values are low this indicates that the data is very distinguishable and thus more trustworthy.

The research into the TF–IDF corpora is surprising. The data is randomly split between the train and test set, therefore it should not give so much better result to also make a split in the corpus instead of using all candidates or only the possible candidates. There are two possibilities for this improved result when making a random split in the corpora: an actual coincidence or the data was not randomly split between training and test. When the training and test data does not have a similar distributions of contents, topics or type of cases it can give misleading outcomes like in this research. It should not be concluded that splitting the TF–IDF corpora with training and test data will always perform best. Moreover this is not feasible to execute in the real world, where there is no training and test data, but only a database of possible candidates.

The Decision Tree performs worse than the Random Forest Classifier on F-measure and Precision, but has a higher Recall. Since Recall is classically regarded as more important than Precision in the legal field, therefore using a Decision Tree is a better option for case law retrieval.

5.2 Preprocessing

The results of the preprocessing research (table 4.10) show that on this data set replacing 'article 2' with 'article_2' and 'subject act' with 'subject_act' improve the performance of the retrieval system. The preprocessing steps that remove words with two or less characters, that replace 'sec 4' with 'section_4', that remove parentheses and the text in between, and that use stemming instead of lemmatization all decrease the performance. That stemming performs slightly worse than lemmatization is in line with the research done by Balakrishnan and Lloyd-Yemoh[28].

Individual preprocessing steps have a lower impact on the performance than scoring functions. While removing or adding a scoring function often increases or decreases the F-measure with 0.04, the preprocessing steps impact the F-measure with 0.017 at most. The results show that data-specific preprocessing steps like 'subject act' to 'subject_act' can have a positive impact on the score. However, this is not the case for all data-specific preprocessing steps, for example 'sec 4' to 'section_4' did not increase the F-measure.

It is important to note that the preprocessing is highly dependent on the data. This means that using preprocessing steps that decrease the F-measure using the data set used in this research could increase F-measure on another data set. Nevertheless, the research does show that these steps can influence the processing of textual data, and are therefore worth trying as preprocessing steps.
Most preprocessing steps were executed using regular expressions and substituting patterns. These steps don't take much time and are therefore worth trying. Using Noun Chunk Phrasing, however, does take a long time. Importing 57,570 documents into Elasticsearch while making substitutions using 11 regular expressions, took around 3 hours. The same documents with the same regular expressions and also using Noun Chunk Phrasing took around 8 hours. Therefore it is advised to not employ Noun Chunk Phrasing when time is scarce.

Chapter 6

Integration into the Legal Field

A functional legal information retrieval system is useless without users. To increase the chance of adoption I interviewed five lawyers and one case law secretary to get more information about how they search for precedence now, and if they would want to integrate the techniques described in this research into a system they are familiar with.

6.1 Methods

The interviews exists out of three parts. The first part is about how they currently do legal research, whether it takes a long time and whether they have missed important precedence that they know of. In the second part, I showcase my system using the designs found in appendix A and I explain how they would use it. The final part of the interview aims to find the usability and desirability of the system, to see whether the system adds enough value and find suggestions for improvement.

The lawyers and the case law secretary were specialized in one (or more) of the following fields:

- Labor Law
- Real Estate Law
- Construction Law
- Commercial Contract Law
- ICT law
- Penitentiary and civil law

The lawyers are from different law firms, and these law firms vary in size.
The interviews will follow a funnel approach with open questions at the start at specific questions towards the end. The topic list is shown in appendix D.
All interviews are recorded. After an interview I note the most important points from the interview, based on memory and the notes during the interview. I check if these points are correct and are all the points made during this interview, by listening to the recording. The records are anonymised and will be saved until the start of 2022.

6.2 Results

I talked with them about their current situation with regards to legal search, the visualizations shown in appendix A, the usability and desirability of a system like this and potential improvements to the system.

6.2.1 Current Situation

When a new case arrives multiple lawyers do not start at Legal Intelligence or Rechtsorde, but with Google or 'Tekst & Commentaar'. This initial search provides an insight in the legal matter at hand and gives search terms to use in a deep dive using one of the legal search engines. The substantial downside all legal employees mention is the great number of search results when searching, sometimes around 30,000. This gives a considerable number of files to read or scan through at least. In addition, it is notable that when a jurist knows something is written about the subject he/she will sometimes not find it, while using

the search terms he/she knows are in there. It differs per jurist how much time they spend on searching precedence, depending on their field and the size of the firm. Lawyers working at large firms where a lot of students/trainees work often search for under 30 minutes before handing it to a student/trainee. The time spend on searching precedence is also dependent on the size of the case. Since small cases have a short time span and contain little information, they don't need a lot of precedent to make a case.

6.2.2 Reaction to the Proposed System

The explanation of the system together with showing appendix A has received a positive response from the jurists. It has made clear what the idea of the system is and what the process would look like.
All six are willing to give query-less searching a try, since they see the downside of query construction and don't see a downsides to using documents. The document to be used as an input for this system can be the setup for a new case or a relevant article that was already found. However they do mention the dependency on ICT system familiarity, meaning that the user needs to understand how to select a document from a computer as input. Whereas this is intuitive for people experienced with computers or different digital systems, this is not the case for the older employees of a law firm, e.g. the senior partners. Old school lawyers (often) are skilled at retrieving relevant information from a paper collection of cases, and have become used to searching with search terms in the search engine, causing them not to be open to change to more complicated digital actions.

6.2.3 Suggestions for Improvement

The lawyers and the case law secretary suggest the following improvements.

Argument Mining & Search Result Improvement

When a lawyer has thought of an argumentation line for a new case it searches precedence to support this line. A legal system can therefore be improved by extracting argumentation from a text. Mining arguments from a text has become an own field of study since 2011[33]. Extracting arguments could improve a search engine in two separate ways. Firstly, it could improve the search results, since searching in a smaller part of crucial information is better than searching long documents with a lot of noise. Secondly, the arguments of all cases could be shown in the search results list. This will help users to find the more interesting results sooner, since they can see the most relevant information in a glimpse instead of opening and scanning the document first. An additional option to improve the search result list is by showing a small summary or the essence of the case. This way the lawyer does not have to click the results to find out if the case is indeed similar to the new case. This could either be manually implemented (a legal employee writes a summary when uploading a case) or using summarization techniques like Textrank[34].

Translating Legal Language

A case may be opened by receiving mail from a customer stating the situation he or she is in. This will not contain words a lawyer normally uses to search. The search engine could be improved using a legal translation system, one that takes the layman's terms and translates it to legal terms, that could appear in a case. This will reduce the time a lawyer spends on doing this translation.

Additional Filtering Options

Whether the name of the company is important for searching precedence depends of the situation. When a lawyer is handling a case for or about a company the lawyer could want to find cases this company is involved in. For example in case of an enforcement company, a lawyer may search for cases where a fine of this company is acted against. In other cases the lawyer only cares about the situation, not about the specific company, the company name should be filtered out so it will not clutter the search results.

Target Audience

Law firms in the Netherlands either use Legal Intelligence, Rechtsorde or use no system. The firms that use no system are often small firms that make use of free sources like `www.rechtspraak.nl`,`www.jure.nl` and `www.raadvanstate.nl`. Since Legal Intelligence and Rechtsorde both cost around 600 euro each year

per fee-earner, small companies that do not want to spend that amount of money for a legal search engine don't have a search engine to look into all the free resources, while they do have a need to search all free resources simultaneously. This means that there is a market for cheaper legal search engines or alternatives like pay per use.

Duplicates

Judgments can be published in multiple journals. When Rechtsorde and Legal Intelligence retrieve these journals the judgments are duplicate. These are currently not removed, resulting in a lot of duplicate cases, cluttering the search results very much. A big improvement to legal search engines in general is to remove duplicates. A list with all journals in which a judgment is published is valuable information to keep track off.

Author Scoring

When authors have a lot expertise in certain fields their articles are more important than those of other authors. This could be incorporated into the ranking of the search results. Authors writing about the field they graduated in, or have written a lot about have presumably more knowledge about this field than others and should therefore be valued higher.

6.3 Discussion

The results of the interviews give insight into the current situation of the jurists, the usability and desirability of the system, and the room for improvement.
It differs per law firm how much use they would get out of a system described in this thesis. Bigger cases need more precedent and therefore benefit the most from an improved search engine decreasing the search time and improving relevance of search results. As bigger offices often have a lot of students who get the tasks of searching relevant precedence, the total time spend searching will be higher and therefore the system described in this research may prove more relevant.
The results show that before using a legal search engine the jurists Google information or cases, or search in 'Tekst & Commentaar'. This gives (ideas for) search terms they use in the search engine. To use the system proposed in this research a document is given as an input to find similar documents. Since jurists already have found relevant documents in 'Tekst & Commentaar' or with Google, it is a small step to input these documents instead of the search terms retrieved from these documents. This shows that the barrier to integrate the system into the legal search process is low, and therefore has a higher level of usability. This is underlined by section 6.2.2 which shows that the jurists understand the usability of the system. This is not necessarily the case for all legal employees. Especially the senior partners with less experience in using digital systems may have a hard time understanding and using it.

The biggest flaws of Legal Intelligence and Rechtsorde is not removing duplicates and not always showing only relevant search results, both often result in an abundance of search results. Removing duplicates will significantly improve search results since the only information a duplicate adds is in what journal the article/judgment also appeared. Since lawyers are searching for judgments instead of journals the additional time this article takes is not worth the information that is retrieved from it.
The other suggestions of the legal employees can result in either an improvement of what is relevant or add functionality. Indicating the importance of an article based on how how influential an author is in the field can be a good improvement on ranking search results. To find how relevant or influential authors are can be done by:

- Sending a survey out to a lot of law firms to find which authors are the most influential in their field of expertise
- Ranking authors on how much they have written about a certain topic
- Finding authors that graduated in the specific field

Implementing this ranking system will improve the system since the more experience/expertise an author has in a field, the more useful articles and judgments he/she will refer to and discuss.

Argumentation mining can be used to extract the used arguments from a case. This would improve the system since users can see based on the arguments if a case is fit for their need. Adding argumentation mining to the system is a big step. With case law it is possible to do it in multiple ways. The more traditional

way is context dependent argument mining[35]. Lippi and Torroni researched context independent claim detection, which is a part of argumentation mining[36].
The suggestion to provide a cheap solution to smaller companies unwilling to pay 600 euro a year per fee-earner for Legal Intelligence or Rechtsorde will be difficult to execute since the money investment to set up a system like this will be significant. Therefore margin on the product will be high, which results in a higher price, repelling this target audience.

Chapter 7

Conclusion

To improve the way jurists search for precedence, a retrieval system without manual query construction was researched and described in this thesis. The research was split in three parts: scoring function selection, preprocessing step choices and integration into the legal field. The research has shown that TF–IDF and More Like This outperform doc2vec and LDA as scoring functions; TF–IDF and More Like This give a fair representation of the documents.
The preprocessing research has shown that basic preprocessing like lowercasing and removing words with two or less characters works well. Data-specific preprocessing steps (like replacing 'sec. 4' with 'section_4' or 'art. 2' with 'article_2') sometimes improve performance. This depends on the specific preprocessing step. These kind of steps are therefore worth investigating when dealing with text.
The qualitative research into the process of integrating a precedence retrieval system into the legal field has shown that the legal search currently takes extra time because of the great number of search results and not being able to find an article or case that he/she knows exist. A queryless retrieval system is not a downside for digitally experienced users, but can be a problem for older employees like senior partners. Using an article or a setup for a new case are both good options to use as an input to the system.
The usability of the system can be increased by improving the search results list with extracted arguments, author score or not showing duplicates. These or other improvements can be implemented easily in the system proposed in this thesis. In all three stages of the system (preprocessing, scoring function extraction, classification) it is possible to alter, replace or extend the system.

7.1 Future Work

Though adding doc2vec to the system did not improve the results, document embedding does still have potential in legal information retrieval. Using a pre-trained model is a good option, or switching to word embedding. The downside of word embedding is that it is computationally very expensive, but it may be worth the trouble. Word2vec, fasttext and GloVe are all good options to try as word embedding models.
Another option is to improve the semantics of the system. Word sense disambiguation is about retrieving the correct meaning of a word out of the context. If words with multiple meanings (club) are substituted by a correct synonym that is not ambiguous (knobstick or association), this could improve the system since documents that do contain the polysemous words won't necessarily match on these words.
More research is needed on classifiers, such as Support Vector Machines or Gradient Boosting, in order to possibly improve the results.

Bibliography

[1] Legal Information Institute, Cornell Law School, "Wex, lii's community-built, freely available legal dictionary and legal encyclopedia." https://www.law.cornell.edu/wex/precedent.

[2] Rechtbank Amsterdam, "ECLI:NL:RBAMS:2016:2573." http://deeplink.rechtspraak.nl/uitspraak?id=ECLI:NL:RBAMS:2016:2573.

[3] Rechtbank Amsterdam, "ECLI:NL:RBAMS:2013:BZ3203." http://deeplink.rechtspraak.nl/uitspraak?id=ECLI:NL:RBAMS:2013:BZ3203.

[4] Rechtbank Amsterdam, "ECLI:NL:RBAMS:2018:5226." http://deeplink.rechtspraak.nl/uitspraak?id=ECLI:NL:RBAMS:2018:5226.

[5] Rechtbank Amsterdam, "ECLI:NL:RBAMS:2016:6891." http://deeplink.rechtspraak.nl/uitspraak?id=ECLI:NL:RBAMS:2016:6891.

[6] Rechtbank Midden Nederland, "ECLI:NL:RBMNE:2018:827." http://deeplink.rechtspraak.nl/uitspraak?id=ECLI:NL:RBMNE:2018:827.

[7] S. A. Lastres, "Rebooting legal research in a digital age," 2015.

[8] Justitia, "Juridische portals." https://www.justitia.nl/juridische-portals.html.

[9] T. Bench-Capon, M. Araszkiewicz, K. Ashley, K. Atkinson, F. Bex, F. Borges, D. Bourcier, P. Bourgine, J. G. Conrad, E. Francesconi, *et al.*, "A history of ai and law in 50 papers: 25 years of the international conference on ai and law," *Artificial Intelligence and Law*, vol. 20, no. 3, pp. 215–319, 2012.

[10] M. Van Opijnen and C. Santos, "On the concept of relevance in legal information retrieval," *Artificial Intelligence and Law*, vol. 25, no. 1, pp. 65–87, 2017.

[11] T. Saracevic, "Relevance reconsidered," in *Proceedings of the second conference on conceptions of library and information science (CoLIS 2)*, pp. 201–218, ACM New York, 1996.

[12] T. Saracevic, "Relevance: A review of the literature and a framework for thinking on the notion in information science. part iii: Behavior and effects of relevance," *Journal of the American Society for information Science and Technology*, vol. 58, no. 13, pp. 2126–2144, 2007.

[13] D. J. MacKay and L. C. B. Peto, "A hierarchical dirichlet language model," *Natural language engineering*, vol. 1, no. 3, pp. 289–308, 1995.

[14] L. Tian, H. Ning, L. Kong, Z. Han, R. Xiao, and H. Qi, "Hljit2017@ irled-fire2017: Information retrieval from legal documents,"

[15] A. Mandal, K. Ghosh, A. Bhattacharya, A. Pal, and S. Ghosh, "Overview of the fire 2017 irled track: Information retrieval from legal documents,"

[16] V. Tran, M. L. Nguyen, and K. Satoh, "Automatic catchphrase extraction from legal case documents via scoring using deep neural networks," *arXiv preprint arXiv:1809.05219*, 2018.

[17] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[18] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[20] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[21] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, pp. 1188–1196, 2014.

[22] Elasticsearch BV, "Elasticsearch the distributed, restful search and analytics engine." https://www.elastic.co/products/elasticsearch.

[23] The Apache Software Foundation, "Apache lucene scoring algorithm." http://lucene.apache.org/core/index.html.

[24] T. K. Ho, "Random decision forests," in *Document analysis and recognition, 1995., proceedings of the third international conference on*, vol. 1, pp. 278–282, IEEE, 1995.

[25] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, NY, USA:, 2001.

[26] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of research and development*, vol. 2, no. 2, pp. 159–165, 1958.

[27] R. Krovetz, "Viewing morphology as an inference process," in *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 191–202, ACM, 1993.

[28] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: a comparison of retrieval performances," *Lecture Notes on Software Engineering*, vol. 2, no. 3, 2014.

[29] COLIEE 2018, "Competition on legal information extraction/entailment." https://sites.ualberta.ca/ miyoung2/COLIEE2018/.

[30] JURISIN 2018, "Twelfth international workshop on juris-informatics." http://research.nii.ac.jp/jurisin2018/.

[31] Textacy, "Make a single spaCy-processed document from 1 or more chunks of text." https://chartbeat-labs.github.io/textacy/api_reference.html#textacy.spacier.utils.make_doc_from_text_chunks.

[32] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," *arXiv preprint arXiv:1607.05368*, 2016.

[33] R. Mochales and M.-F. Moens, "Argumentation mining," *Artificial Intelligence and Law*, vol. 19, no. 1, pp. 1–22, 2011.

[34] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[35] R. Levy, Y. Bilu, D. Hershcovich, E. Aharoni, and N. Slonim, "Context dependent claim detection," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 1489–1500, 2014.

[36] M. Lippi and P. Torroni, "Context-independent claim detection for argument mining.," in *IJCAI*, vol. 15, pp. 185–191, 2015.

Appendices

Appendix A

Queryless Search Visuals

Uitgebreid zoeken

PERSOONLIJK FILTER INSTELLEN | ATTENDERING

Zoeken in alle bronnen

FILTERS COMBINEREN

Nederland

Rechtspraak	911.840
Wet- en regelgeving	481.543
Officiële publicaties	1.401.134
Toezichthouders	54.080
Alternatieve geschiloplossing	73.994
Overig	169.860
Verdragen	107.511

Lokaal

Regelingen	20

Regionaal

Regelingen	1.994

Caribisch Nederland

Rechtspraak	8.631

Europa

Internationale Overeenkomsten	14.568
Rechtspraak	251.708
Wet- en regelgeving	247.762
Officiële publicaties	466.250
Toezichthouders	38.899
Verdragen	9.753
Overig	432

Internationaal

Rechtspraak	8.978
Toezichthouders	12
Alternatieve geschiloplossing	469
Overig	26.118

Zoeken op termen

Zoeken

Moet voorkomen — Zoekterm(en)

Mag voorkomen — Zoekterm(en)

Moet niet voorkomen — Zoekterm(en)

Woorden in titel

Achternaam auteur

Datum — datum publicatie

van DD-MM-JJJJ — tot en met DD-MM-JJJJ

Sleep documenten hier

of

Selecteer van uw pc

Zoek binnen

Zoeken | Alle velden leegmaken

Opslaan als attendering

Uitgebreid zoeken

PERSOONLIJK FILTER INSTELLEN ATTENDERING

Zoeken in alle bronnen

FILTERS COMBINEREN

Nederland

Rechtspraak	911.840
Wet- en regelgeving	481.543
Officiële publicaties	1.401.134
Toezichthouders	54.080
Alternatieve geschiloplossing	73.994
Overig	169.860
Verdragen	107.511

Lokaal

Regelingen	20

Regionaal

Regelingen	1.994

Caribisch Nederland

Rechtspraak	8.631

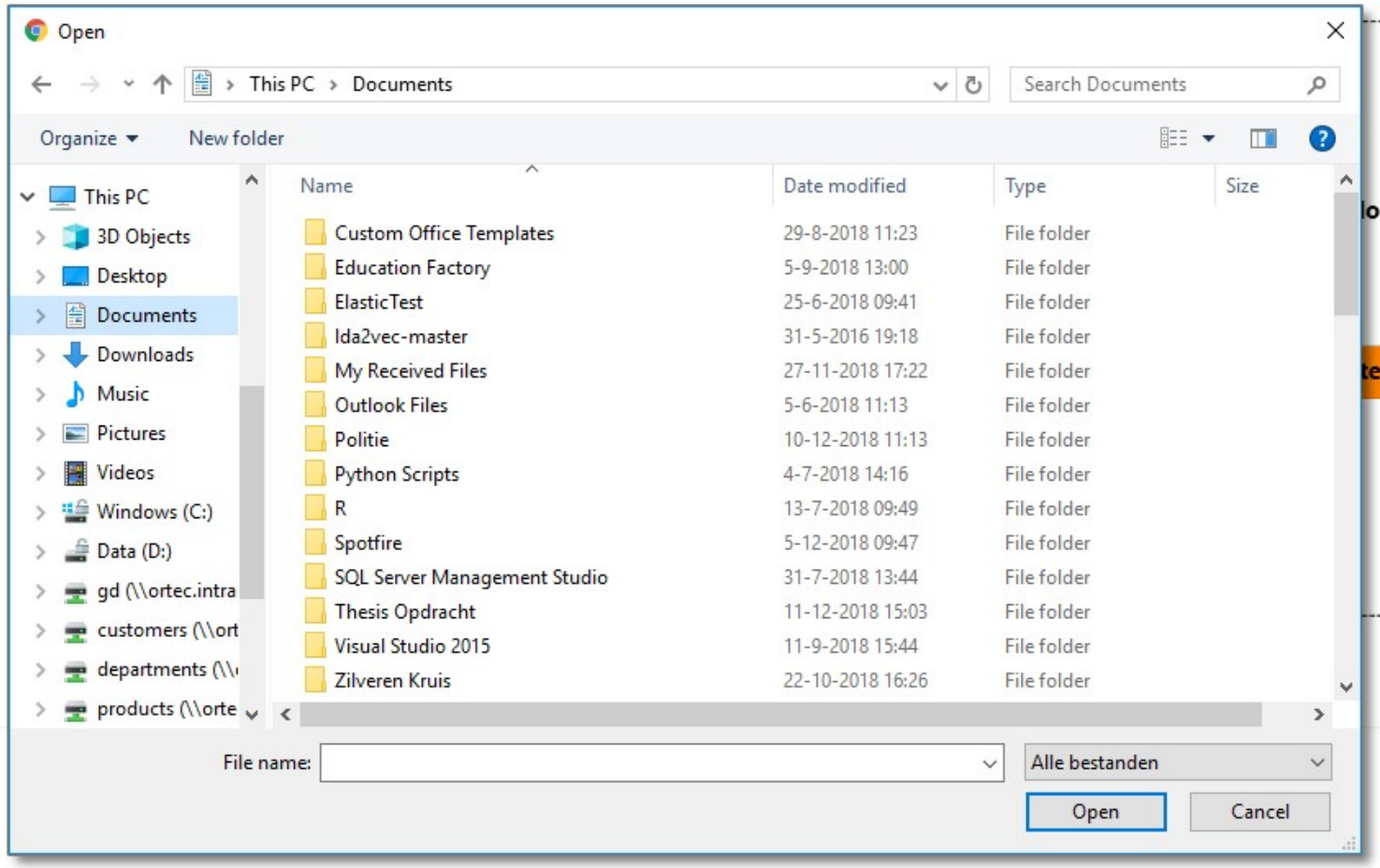
Europa

Internationale Overeenkomsten	14.568
Rechtspraak	251.708
Wet- en regelgeving	247.762
Officiële publicaties	466.250
Toezichthouders	38.899
Verdragen	9.753
Overig	432

Internationaal

Rechtspraak	8.978
Toezichthouders	12
Alternatieve geschiloplossing	469
Overig	26.118

Zoeken op termen



ocumenten hier

of

teer van uw pc

Opslaan als attendering

Alle resultaten > 2010-heden

94.788 Resultaten gesorteerd op Relevantie

ATTENDERING

FILTERS COMBINEREN

Nederland
- Rechtspraak 48.424
- Wet- en regelgeving 313
- Officiële publicaties 3.327
- Toezichthouders 389
- Alternatieve geschiloplossing 4.186
- Overig 1.429
- Verdragen 35

Regionaal
- Regelingen 1

Caribisch Nederland
- Rechtspraak 735

Europa
- Internationale Overeenkomsten 21
- Rechtspraak 2.505
- Wet- en regelgeving 273
- Officiële publicaties 599
- Toezichthouders 2
- Overig 9

Internationaal
- Rechtspraak 390
- Alternatieve geschiloplossing 15

Bron
- Tijdschriften 31.138
- Nieuwsbrieven 876
- Overheidspublicaties 41.049
- Naslagwerken 16.351
- Boeken 89
- Commentaar 3.175
- Modellen 198
- Blogs 801

1 | JIN 2018/209 - Met noot - Vaan de, R.A.G. e.a. - Hoge Raad - 12-10-2018
Aansprakelijkheidsrecht, Rechtspersonenrecht, Onrechtmatige daad, Concernrecht, Onrechtmatige overheidsdaad

2 | Rechtspraak.nl - Rechtbank Zeeland-West-Brabant - 24-10-2018 - ECLI:NL:RBZWB:2018:6539 - Burgerlijk recht
Cessie, subrogatie, onrechtmatige daad

3 | JIN 2018/206 - Met noot - Everhardus, J.R. - Gerechtshof 's-Hertogenbosch - 21-08-2018
Bestuurdersaansprakelijkheid, Afgeleide schade, Profiteren van onrechtmatige daad

4 | Rechtspraak.nl - Gerechtshof 's-Hertogenbosch - 05-06-2018 - ECLI:NL:GHSHE:2018:2378 - Burgerlijk recht
Onrechtmatige daad.

5 | Rechtspraak.nl - Rechtbank Limburg - 07-11-2018 - ECLI:NL:RBLIM:2018:10406 - Verbintenissenrecht
Letselschade; steekpartij; onrechtmatige daad; geen noodweer(-exces); geen eigen schuld.

6 | Rechtspraak.nl - Rechtbank Overijssel - 03-07-2013 - ECLI:NL:RBOVE:2013:1491 - Burgerlijk recht
Onrechtmatige daad.

7 | O&A 2018/51 - Met noot - Oztürk, T.G. e.a. - Gerechtshof Den Haag - 09-10-2018
Onrechtmatige daad bij het behalen van klimaatdoelstellingen. Schending zorgplicht ex artikelen 2 en 8 EVRM.

8 | JIN 2018/161 - Met noot - Everhardus, J.R. - Rechtbank Midden-Nederland - 18-07-2018
Bestuurdersaansprakelijkheid, Onrechtmatige daad, Beklamelcriterium

Appendix B

Summary Example

Bradley was a member of the armed forces who fell while showering on board his ship. He applied under s. 21(2) of the Pension Act for benefits relating to upper back and neck injuries caused by the accident. The Veterans Review and Appeal Board dismissed the claim because the accident was not sufficiently connected with military service as required by s. 21(2). Bradley made a new claim related to low back pain caused by the same accident. The Minister of Veterans Affairs indicated that, because Bradley's condition resulted from an accident that had previously been determined as not directly connected to service, she had no jurisdiction under s. 85(1) of the Pension Act to proceed with the application. Bradley sought judicial review, requesting an order of mandamus to compel the Minister to determine the application.

Appendix C

Facts Example

[1] Phelan, J.: This Applicant, a self-represented former member of the Armed Forces, has been locked in litigation over his disability claim for a slip in the shower of HMCS Qu'Appelle for several years and in this Court since 1999 (see Bradley v. Canada (Attorney General), [1999] F.C.J. No. 144). His first application for a pension based on upper back and neck injuries was finally settled in 2004.
[2] After the 2004 Federal Court decision, the Applicant started a second application for mechanical lower back pain based on the same incident. After further proceedings it was determined that the Department of Veterans Affairs was required to determine this new claim. It is the new claim which is the subject matter of this judicial review.
[3] The Applicant was undergoing officer training as an Acting Sub-Lieutenant aboard HMCS Qu'Appelle, a destroyer escort. The ship was alongside in Vancouver near the end of a training cruise in the Pacific. The ship was ultimately destined to its home port of CFB Esquimalt.
[4] The Applicant, having completed training for the day, went to the mess (presumably the Wardroom) where he had a few beers. The exact quantity was not established but the drinking took place on board a warship in a tightly regulated environment.
[5] Mr. Bradley contends that he slipped while in the shower cleaning up from his daily duties and prior to going ashore. He was found the next day in his bunk in severe pain. There was a notation in the medical file that he had significant alcohol in his system. There was some suggestion drawn that there was a connection between the shower incident and the consumption of alcohol on board; however, how that could have happened in the circumstances of an officer in training, in a wardroom, was not established.
[6] When the Department finally considered this new claim, it reached the conclusion that the fall in the shower did not arise out of, and was not directly connected with, the Applicant's military service. The Department made no further determination as to whether the fall caused the mechanical low back pain.
[7] A Review Panel affirmed the Department's decision to deny the Applicant pension entitlement.
[8] The Applicant appealed to the Appeal Board; however, on August 5, 2008, the Appeal Board affirmed the negative decision. It is this decision which is the subject of this judicial review.

Appendix D

Topic List Interview

Interview Overview

1. Introduction
    - Law Firm
    - Field of Law
2. Current Situation
    - Duration Search
    - Missed Precedence
3. Reaction to the Proposed System
    - Show Visuals and Explanation of Queryless Search
    - Initial Response
    - Idea of Queryless Search
    - Potential of Queryless Search
4. Suggestions for Improvement
    - Usability
    - Desirability
5. Finalization
    - Anything Missed in Interview