University of Glasgow | School of Computing Science

Public vs. Private: Identifying Public Domain Knowledge

Kelvin Fowler
2083905f

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Level 4 Project — April 29, 2017
Joint Honours — 20 Credits

Abstract

The current system of sensitivity review used in the archival process of government documents cannot keep up with the rapid creation of new content. The process is also cumbersome and insecure. Information Retrieval techniques can be leveraged to vastly improve this operation. This project provides a prototype of a system and an interface, incorporating these information retrieval techniques, to allow archivists to identify information from documents which already exist within the public domain. Evaluation deemed the prototype to be fairly successful, however various shortcomings were discovered, which could be addressed in future work.

Acknowledgements

I would like to thank my supervisors, Dr. Craig Macdonald and Graham McDonald, for their continued guidance throughout the course of this project.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: ______________ Signature: ______________

Contents

Chapter 1

Introduction

1.1 Aims

This project aims to deliver a prototype of a system which can be used to assist the process of identifying public domain knowledge within potentially sensitive government documents. The project will use information retrieval (IR) techniques to present public domain documents which could help archivists to make an informed choice during sensitivity review. An archivist then can compare information in government documents to information in the public domain in order to ensure no new information is being inadvertently released.

The public domain is defined as that information which is available to the public, without legal restrictions, such as news articles.

1.2 Motivations

Document review within government archives faces major problems today. There is a constant barrage of new documents which must be archived and eventually reviewed, driven by the rapid creation of new artefacts through technology. Each email and memo is now automatically saved and must, at some point, be reviewed.

Further to this, the current system of document review involves no integrated public domain knowledge checking process. Often the substitute for this is a simple web search, which poses many security risks. The search terms themselves could be very sensitive and yet they are being exposed to the internet, and as such, are vulnerable to unintended discovery.

Understanding this, the final product should be a system which can keep up with the vast number of documents which need to be reviewed within archives. A system which is encapsulated and secure when it comes to further research on the content of a document is also sought.

Interestingly, this field has experienced some media coverage in recent times, with certain current affairs being directly linked to sensitive documents. In the case of the Hillary Clinton email scandal, the FBI used technology to aid their process in reviewing these emails for sensitivities [29].

Combining these concerns this project strives to produce a highly practical and helpful system, which can inspire and encourage further work into this problem domain. To understand further motivations and context behind this project, see Chapter 2.

1.3 Terminology

Throughout this document certain terms will be used which may not be familiar. A **Source Document** is the document which is being reviewed. It is the document from which queries are generated for IR. A **Target Document** is a document in the public domain which may contain information relevant to the review of a source document. These are the documents are which are to be retrieved.

1.4 Structure

This dissertation will take the following structure. Chapter 2 investigates the project's relevance through review of some related literature. Next, the planning, requirements and design of the system are discussed in Chapter 3. Chapters 4 and 5 detail the implementation challenges of the system's components. Chapter 6 explains the experimentation and evaluation undertaken to determine the systems effectiveness. The dissertation concludes with learnings and potential future work in Chapter 7.

Chapter 2

Related Literature

The Freedom of Information Act (2000) allows for all documents held by government to be accessed by the public, upon request, with certain exemptions [7]. These exemptions could be negative sentiment towards another country or the identity of confidential informants, or any other specific sensitivity. With these exemptions in mind, government must decide if any given document can be released to the public. The current sensitivity review procedures used by government archivists are less then ideal. Paper documents must be read in full and much interdepartmental conversation must take place before a decision regarding sensitivity can be made. Digital records, of which there are constantly increasing numbers, only make this process even more complicated. This explosion in magnitude poses many potential problems for archiving organisations, such as "What to Keep?" and how to effectively identify sensitivities [39]. The current process is unreasonable and impractical in today's digital context [27].

2.1 Sensitivity Review

Some work has already been done to begin to tackle this issue of digital sensitivity review. Prior to review a "classifier" may be used to automatically identify potential sensitivities through use of named entities and document sentiment [36]. This type of work can be used to lower the burden on reviewers by attempting to identify documents which may be of the most concern.

In other research into the challenges and potential avenues for improving the digital sensitivity review process it was found that document reviewers are reluctant to trust technology alone to identify sensitivities, and that a manual review process will likely always be necessary [31]. As such, technology which can assist the manual review process (such as that developed in this project) will be increasingly important as the review process is improved and iterated upon.

Sensitivity review is a category of Technology Assisted Review, a field which aims to provide solutions to document review problems using technology. Another, very successful, application of Technology Assisted Review is E-Discovery. This is the discovery of all relevant documents pertaining to the opposing party in a civil litigation case. The idea being that each party should have access to all documents that they are legally entitled to view, so as to gain any possible advantage. Information retrieval has been used extensively to provide E-Discovery services, and is an extremely successful and high profile application of this technology [40]. E-Discovery is a high recall task, which means it focusses on returning all of the relevant documents rather than returning the most relevant ones. At a high-level, sensitivity review is also a high recall task, in that all sensitive documents must be found. In the case of this project, however, identifying public domain knowledge is a high precision task. Only one relevant target document is needed to identify the contents of a source document as existing in the public domain.

2.2 Information Retrieval

The intended final product of this project acts as a highly focussed automatic search engine for public domain documents. To facilitate this, some underlying search engine must be in place to process queries and return results. This is done through a search engine specifically designed for this type of information retrieval. Two such search engines are Terrier and Lucene [33, 23, 1]. These search engines are capable of performing many different tasks outside of the realm of this project, but are configurable to each specific use case. This type of technology is also used extensively in the works discussed above.

Further to searching, it is important to link the information in a target document to the public domain in some way. There has been some work into automatically identifying key named entities and concepts witthin documents. One such work is Wikify [38]. Wikify automatically hyperlinks phrases and entities to their relevant Wikipedia articles [26]. A system which can understand and identify the best and most important entities within a document is extremely useful in the context of this project, where some kind of searching based on the content of a given source document is a key aim.

This project differs somewhat from the standard search task familiar to most people, that of a free-form query entered by the user, as exemplified by Google [11]. This type of searching is known as Ad-Hoc Retrieval [28]. The queries formed in this project are, instead, formulated as subsets of the collection of terms a source document.

2.3 The Need for this Project

In recent times, a great deal more files have been generated by government than ever before. E-mail and digital documents are automatically archived and all must be reviewed and released to the public as mandated by law. This project seeks to address and help remedy some of these issues by providing efficient and effective tools to ease the digital transition. These needs are summarised below:

Need.1 Efficient and effective tool to review digital documents.

Need.2 An easier and better way to review public domain knowledge in the context of a source document.

Need.3 Encapsulated environment to do sensitive public domain research.

Chapter 3

Requirements and Design

Before beginning implementation, some initial planning was performed to understand the key requirements of the system. This chapter outlines these requirements as well as those requirements that materialized through the course of the project. Further to this, the architecture and technologies used in the project are introduced.

3.1 Requirements

Initially the project's requirements were elicited from the project supervisors, who have extensive knowledge of the problem domain, through related research and a working relationship with the The National Archives of the UK and Scotland. These initial requirements allowed the developer to begin implementation, after which more detailed requirements were identified through discussion and experimentation. The progress and requirements were reviewed at weekly meetings, however the process of implementation was fairly fluid, allowing for experimentation and changing priorities. This loosely follows an agile approach, however it is not particularly important to label the process so specifically [41].

3.1.1 Functional Requirements

These functional requirements were compiled throughout the course of the project and evolved as features were experimented with. The functional requirements of the project were categorised using the MoSCoW method (**Must Have**, **Should Have**, **Could Have** or **Would Like to Have**) [42]. These were further ratified through user stories, which means summarising the requirement through a user scenario using the template "*As a..., I want to..., So that*" [42]. User stories help us to understand why a requirement is necessary and how it benefits the end user. Further to this a more detailed use case to contextualize the following requirements can be given:
"Bob is an archivist at the national archives. He must review a collection of emails for sensitivities. Using the new tool he loads up the collection of the emails. When he opens the first email, some relevant news articles are automatically retrieved at the same time. He reads some of the source document and also the top news article. This news article confirms that the content of the email is already in the public domain. Bob classifies the email as not sensitive and opens the next email to repeat the process."

Must Have These are the requirements the the project must fulfil to be considered successful.

M.1 Allow user to identify when a document contains public domain knowledge.
As a document reviewer,

I want the ability to decide that a document contains public domain knowledge
So that I can make a decision regarding its status.

M.2 Frontend user interface displaying source document for review and target documents automatically identified by the software.
As a document reviewer,
I want to view both the document up for review and associated public domain documents
So that I can easily compare the contents of both and see what is known in the public domain.

M.3 Trials of different retrieval methods and some form of quantitative analysis based on these.
As a document reviewer,
I want to be confident that this software will return the most relevant results in reasonable time,
So that I can trust the service to return excellent related documents.

Should Have These are the requirements which are not vital but are to be fulfilled if possible.

S.1 Generate suggested queries to allow user to refine returned related documents.
As a document reviewer,
I want to refine search terms for related documents,
So that more relevant documents are returned.

S.2 Ability to run custom queries.
As a document reviewer,
I want to write my own queries,
So that I can retrieve the refine my queries for better results.

S.3 Date range searches.
As a document reviewer,
I want to limit date range for retrieval of related documents,
So that I can ensure relevance of related documents.

Could Have These requirements should only be implemented if all of the above requirements have been completed.

C.1 Machine learning to improve system performance over time based on usage by reviewers.
As a document reviewer,
I want my actions using the system to improve its performance to better suit my needs,
So that it is easier to find related documents that I would deem relevant.

Would Like To Have These are requirements which are outside the scope of this project and are not deemed important enough to be implemented within the current time frame.

W.1 Wikify! Like functionality to automatically highlight and link to concepts in source documents which have Wikipedia articles associated with them.
As a document reviewer,
I want to have easy access to Wikipedia articles of items mentioned in the document I am reviewing,
So that I can easily find out more about concepts mentioned in documents.

W.2 Eye tracking to automatically identify points of interest in the source document from which to formulate queries.
As a document reviewer,
I want the system to react to what my eyes are drawn to,
So that retrieved target documents are dictated by my actions.

3.1.2 Non-Functional Requirements

Non-Functional requirements are those requirements that do not define the specific behaviour of the system, but rather how the system performs.

NF.1 Intuitive user interface.

NF.2 Results must be relevant.

NF.3 Must return results in reasonable time.

NF.4 Scalability to handle large amounts of files.

NF.5 Robust and fault tolerant.

3.2 Design and Architecture

The application is based on a client-server design. More specifically, a web application acts as a client which interfaces with a RESTful API acting as a server. This paradigm was chosen so as to abstract all of the complicated logic away from the system presented to the user. This also meant that implementation could be split across the two distinct components, simplifying development. The client communicates with the server using HTTP requests which request information from the server or update information on the server. An overview of the system can be seen in Figure 3.1 with details following in the sections below.

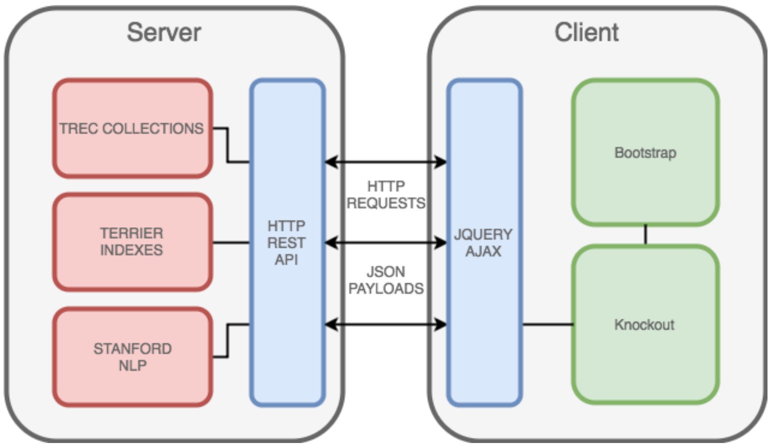


Figure 3.1: Overview of System Architecture

3.2.1 Server Architecture

Information Retrieval The project depends on some IR system so that the Must Have requirements can be fulfilled (specifically, **M.3**). The IR system chosen for these needs was Terrier. Terrier is a open-source search engine with an extensive Java API allowing it to be easily used for development of various IR tasks [23, 33]. Terrier, being developed at The University of Glasgow, was the best and obvious choice of IR technology for the project. The local expertise meant that problems encountered could be easily resolved or explained, ensuring minimal downtime due to learning. Other choices could have been Apache Lucene or MG4J [37].

Programming Language As Terrier had already been chosen as the IR tool for the project, it was necessary that the supporting system was written in Java, in order that it could interact with the Terrier Java API.

Persistence One fundamental aspect of Terrier is the concept of indexes. These are files containing the important information from a collection of documents necessary for information retrieval. This provides a persistent data storage mechanism which can be used to store additional information about documents or groups of documents. This kind of data is saved in a *meta-index* within a normal index.

Terrier is able to handle TREC test collections, which allow documents to be defined through tags, much like XML [24]. Since Terrier is easily configurable to a custom version of this TREC document style it was decided that the documents pertinent to the project should be arranged like this. This provides an opportunity to store additional information within new tags in each `.trec` document. This technique is used to save information too large for the Terrier meta-index. These techniques produce a fully functional state system capable of retaining all the necessary information needed for the system.

This design entirely eliminates the need for a DBMS (Database Management System) like PostgreSQL and the complicated Java Object to Database Model mapping that often comes with such designs.

Natural Language Processing (NLP) As Terrier mandated the use of Java, any NLP framework chosen was required to have a Java API. Stanford Natural Language Processing was used as an NLP framework to identify terms from which to form effective queries [35]. The Java API allows text to be analysed and annotated. More specifically, the named entity identification capability of StandordNLP was used [30]. Stanford NLP was initially chosen due to its preferable documentation. It also offers the ability to identify specifically which tokenisers to use when processing some text, allowing the developer to streamline the NLP process to their specific use-case.

RESTful API Jersey is a framework which provides a reference implementation of the JAX-RS API as defined by Oracle, allowing a REST API to be developed [13, 12]. Jackson is used alongside Jersey to provide support for JSON [17]. This allows the frontend and backend to communicate using one format. It handles the conversion of JSON to Java Objects and vice versa. This set-up allows one to define a fully functional HTTP REST API which can consume and produce JSON.

3.2.2 Client Architecture

The web application maintains no state between usages, but rather requests all necessary data from the server upon loading. Having no state system in the client application greatly reduces the complexity of the web application.

Model View ViewModel The architecture of the frontend user interface is based around the Model View ViewModel (MVVM) design pattern. Facilitated by the JavaScript library Knockout.js, MVVM allows us to consolidate the logic associated with the DOM (Domain Object Model) into one place [16]. This vastly simplifies the automatic update of DOM elements based on JS variables. This lends itself well to the problem domain of many rapidly changing documents and results sets as the user identifies relevant documents.

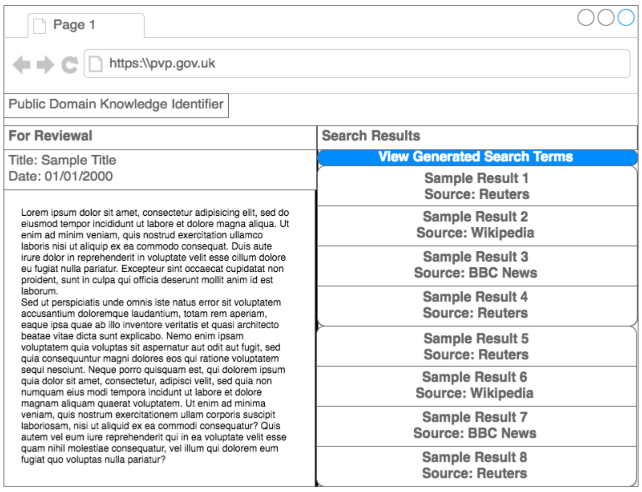
There are various other libraries that allow for similar designs, however developer familiarity with Knockout's syntax and best practices meant that development was easy and fast. This was important as the learning curve for the backend technologies required more attention.

Style Although our data manipulation is handled by Knockout, information must also be presented in an intuitive and user friendly way. There are many frameworks which provide pre-built components to build user interfaces but Bootstrap was chosen mainly due to familiarity reasons. It provides myriad customisable components, as well as being extremely well documented [3]. This dramatically simplifies the rapid development of a user interface.

AJAX JQuery is used in the frontend to interface with the backend through HTTP (AJAX) requests [14]. JQuery provides a very well documented and widely adopted API for this. JQuery is a dependency of both Knockout and Bootstrap and so utilizing its already present features was sensible and easy.

Web Application Framework The web application currently uses an Express.js server running on Node.js [6, 18]. This is not particularly necessary as the web application could, with some refactoring, be served through a static web server such as GitHub Pages [4].

Interface Design Draw.io was used in the initial stages of design to wireframe ideas for how the web application should look [10]. An example of this can be seen in Figure 3.2.



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Figure 3.2: An Initial Wireframe for the System

3.3 Summary

This chapter has detailed the requirements of the system and provided some high level explanation of the architecture used to fulfil these. The next two chapters look at how this design was implemented and the challenges that arose during this. Chapter 6 also investigates if the requirements were fulfilled and how effective the implementation of the system was.

Chapter 4

Server Implementation

The documents pertinent to the project (both source and target) are stored within the server file system as `.trec` files. There is one document per file. This Chapter will discuss the implementation of the server component of the system. It will cover in detail the use of Terrier to index and retrieve target documents as well as how queries were generated. Also covered is the implementation of the RESTful API to allow communication with the client.

4.1 Tools

Git and GitHub Git was used, alongside GitHub as Source Control Management (SCM) for the project [8, 9]. This allows one to maintain version history for the project. Hosting the repository on GitHub also served as a back-up for the project, and allows it to be cloned onto any machine.

Trello Trello was used throughout the project to plan implementation steps and track bugs [25].

4.2 Dependency Management and Build System

Terrier and StanfordNLP were the two major dependencies of this project. They both use Maven as a build tool and so Maven was chosen as a dependency management and build system [2]. This was motivated by the ability to easily include Terrier and StanfordNLP as dependencies using Maven. Further to this, it is easy to use and has clear documentation. It also allowed for inclusion of all other dependencies needed for the project such as Jackson and Jersey. Maven's build system also ensures all unit and integration tests pass before building the system into a runnable jar file.

4.3 REST API

The HTTP REST API is the invocation point for most of the behaviour described below, with the exception of target document indexing. The use of different HTTP verbs in different circumstances allows for varied requests. `POST` allows a request body to be defined, which allows for more information to passed to the server. The `GET` verb does not allow for this, so all information must be passed in the url. This is useful in the method used to query the target index: `POST api/query`, since the query can be too long to fit in a URL alone.

Jackson allows us to serialize Java objects to JSON and vice-versa, which is used extensively throughout the API to send and receive information. Serializing an object as a response body is trivial and can be seen in Code Sample 4.1. Analogously, Jackson can also convert JSON to a Java object, provided the variables are named consistently between both.
The API also returns errors with the correct HTTP error codes if necessary. This is used in `POST api/topic` to return a `400 BAD REQUEST` error in the case that a repeated topic is sent to be recorded.

```
Indexes indexes = new Indexes(SOURCE_INDEXES_PATH);
  return Response
   .ok()
   .entity(indexes)
   .header("Access-Control-Allow-Origin", "*")
   .build();
```

Code Sample 4.1: Returning a Response from within an API method

4.4 Document Preparation

Before IR tasks can be performed and return useful results the collections of documents must first be properly prepared and analysed. This involves forming queries from source documents, indexing target documents and saving our data where it can be easily accessed.

4.4.1 Indexing with Named Entity Identification

All documents (both source and target) are indexed by the system using Terrier. This produces a collection of *terms* for each document, which can be used to analyse the contents of our documents. The entity relationship diagram in Figure 4.1 summarises this.

Named Entity Tokeniser Terrier can use any subclass of its `Tokeniser` class in its indexing process to generate the terms saved in an index. In order to find named entities within documents a custom subclass of the `Tokeniser` class was created which uses Stanford NLP to annotate terms in text which it recognises as named entities. If a term was identified to be a named entity by Stanford NLP then an underscore and the named entity type was appended to the terms. For example, `kelvin` becomes `kelvin_person`. The code that achieves this can be seen in Code Sample 4.2 The logic for annotating a section of text was further abstracted to a separate class so that it could be reused to annotate any text, outside of the context of tokenisation during indexing. This was used specifically to annotate subject queries in the query generation phase, but could provide further applications in the future.

```
String word = token.get(TextAnnotation.class);
String ne = token.get(NamedEntityTagAnnotation.class);
if (!"O".equals(ne)){
  neString.append(word).append("_").append(ne);
  tokens.add(neString.toString());
} else {
  tokens.add(word);
}
```

Code Sample 4.2: Annotating Text with Named Entities

Classifiers and Distributional Similarity An instance of `StanfordCoreNLP` must be created to use the named entity annotation features. In this instance the fewest possible annotators were used to achieve named entity recognition. The simplest classifier model provided by StanfordNLP which only looks for Location, Organisation and Person named entities is used. This classifier was chosen to focus research only on these three most important named entity types, rather than the other potential options of “Numerical" and “Temporal" named entities [20].

As discussed in the evaluation section, Named Entity models with and without Distributional Similarity (DistSim) Features were tested (the documentation suggests this can improve performance while sacrificing efficiency [21]).

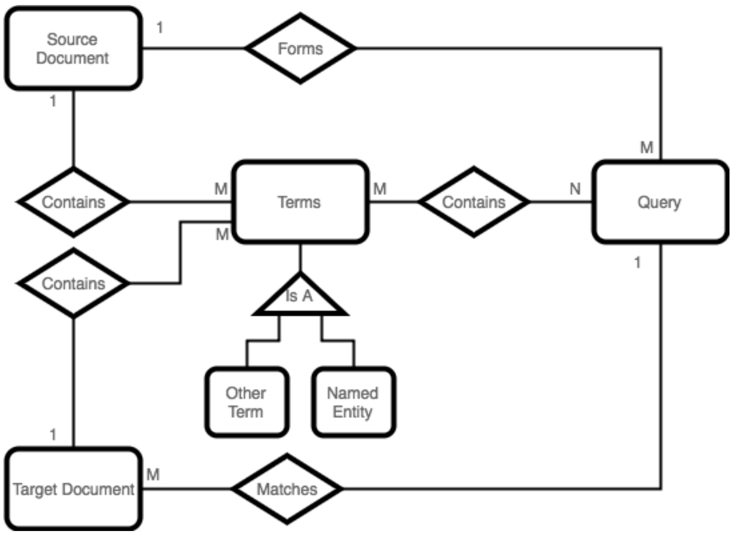


Figure 4.1: Entity Relationship Diagram for Documents

4.4.2 Source Documents

Before a collection of source documents can be presented for review by the system, they must first be analysed to procure the queries necessary to perform information retrieval.

Query Generation Having indexed the collection of source documents, there now exists a list of terms for each document, which can be retrieved at will. This list contains some terms which are tagged as named entities. This can be used to produce different formulations of queries for each source document. This process of query generation is invoked automatically immediately after indexing of a source document collection. Created are 4 queries: *All Terms Query*, *Named Entity Query*, *Tf-Idf Named Entities* and *Subject Query*. Throughout this section the document shown in Figure 4.2 will be used to give examples of the generated queries. The code shown in Code Sample 4.3 displays the steps takes to formulate the *All Terms Query*, *Named Entity Query* and *Tf-Idf Named Entities Query*.

```
<DOC>
...
<SUBJECT>UK WILL NOT PARTICIPATE IN IRAN'S NUCLEAR CONFERENCE, WILL DISCOURAGE OTHERS
FROM ATTENDING REF: STATE 112229 </SUBJECT>
UK WILL NOT PARTICIPATE IN IRAN'S NUCLEAR CONFERENCE, WILL DISCOURAGE OTHERS FROM
ATTENDING REF: STATE 112229
Classified By: Political Counselor Rick Mills for reasons 1.4 b and d (C)
```

```
 HMG does not intend to participate in Iran's November 30 nuclear conference, since
 attendance would "give credibility" to Iran's claims about its nuclear activities,
 Foreign and Commonwealth Office (FCO) Counterproliferation Department officer Lesley
 Craig told Poloff October 24. Craig said the FCO will tell anyone who contacts HMG about
 the conference, such as academics and NGOs, that it would be advisable not to attend. She
 added that the UK probably would not do a "formal demarche" to other countries regarding
 non-attendance; she suggested that countries likely to attend the conference would not be
 amenable to a demarche in any event. Separately, FCO Iran Coordination Group Regional
 Team Leader Rachel Martinek made comments to Poloff consistent in substance with all
 Craig's points. Visit London's Classified Website: XXXXXXXXXXXX TUTTLE
</DOC>
```

Figure 4.2: An Example Source Document

All Terms Query To create the *All Terms Query* the list of terms in the index for a given document is iterated over and each term is appended to a string. This is very crude and simple query, however it contains all of the information present in the document. The consequence of this is that the query is very long. A sample of the *All Terms Query* for the document shown in Figure 4.2 can be seen in Figure 4.3 with the full version available in Figure A.1. It can be seen that there are a mixture of named entities and other terms in this query.

```
s countries lrb  state_location who  rrb   will 24 participate made tell event 30 told
attending  give suggested demarche iran_location added officer ref foreign_organization
department_organization likely formal  nuclear group_organization reasons conference
```

Figure 4.3: An Sample of an All Terms Query

Named Entities Query Producing the *Named Entities Query* is similar to the *All Terms Query*. The collection of all terms for a document is iterated over, but this time only those terms which contain an underscore are selected, the identifier for a named entity found during indexing using the `NamedEntityTokeniser`. This query obviously lacks the non named entities terms in the document, and so is much shorter. The motivation here was to understand if the named entities alone were enough to produce meaningful results. A sample of the *Named Entities Query* for the document shown in Figure 4.2 can be seen in Figure 4.4 with the full version available in Figure A.2. This query is merely the above query with all non named entity terms removed.

```
state_location iran_location foreign_organization department_organization
group_organization uk_location political_organization office_organization
regional_organization coordination_organization poloff_person c_organization
```

Figure 4.4: An Sample of a Named Entities Query

Tf-Idf Named Entities Query The *Tf-Idf Named Entities Query* leverages the Terrier API to retrieve all the data necessary for the calculation. Tf-Idf (Term Frequency - Inverse Document Frequency) can be defined as:

$$Tf\text{-}Idf = Tf \cdot Idf$$

where:

$$Tf = \frac{Number\ of\ Times\ Terms\ Appears\ in\ Document}{Total\ Number\ of\ Terms\ in\ Document}$$

$$Idf = \ln \frac{Total\ Number\ of\ Documents\ in\ Collection}{Number\ of\ Documents\ Containing\ Relevant\ Term}$$

[34]
For any given term, its term and document frequency are obtained in order to easily calculate the Tf-Idf of the term. Once calculated the term and value are inserted into a data structure from which retrieve the top-ten terms to formulate the query. This query explores whether all of the named entities are necessary for good results, or if some are more important than others. Tf-Idf is a good, initial way of identifying these potentially important terms. The *Tf-Idf Named Entities Query* for the document shown in Figure 4.2 can be seen in Figure 4.5.

```
craig_person fco_organization hmg_organization participate_organization martinek_person
lesley_person counterproliferation_organization iran_location rachel_person rick_person
```

Figure 4.5: An Example of a Tf-Idf Named Entities Query

Subject Query The subject of the document is not indexed and so is not retrievable from the index. This is due to its unpredictable length (Terrier can only save fixed length items in the index that are not the standard list of terms). The subject query is created at source document load time. When the API gets a request to return the source document, the subject is found within the document and then is analysed. The StanfordNLP documentation suggests that analysis should be run on text which is most like full sentences [19]. The named entities are identified and tagged and the query is then formatted and saved in an instance field to be passed as JSON to the client.

Due to needing a StanfordNLP instance to create the subject query the first source document analysed can often take slightly longer than the others if StanfordNLP has not been used before this point. This is due to the long start up time required to use the StanfordNLP features. This query was chosen in order to explore if the subject provided enough information to form a meaningful query, without even looking at the main body of the document. The *Subject Query* for the document shown in Figure 4.2 can be seen in Figure 4.6. It can be seen that not all named entities are identified, which may reduce the performance of the query.

```
uk will not participate in irans_location nuclear conference will discourage others from
attending ref state 112229
```

Figure 4.6: An Example Subject Query

```
Map.Entry<String, LexiconEntry> lee = lexicon.getLexiconEntry(postings.getId());
String strippedKey = stripPunctuation(lee.getKey());
// Add each term to the all terms query
allTermsQuery.append(strippedKey).append(" ");
// Add only NE terms to the NE Query
if (lee.getKey().contains("_")){
  NEQuery.append(strippedKey).append(" ");
  // Term Frequency
  double tf = ((double)postings.getFrequency())/((double)totalTermsInDocument);
  // Inverse Document Frequency
  LexiconEntry lexEntry = lexicon.getLexiconEntry(lee.getKey());
  double idf = Math.log(((double)INDEX_SIZE)/((double)lexEntry.getDocumentFrequency()));
  namedEntities.tfIdfNamedEntitiesList.add(namedEntities.new
  TfIdfNamedEntity(lee.getKey(), tf*idf));
```

Code Sample 4.3: Generating Queries by Analysing Index Terms

Storing the Queries Having formulated all four queries, they are wrapped in tags and appended to the end of the `.trec` source document from which they were generated. This allows the process described in Section 4.6.1 to find the queries without the need for any new storage mechanisms. The very fact that target documents are stored and indexed locally also helps to satisfy **Need.3** by not exposing unnecessary information to the internet.

4.4.3 Target Documents

The system operates on the assumption that there already exists a target index within its directory structure. As such, a separate Java program was written which can index target documents through command line invocation. The program loads the appropriate Terrier properties and invokes indexing through Terrier's API.

During target document indexing the Terrier properties are configured so as to create abstracts. Abstracts are parts of a tagged document to be saved in meta-data to provide helpful snippets of the document when it is found during retrieval. In this case abstracts are generated for the title, data, keywords and body of the document to give a sample of document contents when looking at query results. This is intended to make it easier for a reviewer to select a relevant target document to inspect.

Configuring Terrier correctly in order to create these abstracts caused considerable trouble during development. It was, in fact, the identifier of a flaw in the source code of Terrier. Owing to the fact the project supervisor was a key Terrier developer, this was fixed through a patch. Target document indexing took a considerably long time (upwards of 8 hours), so it was important to make sure the configuration was correct before invoking it, in order to not waste time.

As target document indexing uses the same tokeniser (see Section 4.4.1) as source document indexing, it can be guaranteed that terms will match across the two corpora when performing retrieval.

4.5 Retrieval

Having formulated queries and indexed the requisite documents a system must be implemented to allow target documents to be retrieved by performing a query on the target document index. This is not complicated and is almost identical to the example retrieval code provided in the Terrier documentation [22]. Retrieval is a fairly simple affair, almost identical to the provided example retrieval code provided in the Terrier documentation. A query is provided through the HTTP API which is run against the target index. This returns a set of results identifying target documents which match the query. These results contain the abstracts generated during target indexing. All of this information is saved as instance fields in class which is then serialized as JSON to be sent to the client as a HTTP response.

Decoration At this stage Decoration is also used, a functionality provided by Terrier to highlight and emphasise terms within a result set which appear in the query. Conveniently the decoration wraps found terms in <b> HTML tags, allowing us to render them as bold in the frontend.

4.6 Document Presentation

4.6.1 Document Parsing

In order to display the source and target documents to users on the frontend they must be parsed and converted to JSON. Both target and source documents are provided in TREC format with various meta-data tags and an untagged body section. An abstract class, `TaggedDocument` was created, containing a method which reads these documents identifying tag types and content. Once identified, this method can pass the tag type and tag content to a method within the the concrete subclass which matches tag types and puts the content in an instance variable ready to be serialized to JSON and sent to the client.

4.7 Evaluation Preparation

To to create a reasonable data set which could be used in the evaluation discussed in Chapter 6 it was necessary to create `.topics` and `.qrels` files. Sample queries are saved in `.topics` files and judgements on results are saved in `.qrels` files. To achieve this API methods were added to receive queries and judgements from the client. Another method was added which read the `.qrels` to send those documents to the client which had already been reviewed. This section was fairly basic file reading and writing although some care was taken to ensure duplicate topics were not added.

4.8 Properties

Terrier operates with a collection of properties which change the behaviour of its various operations. These properties can be set inside a terrier.properties file, however this complicates changing these properties between different operations. These properties can be set programmatically and so a static class was made which allows for easy switching of settings to facilitate correct invocation of indexing and retrieval.

4.9 Summary

Implementation of the server component of the system was challenging and comprised many new techniques and challenges. Produced was a functional REST API capable of handling the specific IR tasks necessary for this project. The implementation of the server component of the system prompted the following research questions:

Server.1 Was query generation effective?

Server.2 Which query is best?

Server.3 Did NLP models effect the query results?

Server.4 Is the system usably fast?

Chapter 5

Client Implementation

This chapter will deal with the steps taken in the implementation of the web application which acts as the client in the client-server paradigm. The client mainly consists of two side by side panels. The left side presents source documents and the right side presents query results and target documents using a tab system. The web application could be considered as a "single page application".

5.1 Tools and Methods

Dependency Management Bower was used for dependency management within the realm of frontend JavaScript. Bower allows a programmer to define dependencies through a `bower.json` file. These can then be downloaded through a bower install command into a bower components folder, allowing for easy installation on a new machine. Bower was used to install JQuery, Bootstrap and Knockout.

5.2 ViewModels

The logic for the user interface exists almost entirely within two JavaScript files: `IndexViewModel.js` and `TargetDocumentViewModel.js`. Within these files *observables* are defined. These observables are like regular variables except that they generally have a DOM element associated with them which somehow renders the variable contents. Take for example the code in Code Sample 5.1. This means that whenever the observable `self.sourceDoc.created` is updated the shown `span` element will have its contents updated. Similar syntax can also be used to invoke functions when clicking an element, or even to update CSS classes.

```
<span data-bind="text: sourceDoc.created"></span>

self.sourceDoc = {
  ...
  created : ko.observable("Loading..."),
  ...
}
```

Code Sample 5.1: Data Binding with Knockout Observables

5.3 Computed Observables

Aside from standard observables, the programmer can also define “computed" observables. This means that their value depends on an automatically evaluating function. These functions re-evaluate any time an observable which is a dependency to the function changes. This system allows for complex logic which can be triggered automatically depending on certain circumstances. This feature is used extensively to automatically fire HTTP requests when certain observables change.

5.4 Components for Tab System

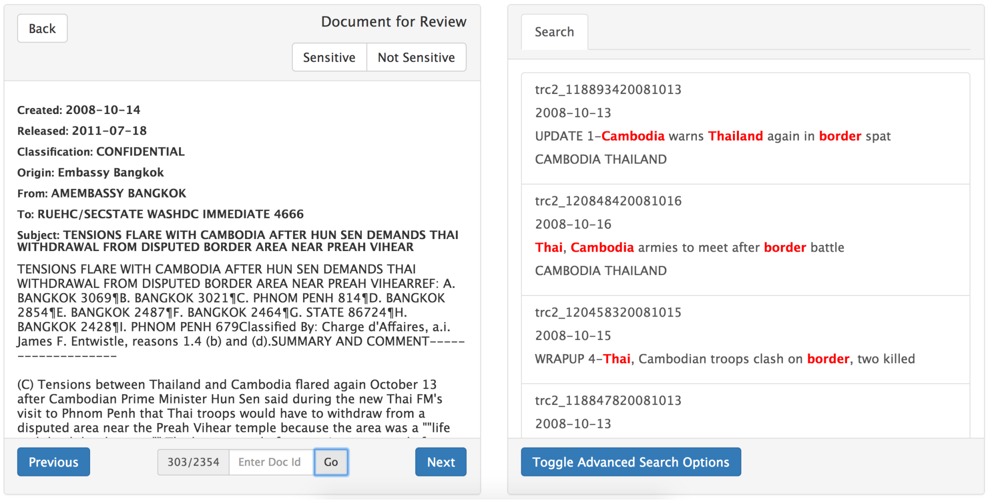
Knockout provides a component system which allows one to register new HTML elements, along with a specific view model with which these objects interact. The programmer provides a HTML template (complete with data-binding) and a ViewModel. When instantiating one of these components one passes in arguments to fill the ViewModel. This system lends itself excellently to the tab system present in the system. Using a “foreach" binding along with these custom components a fully functional tab system exists, which is, at its foundation, simply an array of docNos. A new instance of the TargetDocumentViewModel exists for every tab in the system, containing all of the information needed to render each document. The tab system can be seen in Figure 5.2.

5.5 Decoration

CSS is used to highlight in red those terms which are common to the query and the retrieved target document abstracts. Terrier’s Decoration functionality wraps these terms in <b> tags. Further highlighting is applied to these terms using the CSS show in Code Sample 5.2. This can also be seen in Figure 5.1.

```css
.decorate > b{
    color: red;
}
```

Code Sample 5.2: CSS for Query Results Decoration



Created: 2008-10-14
Released: 2011-07-18
Classification: CONFIDENTIAL
Origin: Embassy Bangkok
From: AMEMBASSY BANGKOK
To: RUEHC/SECSTATE WASHDC IMMEDIATE 4666
Subject: TENSIONS FLARE WITH CAMBODIA AFTER HUN SEN DEMANDS THAI WITHDRAWAL FROM DISPUTED BORDER AREA NEAR PREAH VIHEAR

TENSIONS FLARE WITH CAMBODIA AFTER HUN SEN DEMANDS THAI WITHDRAWAL FROM DISPUTED BORDER AREA NEAR PREAH VIHEARREF: A. BANGKOK 3069¶B. BANGKOK 3021¶C. PHNOM PENH 814¶D. BANGKOK 2854¶E. BANGKOK 2487¶F. BANGKOK 2464¶G. STATE 86724¶H. BANGKOK 2428¶I. PHNOM PENH 679Classified By: Charge d'Affaires, a.i. James F. Entwistle, reasons 1.4 (b) and (d).SUMMARY AND COMMENT-------------------

(C) Tensions between Thailand and Cambodia flared again October 13 after Cambodian Prime Minister Hun Sen said during the new Thai FM's visit to Phnom Penh that Thai troops would have to withdraw from a disputed area near the Preah Vihear temple because the area was a ""life

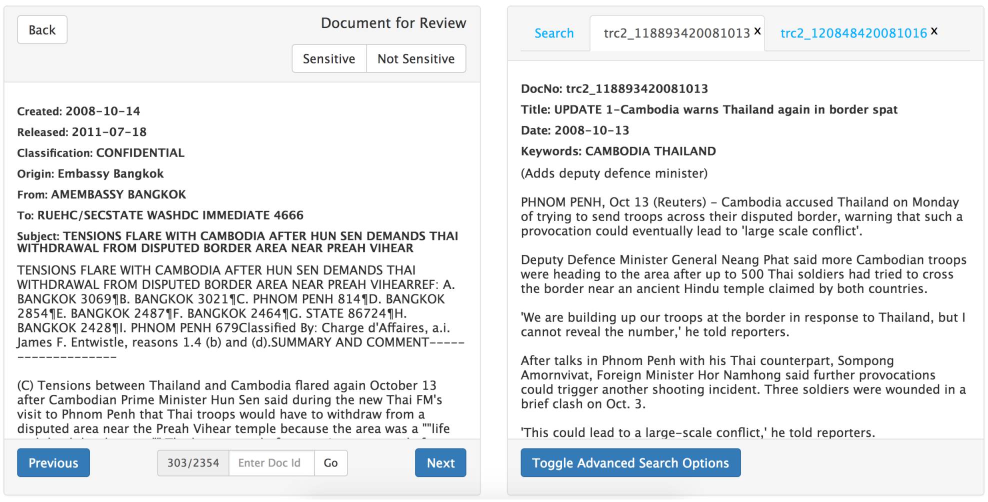
trc2_118893420081013
2008-10-13
UPDATE 1-**Cambodia** warns **Thailand** again in **border** spat
CAMBODIA THAILAND

trc2_120848420081016
2008-10-16
Thai, **Cambodia** armies to meet after **border** battle
CAMBODIA THAILAND

trc2_120458320081015
2008-10-15
WRAPUP 4-**Thai**, Cambodian troops clash on **border**, two killed

trc2_118847820081013
2008-10-13

Figure 5.1: The Front-End Search Results



Created: 2008-10-14

Released: 2011-07-18

Classification: CONFIDENTIAL

Origin: Embassy Bangkok

From: AMEMBASSY BANGKOK

To: RUEHC/SECSTATE WASHDC IMMEDIATE 4666

Subject: TENSIONS FLARE WITH CAMBODIA AFTER HUN SEN DEMANDS THAI WITHDRAWAL FROM DISPUTED BORDER AREA NEAR PREAH VIHEAR

TENSIONS FLARE WITH CAMBODIA AFTER HUN SEN DEMANDS THAI WITHDRAWAL FROM DISPUTED BORDER AREA NEAR PREAH VIHEARREF: A. BANGKOK 3069¶B. BANGKOK 3021¶C. PHNOM PENH 814¶D. BANGKOK 2854¶E. BANGKOK 2487¶F. BANGKOK 2464¶G. STATE 86724¶H. BANGKOK 2428¶I. PHNOM PENH 679Classified By: Charge d'Affaires, a.i. James F. Entwistle, reasons 1.4 (b) and (d).SUMMARY AND COMMENT-------------------

(C) Tensions between Thailand and Cambodia flared again October 13 after Cambodian Prime Minister Hun Sen said during the new Thai FM's visit to Phnom Penh that Thai troops would have to withdraw from a disputed area near the Preah Vihear temple because the area was a ""life

DocNo: trc2_118893420081013

Title: UPDATE 1-Cambodia warns Thailand again in border spat

Date: 2008-10-13

Keywords: CAMBODIA THAILAND

(Adds deputy defence minister)

PHNOM PENH, Oct 13 (Reuters) - Cambodia accused Thailand on Monday of trying to send troops across their disputed border, warning that such a provocation could eventually lead to 'large scale conflict'.

Deputy Defence Minister General Neang Phat said more Cambodian troops were heading to the area after up to 500 Thai soldiers had tried to cross the border near an ancient Hindu temple claimed by both countries.

'We are building up our troops at the border in response to Thailand, but I cannot reveal the number,' he told reporters.

After talks in Phnom Penh with his Thai counterpart, Sompong Amornvivat, Foreign Minister Hor Namhong said further provocations could trigger another shooting incident. Three soldiers were wounded in a brief clash on Oct. 3.

'This could lead to a large-scale conflict,' he told reporters.

Figure 5.2: Viewing a Target Document

5.6 Identifying Sensitivities

As seen in Figure 5.2 there are buttons to allow a reviewer to identify a source document as "Sensitive" or "Not Sensitive". Although not attached to any logic, these serve as a place holder in the prototype for what would eventually be real functionality attached to the document review process. Once out of the prototype stage this project could integrate with current systems that allow documents to be classified. The wording could also be easily changed if necessary. This feature helps to satisfy requirement **M.1** by providing a way for a reviewer to identify public domain knowledge in a source document.

5.7 Index Choices

There exists functionality in the frontend to allow a user to choose which index they would like to work on. In a fully functioning system this would be updated to interface with whatever file system or technology is in place.

5.8 Summary

The implementation of the client component of the system did not pose many major difficulties. The single page application provides a functional user interface to interact with the IR capabilities of the server. The implementation of the client component prompted the following research questions:

Client.1 Is the user interface easy to use?

Client.2 Are the documents presented in an intuitive way?

Chapter 6

Evaluation

This chapter discusses the various stages of evaluation and experimentation performed on the system in an attempt to answer the research questions proposed above and to ensure requirements are fulfilled. Section 6.1 looks at unit testing to ensure the system is robust, fulfilling **NF.5**. Section 6.2 refers to some qualitative evaluation and impressions about the system, understood through experimentation and general use. Section 6.3 discusses experiments performed to fulfil **M.3** and to choose the best query to perform automatically, satisfying **M.2** and answering research question **Server.2**. This section also generally covers how effective query generation was, answering **Server.1** Finally, in Section 6.4 evaluations are conducted in order to look into the thoughts of a potential user of the system and to understand how the UI can be improved for usability to better satisfy **NF.1** and answer **Client.1** & **Client.2**.

6.1 Testing

6.1.1 Unit Testing

To ensure **NF.5** (robustness) was fulfilled, the server component of the system was unit tested. JUnit was used to organise and run unit tests on the various Java classes that comprised this component [15]. The Arrange-Act-Assert pattern was used to organise unit tests in a structured way [32]. This allows for clear identification of the method being tested, separate from the code needed to prepare for the test.

Testing necessitated the refactoring of some of the classes which read and write information to files. Many of these classes originally had constructors which took file paths as arguments. The classes would then open the files and read or write to them as necessary. This approach is inflexible and leads to high coupling [42]. It also means that unit tests need either a sandbox file system to prepare tests, or some kind of mocking of the file system. Mocking a file system is overly complex and adding a sandbox filesystem adds unpredictable side effects.

```
// Arrange
StringReader sr = new StringReader("<DOC>\n"
  + "<DOCNO>docNo value</DOCNO>\n"
  + "<KEYWORDS>keywords value</KEYWORDS>\n"
  + "</DOC>");
// Act
TargetDocument doc = new TargetDocument(sr);
// Assert
```

```
  assertEquals(doc.docNo, "docNo value");
  assertEquals(doc.keywords, "keywords value");
```

Code Sample 6.1: A Unit Test Demonstrating *Arrange, Act, Assert* and a `StringReader` Mimicking a File

A better, more general, and self contained approach is to have the constructor take a `Reader` or `Writer` interface as an argument. Any implementing class of these interfaces can then be given as argument to be acted upon for the rest of the operations. This meant that `StringReaders` or `StringWriters` could be used in unit tests to substitute for real files as seen in Code Sample 6.1. This approach was extended to several classes which handle interacting with files.

Code which interacted with the Terrier API posed problems in term of unit testing. Terrier often introduces side effects in the form of interacting with established index files in the file system.

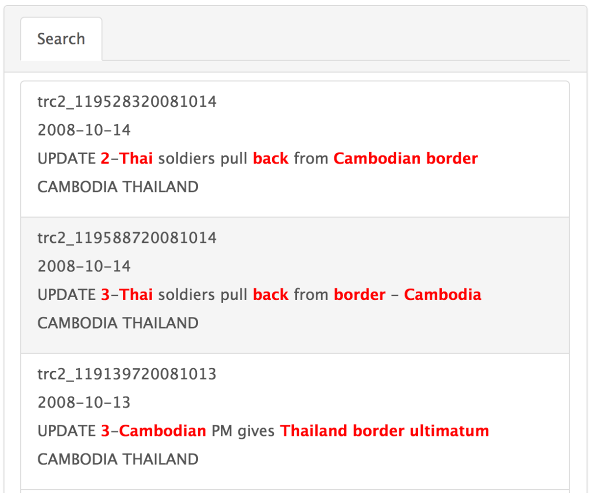
The client component of the system was not unit tested, as the logic necessary for its functions was not overly complex. Time constraints necessitated time be spent developing more features. Future work would aim to see a much more robust, tested client application to ensure good usability.

6.2 Qualitative Analysis and Defining Relevance

Through the course of developing the system and producing the ground truth necessary for the Offline Evaluation as discussed in the Section 6.3, certain nuances of the system became clear.

Named entity analysis is not consistent and often produces unexpected or wrong results. Sometimes entities were missed and often words such as "and" or "the" were tagged as named entities incorrectly. Combating this, the variations of "and" and "the" created in the named entity identification process were added to the stopwords file to avoid them appearing in queries and being matched in retrieval.

In order to produce the ground truth it was necessary to define returned target documents as "relevant" or "not relevant". This was extremely nuanced and relied on the discretion of the reviewer. Figure 6.1 shows the varied target document results when reviewing a source document regarding border clashes between Thailand and Vietnam.



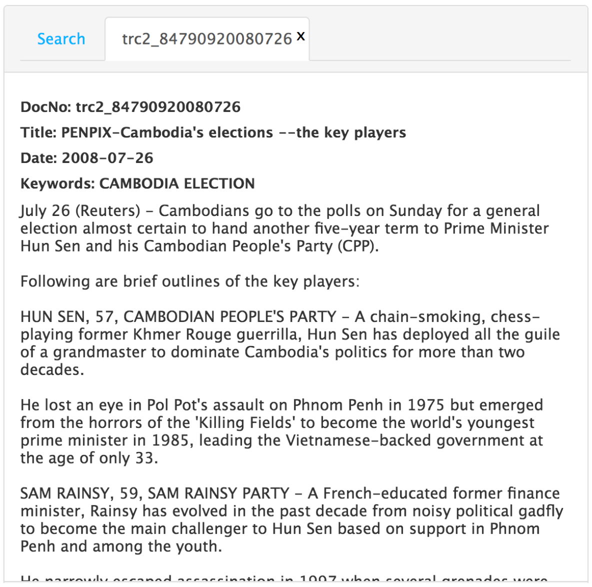
Search

trc2_119528320081014
2008-10-14
UPDATE 2-Thai soldiers pull back from Cambodian border
CAMBODIA THAILAND

trc2_119588720081014
2008-10-14
UPDATE 3-Thai soldiers pull back from border - Cambodia
CAMBODIA THAILAND

trc2_119139720081013
2008-10-13
UPDATE 3-Cambodian PM gives Thailand border ultimatum
CAMBODIA THAILAND

(a) Recognisable Relevant Results



Search | trc2_84790920080726 ×

DocNo: trc2_84790920080726
Title: PENPIX-Cambodia's elections --the key players
Date: 2008-07-26
Keywords: CAMBODIA ELECTION

July 26 (Reuters) - Cambodians go to the polls on Sunday for a general election almost certain to hand another five-year term to Prime Minister Hun Sen and his Cambodian People's Party (CPP).

Following are brief outlines of the key players:

HUN SEN, 57, CAMBODIAN PEOPLE'S PARTY - A chain-smoking, chess-playing former Khmer Rouge guerrilla, Hun Sen has deployed all the guile of a grandmaster to dominate Cambodia's politics for more than two decades.

He lost an eye in Pol Pot's assault on Phnom Penh in 1975 but emerged from the horrors of the 'Killing Fields' to become the world's youngest prime minister in 1985, leading the Vietnamese-backed government at the age of only 33.

SAM RAINSY, 59, SAM RAINSY PARTY - A French-educated former finance minister, Rainsy has evolved in the past decade from noisy political gadfly to become the main challenger to Hun Sen based on support in Phnom Penh and among the youth.

(b) An Irrelevant Result

Figure 6.1: Comparing Target Documents

It is clear why the result shown in Figure 6.1b was returned from some formulation of a query, as it mentions Cambodia and their President, however it requires the reviewer to see that it does not mention Thailand or border clashes. Sometimes, this was not as simple and things like dates were important. An example is an article regarding "Fighting in North Darfur". The event being referred to was around a specific date, and it had to be ensured that any target document pertaining to fighting in North Darfur matched the correct time frame.

The query formed from Tf-Idf ranking of named entities (see Section 4.4.2) was interesting. It often produced relevant results, but was also the cause of many strange results, such as reports on international sports fixtures. One can see how this occurred, since the names of several countries (often small African nations) not mentioned frequently throughout the collection can push these terms to the top of this query. In creating the ground truth, it was evident that the system can indeed be used to identify when public domain knowledge exists inside source documents, thereby in part satisfying **M.1**. It can also be noted that the system performed well when moving through the large collections of source documents and viewing target documents, indicating that it can deal with relatively large amounts of documents, fulfilling **NF.4** (scalability).

6.3 Offline Evaluation Using Terrier

6.3.1 Experimental Environment

Experiments were performed inside a Terrier distribution separate to the main server application. For this experimentation (and throughout development) a collection of source documents analogous to those created by government was used. Target documents consisted of a collection of news articles in the public domain.

For 20 source documents the top 20 results of the 4 query types discussed in Section 4.4.2 were marked as either relevant or not relevant. This formed the ground truth or gold standard against which offline evaluations could be run [34]. Topics (queries) to be tested exist in `.topics` file. These judgements were recorded in a `.qrels` file. The system also provided functionality to record the time spent reviewing a target document and a user comment about why the judgement was made. This system could be expanded upon in future work to further understand the document review process.

The offline evaluations were conducted using command line tools provided with Terrier. These can produce customisable and in depth measures of system performance. One of the most important and relevant measures of IR performance in a system is given by Mean Average Precision (MAP). MAP is given by:

$$MAP = \frac{\sum_{n=1}^{Q} Ave(P(q))}{Q}$$

Where:

- Q is the number of queries.
- $Ave(P(q))$ is is the average precision of a given query.

[34]

In order to ensure the choice of query conformed to the non-functional requirements, specifically **NF.3** (must return results in reasonable time), mean retrieval time and mean query length are also considered. *Precision at 1*, *Precision at 4* and *Mean Reciprocal Rank* are also investigated.

It can be assumed Stanford NLP Models with No Distributional Similarity features are used for all experiments other than those in the section which discuss this explicitly (Section 6.3.3).

6.3.2 Which Query Formulation is Best?

As posed in the research questions in Section 4.9 (specifically, **Server.2**) it must be investigated which formulation of the query produced from each source document is the most appropriate to be run automatically. The methods used to produce these queries are discussed at length in the Server Implementation chapter, see 4.4.2. Using the ground truth discussed above Terrier is used to run an offline evaluation with a `topics` file consisting of the 4 different query formulations for 20 source documents. These are: *All Terms Query, Named Entities Query, Tf-Idf Named Entities Query,* and *Subject Query*.

Query	MAP	Mean Query Length (Words)	Mean Query Processing Time (s)	Mean Reciprocal Rank	Mean Precision at 4
All Terms	**0.4554**	358.5	2.51435	**0.8500**	**0.6750**
Named Entities	0.3979	93.45	0.4132	0.7526	0.5875
Tf-Idf Named Entities	0.3038	10	**0.13675**	0.7236	0.4375
Subject	0.1840	**9.75**	0.1578	0.3367	0.2500

Table 6.1: Results of Offline Evaluation

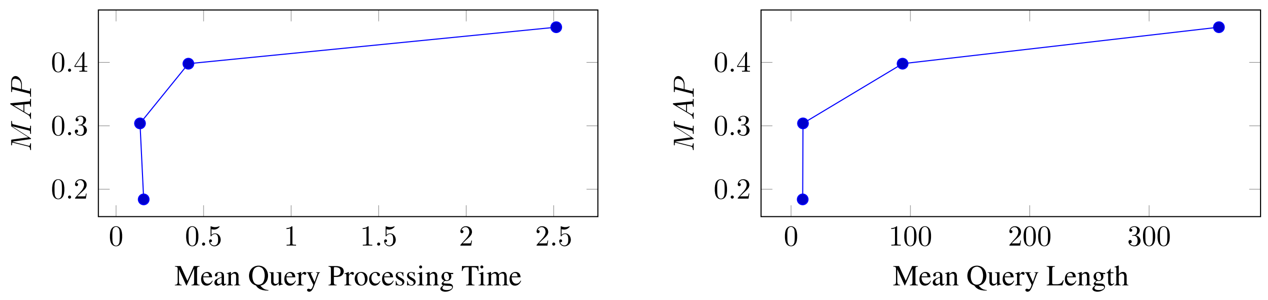


Figure 6.2: Analysis of Mean Average Precision (MAP) Scores against Query Processing Time and Query Length

As seen in Table 6.1 and Figure 6.2, the All Terms Query consistently performs the best, with little variation, in terms of MAP score. It is however, by far the longest query and so, takes the longest to run. The increase in precision is undoubtedly due to the context provided by including terms which are not themselves named entities.

The *Named Entities Query* performs slightly less well than the *All Terms Query*. It is, however, far faster than the *All Terms Query* as it consists of far fewer terms.

The *Tf-Idf Named Entities Query* performs worse than the *Named Entities Query*. It seems that by eliminating the terms which occur regularly in the document collection some vital terms are removed which increase the precision of a given query.

The *Subject Query* performs the most poorly. This can be expected, as other than running named entity recognition on the line, no other analysis was done to improve its performance. There is also no guarantee that the subject line will contain keywords which provide vital context to a query. Take, for example, this subject from a document in the source collection:

“PARLIAMENT’S FALL SESSION: A PREVIEW"

No insight is given into which country this refers to and although this is not true for many documents, the results clearly demonstrate that the lack of depth in subject queries can cause under-performance.

Although the *All Terms Query* produced the highest MAP scores, it took upwards of 5x as long to run compared to the *Named Entities Query*. At this point the importance of relative result relevance against time concerns must be considered. Users are only willing to wait so long for results (Google and Microsoft suffer business impacts for 0.5s delays! [5]).

Aside from MAP score, other measures can be considered. In general, the frontend displays the top 4 results of any given query after automatic querying. Terrier can return measures of the precision specifically for these top 4 results. As seen in Figure 6.1, the *Tf-Idf Named Entities Query* is close to matching the performance of the *All Terms Query* for the these top 4 results. The *All Terms Query* again performs the best, but this can be expected.

Another score which is important and relevant is Mean Reciprocal Rank, which is a measure of how close to the top the first relevant result was ranked. The same narrative presents itself here. The *All Terms Query* performs best, with the *Named Entities Query* scoring relatively high also. The difference in these scores is not great enough to neglect the vast difference in querying time.

As such, the most appropriate query to run automatically is that which consists of Named Entities. Choosing this query ensures Non-Functional Requirements **NF.2** (relevance) & **NF.3** (speed) are fulfilled, as well as ensuring the system is usably fast, answering research question **Server.4**.

6.3.3 Do StanfordNLP NE Models Affect Search Results?

As discussed in Section 4.4.1 the named entity identification facility in Stanford NLP can be altered by changing the model used to analyse the text. The same experiment as in Section 6.3.2 is now performed, however this time the Distributional Similarity (Dist-Sim) models are used, which produces remarkably different results, which can be seen in Table 6.2. This allows us to explore research question **Server.3**.

Query	MAP	Mean Query Length (Words)	Mean Query Processing Time (s)	Mean Reciprocal Rank	Mean Precision at 4
All Terms	**0.4397**	337.25	2.6132	**0.8750**	**0.6625**
Named Entities	0.2632	38.8	0.2323	0.6819	0.4625
Tf-Idf Named Entities	0.2249	10	**0.1506**	0.5857	0.3625
Subject	0.2349	**9.75**	0.1741	0.5992	0.3250

Table 6.2: MAP Scores and Query Metrics with Dist-Sim Models

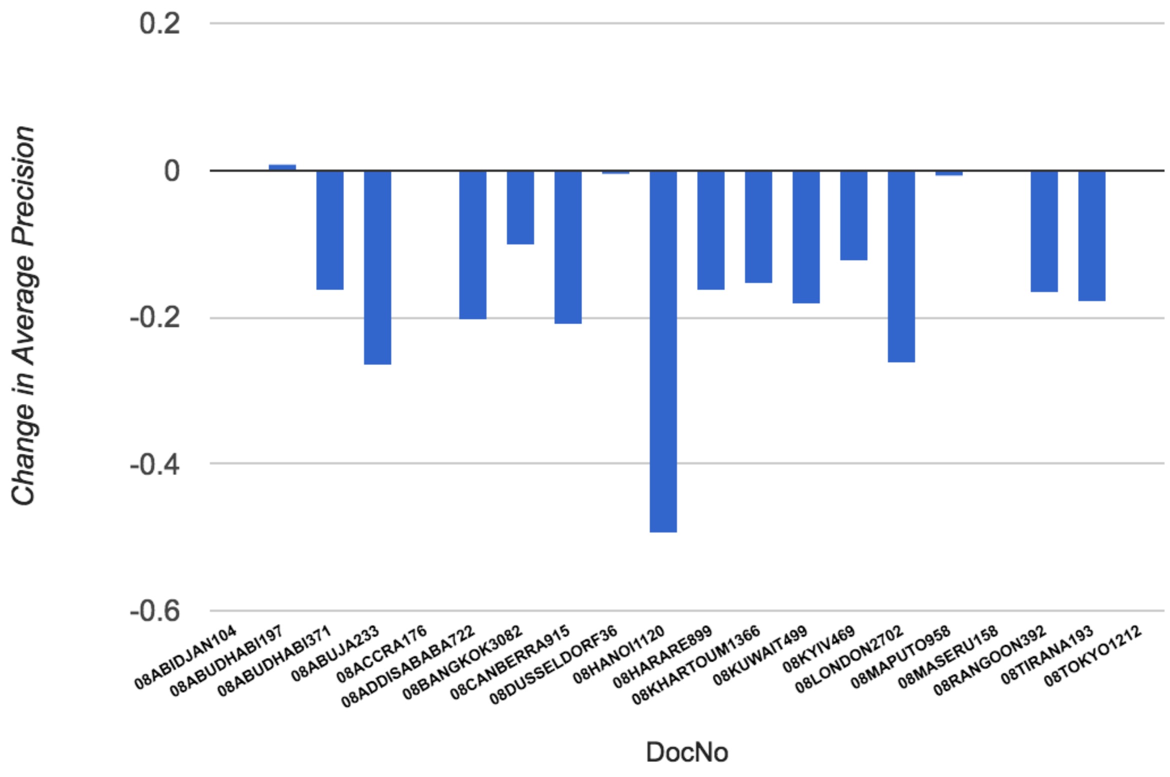


Figure 6.3: Change in Average Precision When Using DistSim NE Models

It can be seen that, when using the Dist-Sim models, the *All Terms Query* performance did not change by much, however the *Named Entities Query* suffered a large loss in precision. The *Tf-Idf Named Entities Query* was less precise and the *Subject Query* slightly more precise. Terrier provides the functionality to review evaluation scores on a query by query basis. In Figure 6.3 this is used to show the difference in precision when using the Dist-Sim models compared to the standard no Dist-Sim models. These results are from the *Named Entity Query* in both instances, as this has been identified as the "best" query in Section 6.3.2.

Document `08HANOI1120` exhibits a significant loss in average precision. The *Named Entities Queries* formed when using both types of models can be seen in Appendix B. The new query is much shorter, at 33 terms rather than 84 terms. The original (no Dist-Sim Models) query contains terms like 'what_person", "security_person" and "end_location". These are clearly not named entities and should not be tagged as such. The new (Dist-Sim Models) query seems, upon inspection, to be much more accurate. This example would suggest that the distributional similarity models do, indeed, produce "better" named entity tagging. Ironically, this actually worsens the performance of our queries by removing context words that result in better results. This is known to be the case, because these are the type of words that make the *All Terms Query* perform better (as discussed in Section 6.2). These findings indicate that the best precision in the system actually relies on a combination of named entities and other terms. Consequently, it was decided that the No Distributional Similarity Models should be used for now, as these produce the best results for the *Named Entities Query*.

6.3.4 Can the Tf-Idf Query Perform Better?

The *Named Entities Query* performs better than the *Tf-Idf Query* with only 10 terms. It can now be considered whether the number of terms in the *Tf-Idf Query* can improve the performance of this query formulation. The Tf-Idf Query was therefore generated with 5, 10 and 20 terms in order to compare their precision.

Query Length (Words)	Mean Average Precision (MAP)
5	0.2861
10	0.2635
20	**0.2890**

Table 6.3: Varying *Tf-Idf Query* Length

It can be seen from Figure 6.3 that this does not have particularly helpful or conclusive results. The original choice of 10 terms actually performs the worst, with 20 only narrowly beating 5. It seems then, that the top 5 terms are the most important contributor to the *Tf-Idf Query* in terms of precision.

6.3.5 Are Some Named Entity Types More Important Than Others?

The *Named Entities Query* performs well, but it is important to understand if splitting it down into its component parts can reveal which types of named entities have the greatest effect on results.

Named Entity Type	Mean Average Precision (MAP)
Person	**0.1236**
Location	0.1070
Organisation	0.0303

Table 6.4: Querying Using Only 1 Named Entity Type

From Table 6.4 it appears that the "person" named entities have the greatest impact on results. When considering future query formulations emphasis on "person" named entities should be increased.

6.3.6 What makes a query good or bad?

The queries formed from source documents `08ACCRA176` and `08MASERU158` score 0 precision in all four categories. This can be put down to the content of both of these documents. `08ACCRA176` describes a meeting of Ghanaian official following the raid of a brothel and `08MASERU158` describes a public transportation dispute. These stories are very local and so the number of relevant documents to find in the target collection will be vanishingly slim.

Precision at 1 and Reciprocal Rank Intuitively, the top result of a query is the most important one, and is the result the reviewer will be most likely to look at. It is important then to consider the "Precision at 1" of the queries. This whether or not the top result is relevant for a given query. Reciprocal rank can also be considered. This is a measure of how close to the highest ranking the first relevant document appears. When considering the *Named Entities Query*, by far the worst performing source document was: `08TOKYO1212` (other than `08ACCRA176` & `08MASERU158` which are discussed above). This document is addressing the issue of "cluster munitions" primarily. This is not a named entity and so will not be included in the *Named Entities Query*. If evaluation results of the corresponding *All Terms Query* are reviewed a near 100% increase in reciprocal rank can be seen, indicating that the inclusion of more context terms improve results. This reinforces the points made in Sections 6.3.2 & 6.3.3.

6.4 User Evaluation

User evaluations are performed to ensure the product is satisfactory to end users. Since this project was an experiment and a prototype, mainly focussing on IR techniques, it was less important to rigorously evaluate the frontend.

6.4.1 Think-Aloud

Timothy Gollins, Head of Digital Archiving at the National Records of Scotland, and key potential beneficiary of this project participated in a think-aloud study while examining the software in use. Also being former Head of Digital Preservation at The National Archives (UK), Timothy has expert insight into the problem domain. Some of the key points learned from this session are listed below:

- *"Named Entity importance is nuanced and frequency (or Tf-Idf) of named entities does not necessarily produce the most important terms."*

  This raises an important point regarding the limitations of automatic query generation. The system must allow the user to run custom queries and in the future, explore and learn from how real users formulate their queries. This suggests that the machine learning identified in Requirements **C.1** could be very useful.

- *"Further term highlighting or importance weighting is necessary to help produce the best results."*

  This point highlights the fact that the queries generated right now are fairly general, and some deep understanding of the way archivists review documents could help produce better results. Briefly mentioned was weighting the first and last paragraphs higher than the middle of the document, as these are likely to contain the most pertinent information. This again confirms the last point.

- *"More advanced methods could be leveraged to improve yet further the capabilities of the software, like eye tracking or automatic querying upon clicking terms."*

  Although beyond the scope of this project, this point identifies the potential expansion of this project. Eye-tracking could be used to automatically track what the user is interested in from the source document, forming the basis of new automatic queries. A simplified version of this could use clicking to accomplish the same results.

- *"UI considerations are equally as important as information retrieval considerations when it comes to improving the effectiveness of the software."*

  At this point in the project, the implementation of most of the IR functionality was complete. It now became important to consider how best to modify and improve the user interface to ensure the easiest experience possible. It was at this point, and in direct reaction to this comment, that decoration was implemented (see Sections 5.5 & 4.5).

Further to the above points, the participant was generally very enthusiastic about the applications of such a system and made it clear that this would be an extremely welcome introduction to the problem of sensitivity review. The user interface was clear and unambiguous and allowed discussing of the more specific points above. This user evaluation helps to answer the research questions about usability from Section 5.8. It also addresses Non-Functional Requirement **NF.1**, in that the system is somewhat intuitive, but improvements could be made. The system needs to do everything possible to highlight potential matches between source and target documents in order to help the reviewer identify public domain knowledge in the most efficient way.

6.5 Summary

Primarily the evaluations addressed the research questions proposed in Sections 4.9 & 5.8. Answering **Server.1**, query generation is somewhat effective, however could be improved through greater research and understanding. The *Named Entities Query* was found to be "best", as it produced relatively good results within reasonable time. This answers **Server.2**. In investigating **Server.4** it can be seen that the StanfordNLP models have a considerable effect on the performance of queries and that with added analysis, this could be optimised. The system is usably fast, answering **Server.4**, although only because the query choice is in line with this concern. As for the concerns of **Client.1** & **Client.2** the user interface is fairly easy and intuitive to use, as exemplified through a think aloud. It could be improved somewhat, however provides an excellent starting point for future work.

These evaluations drove changes to improve the project in various ways. The most appropriate query to automatically run was chosen and the best means (currently) of identifying named entities was identified. The IR accuracy was further increased by altering the stopwords file to remove incorrect named entities and terms. User evaluation drove the implementation of decoration to ensure the frontend was as intuitive as currently possible.

There are, however, many more changes that could be made to improve the performance of the system that were infeasible to implement within the time frame of the project. More decoration could be applied to target documents when displayed in full to help reviewers more easily spot similarities. Named entity recognition could be continually refined to ensure the best queries possible are formed. This could even come in the form of multiple indexes against which to run different queries, with each using different named entities techniques. The evaluations highlighted the successes and shortcomings of the project, answered the research questions proposed during implementation and ensured the requirements were fulfilled.

Chapter 7

Conclusions

7.1 Fulfilled Requirements

The Identifying Public Domain Knowledge project set out to create a system capable of improving the document review system necessary as digital records become more and more widespread. The final product satisfies all of the Must Have requirements and addresses some of the Should and Could Have sections. In terms of these requirements, the project can be classified as a success. More generally the project certainly addresses the needs set out in Section 2.3. The tool is fairly efficient and effective for reviewing documents addressing **Need.1**. **Need.2** is satisfied through automatic querying for target documents and the two panel display interface. **Need.3** is addressed by the tool not accessing information on the internet. All information is stored locally on the server, preventing sensitive searches from being exposed.

It demonstrates the applications IR and technology can have in the field of sensitivity review and paves the way for continued work to improve the ability of government to be transparent and fair, a matter which is very relevant in today's geo-political landscape. The project fails, however, to deal with all of the requirements and lacks some features which could greatly improve the usability and functionality.

7.2 Opportunities for Future Work

If work were to continue on this project many sections could be improved and new features could be introduced. Addressing the concerns raised in Chapter 6 it would be highly advisable to experiment more thoroughly with different query formulations to achieve the very best performance possible. This would require lots of very domain specific tweaking and testing. A key initial improvement may be to implement date analysis to ensure target documents are with in a reasonable time frame. This was identified as requirement **S.3** but was never implemented due to time constraints. Further to this, it would be achievable and worthwhile to perform better key phrase extraction upon source documents to form queries with named entities and context, other than the behemoth *All Terms Queries* which took too long to return results.

Further in the future, machine learning could be introduced into the system so it could begin to improve over time and with repeated use. This would be more useful if the product could be used in a real working environment, so it could adapt to the procedures of its end users. Even later in development, improved HCI features like eye tracking could be implemented to truly respond in real time to user input, but this would require all other aspects of the system to be much more robust.

7.3 Reflection

As a developer, I learned many things during the course of this project. Some of these things were technical, like a new found understanding of IR concepts and REST API design. However much of what was learned was regarding the effective planning and execution of a long term solo-project. Early planning and research is vital and I feel that could I repeat the project I would have made efforts to organise the early stages of development far better. This led to some negative impacts towards the end of the project, resulting in some features not being implemented (like date range searching). I am, however, very happy with the work I did complete and I have a keen interest in continuing research in the fields of IR and Sensitivity Review.

Appendices

Appendix A

Examples of Queries

```
s countries lrb  state_location who  rrb   will 24 participate made tell event 30 told
attending  give suggested demarche iran_location added officer ref foreign_organization
department_organization likely formal  nuclear group_organization reasons conference
comments attend uk_location political_organization office_organization
regional_organization discourage coordination_organization activities poloff_person
claims classified 14 c_organization london_location points separately
conference_organization mills_person leader consistent ngos_organization
team_organization october_person contacts nuclear_organization website
november_organization advisable intend substance will_person attendance iran_organization
credibility visit_location hmg_organization counselor_organization academics
will_location commonwealth_organization xxxxxxxxxxxx_person conference_location
nuclear_location 112229 rick_person craig_person amenable rachel_person nonattendance
fco_organization tuttle participate_organization martinek_person lesley_person
counterproliferation_organization
```

Figure A.1: An Example All Terms Query

```
state_location iran_location foreign_organization department_organization
group_organization uk_location political_organization office_organization
regional_organization coordination_organization poloff_person c_organization
london_location conference_organization mills_person ngos_organization team_organization
october_person nuclear_organization november_organization will_person iran_organization
visit_location hmg_organization counselor_organization will_location
commonwealth_organization xxxxxxxxxxxx_person conference_location nuclear_location
rick_person craig_person rachel_person fco_organization participate_organization
martinek_person lesley_person counterproliferation_organization
```

Figure A.2: An Example Named Entities Query

```
craig_person fco_organization hmg_organization participate_organization martinek_person
lesley_person counterproliferation_organization iran_location rachel_person rick_person
```

Figure A.3: An Example TF-IDF Named Entities Query

```
uk will not participate in irans_location nuclear conference will discourage others from
attending ref state 112229
```

Figure A.4: An Example Subject Query

Appendix B

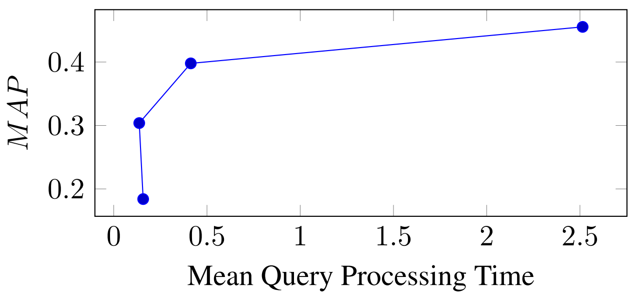
Named Entity Queries for `08HANOI1120`

```
council_organization summary_location sbu_organization comment_organization
ambassador_location us_location united_location control_organization embassy_organization
states_location affairs_organization end_location end_organization office_organization
government_organization committee_organization department_organization state_organization
land_location ministry_organization c_organization security_organization
foreign_organization on_organization social_organization but_organization
chairman_organization public_organization alike_person s_organization allow_location
media_location people_organization vietnam_location what_person outreach_organization
catholic_organization continued_location freedom_person freedom_organization
demonstrations_organization church_organization catholics_person thai_location
hanoi_organization gvn_organization hanoi_location vietnamese_location hanoi_person
dung_person religious_person thai_organization parish_location ha_location quang_person
religious_organization ngo_person church_location papal_organization
archbishop_organization kiet_person parishioners_organization joseph_person
archbishop_person papal_location nunciate_organization nuncio_location ap_organization
papal_person nuncio_person county_location speculating_person irf_organization
pope_person orange_location disputes_location papul_location vatican_location
vatican_organization religious_location karaoke_person vexed_organization
septebmer_organization ha_organization
```

Figure B.1: Named Entity Query When Using Models Without Distributional Similarity

```
council_organization sbu_organization us_organization us_location united_location
states_location affairs_organization office_organization ministry_organization
on_organization government_organization foreign_organization committee_organization
s_organization embassy_organization security_organization vietnam_location
people_organization catholic_organization public_organization church_organization
hanoi_location quang_person ngo_person kiet_person joseph_person hanoi_organization
religious_organization county_location orange_location archbishop_person vatican_location
vatican_organization
```

Figure B.2: Named Entity Query When Using Models With Distributional Similarity


MAP
0.2
0.3
0.4
0
0.5
1
1.5
2
2.5
Mean Query Processing Time

Bibliography

[1] Apache Lucene, homepage. https://lucene.apache.org/.

[2] Apache Maven, homepage. https://maven.apache.org/.

[3] Bootstrap homepage. http://getbootstrap.org/.

[4] Draw.IO, homepage, howpublished = https://pages.github.com/.

[5] Eric Schurman (Microsoft) and Jake Brutlag (Google), Performance Related Changes and their User Impact, 2009. https://www.youtube.com/watch?v=bQSE51-gr2s.

[6] Express, Fast, unopinionated, minimalist web framework for Node.js. https://expressjs.com/.

[7] Freedom of Information Act, 2000. http://www.legislation.gov.uk/ukpga/2000/36/contents.

[8] Git SCM. https://git-scm.com/.

[9] GitHub, homepage. https://github.com/.

[10] GitHub Pages, homepage. https://www.draw.io/.

[11] Google Search. https://www.google.com.

[12] JAX-RS API, Java API for RESTful Services. https://jax-rs-spec.java.net/.

[13] Jersey, RESTful Web Services in Java. https://jersey.java.net/.

[14] jQuery JavaScript Library, homepage. https://jquery.com/.

[15] JUnit Testing Framework, homepage. http://junit.org/.

[16] Knockout.js homepage. http://knockoutjs.com/.

[17] Main Portal Page for Jackon Project. https://github.com/FasterXML/jackson/.

[18] Node.js homepage. https://nodejs.org/en/.

[19] StanfordNLP Documentation, Caseless Models. http://stanfordnlp.github.io/CoreNLP/caseless.html.

[20] StanfordNLP Documentation, NER Options. http://stanfordnlp.github.io/CoreNLP/ner.html.

[21] StanfordNLP Documentation, Stanford Named Entity Recogniser, Models. https://nlp.stanford.edu/software/CRF-NER.shtml#Models.

[22] Terrier Documentation, Extending Retrieval. http://terrier.org/docs/v4.2/extend_retrieval.html.

[23] Terrier IR Platform. http://terrier.org/.

[24] Text REtrieval Conference (TREC). http://trec.nist.gov/.

[25] Trello, homepage. https://trello.com/.

[26] Wikipedia, the Free Encyclopedia. https://en.wikipedia.org/.

[27] Sir Alex Allan. Record review. https://www.gov.uk/government/publications/records-review-by-sir-alex-allan.

[28] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. 1999.

[29] Ted Barrett. FBI worked 'around the clock' to review emails in Clinton server probe. http://edition.cnn.com/2016/11/06/politics/fbi-review-hillary-clinton-emails-comey/.

[30] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005. https://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf.

[31] Timothy Gollins, Graham McDonald, Craig Macdonald, and Iadh Ounis. On using information retrieval for the selection and sensitivity review of digital public records. In *PIR@ SIGIR*, pages 39–40, 2014. http://www.dcs.gla.ac.uk/~graham/.

[32] J Grigg. Arrange act assert. http://c2.com/cgi/wiki, 2012.

[33] Craig Macdonald, Richard McCreadie, Rodrygo LT Santos, and Iadh Ounis. From puppy to maturity: Experiences in developing Terrier. *Proc. of OSIR at SIGIR*, pages 60–63, 2012. http://terrierteam.dcs.gla.ac.uk/publications/macdonald12terrier.pdf.

[34] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[35] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. http://www.aclweb.org/anthology/P/P14/P14-5010.

[36] Graham McDonald, Craig Macdonald, Iadh Ounis, and Timothy Gollins. Towards a classifier for digital sensitivity review. In *European Conference on Information Retrieval*, pages 500–506. Springer, 2014. http://www.dcs.gla.ac.uk/~graham/.

[37] Christian Middleton and Ricardo Baeza-Yates. A comparison of open source search engines. http://wrg.upf.edu/WRG/dctos/Middleton-Baeza.pdf.

[38] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA, 2007. ACM. http://dx.doi.org/10.1145/1321440.1321475.

[39] Michael Moss. Where have all the files gone? lost in action points every one? *Journal of Contemporary History*, 47(4):860–875, 2012.

[40] Douglas W Oard, William Webber, et al. Information retrieval for e-discovery. *Foundations and Trends® in Information Retrieval*, 7(2–3):99–237, 2013. http://w.codalism.com/research/papers/ow13fntir.pdf.

[41] Ian Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition, 2010.

[42] Tim Storer and Jeremy Singer. *Lecture Notes on Software Engineering*. School of Computing Science, University of Glasgow, 2014. Course notes for the Professional Software Development Course COMP-SCI4015.