STATISTICAL INFORMATION RETRIEVAL MODELS:
EXPERIMENTS, EVALUATION ON REAL TIME DATA

by

Ashwani Pratap Rao

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Summer 2014

© 2014 Ashwani Pratap Rao
All Rights Reserved
STATISTICAL INFORMATION RETRIEVAL MODELS:
EXPERIMENTS, EVALUATION ON REAL TIME DATA

by
Ashwani Pratap Rao

Approved:_____________________________________________________
Dr. Benjamin Carterette, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved:_____________________________________________________
Dr. Errol Lloyd, Ph.D.
Chair of the Department of Computer Science

Approved:_____________________________________________________
Dr. Babatunde Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved:_____________________________________________________
James G. Richards, Ph.D.
Vice Provost for Graduate and Professional Education
ACKNOWLEDGMENTS

I am thankful to my advisor Dr. Benjamin Carterette for his guidance and support. I would also like to thank the faculty of Department of Computer and Information Sciences for valuable guidance during my research. I had opportunity to learn from some very excellent teachers at University of Delaware. I will always remember the courses I took from Dr. Errol Lloyd and Dr. Sandra Carberry. I learned to ask relevant research questions from Dr. Kathleen McCoy and Dr. Vijay Shanker. My advisor and professors at CS department will always remain source of inspiration for me. The staff at CS department has always been very helpful. I would like to thank my thesis committee, Dr. Kathleen McCoy, Dr. Hui Fang, Dr. Benjamin Carterette for review and suggestions on this thesis. My colleagues, Mustafa Zengin, Praveen Chandler, Ashraf Rabiou always made working at lab fun and learning experience. Thanks to my family for everything. They have always helped and supported me.
# TABLE OF CONTENTS

| LIST OF TABLES | vii |
| LIST OF FIGURES | ix |
| ABSTRACT | x |

Chapter

## 1 INTRODUCTION

1.1 Relational Knowledge Base Systems and Retrieval 
1.2 XML Knowledge Base Systems and Retrieval 
1.3 Semantic Web and Retrieval

## 2 PREVIOUS WORK IN INFORMATION RETRIEVAL

2.1 Statistical Information Retrieval models

- 2.1.1 Okapi BM25
- 2.1.2 Language Modeling Approach
- 2.1.3 Poisson Query Generation Modeling
- 2.1.4 KL-Divergence

2.2 Information Filtering

- 2.2.1 Query Adaptation
- 2.2.2 Threshold Score

- 2.2.2.1 Model-Free Threshold Setting
- 2.2.2.2 Model-based Threshold Setting

- 2.2.3 Experimental Observations

2.3 Topic Detection and Tracking

- 2.3.1 Corpus
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.2</td>
<td>Event Tracking Task</td>
<td>23</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Approaches and Evaluation</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>2.3.3.1 System Comparison and Results</td>
<td>25</td>
</tr>
<tr>
<td>2.4</td>
<td>TREC KBA Track</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>2.4.1 Corpus and Topics</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>2.4.2 Work by Participating Groups</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>FILTERING SYSTEM</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>3.1 Problem Definition</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>3.2 System Process Flow</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>3.3 Retrieval Models</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>3.3.1 BM25</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>3.3.2 Poisson Query Generation Model</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>3.3.3 Language Model</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>3.3.4 KL-Divergence</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>3.4 System Design</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>EXPERIMENT AND RESULTS</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>4.1 Corpus</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>4.2 Topics</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>4.2.1 Topic Query Generation</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>4.3 Evaluation Measures</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>4.4 Retrieval Models</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>4.5 Results</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>4.5.1 Effect of Reducing Search Space</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>4.5.2 Effect of Retrieved Set Size</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>4.5.3 Effect of Query Type</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>4.5.3.1 Short Queries</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>4.5.3.2 Long Queries</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>4.5.3.3 Effect of Vital Document Counts</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>4.5.4 Effect of History Size</td>
<td>59</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>System comparison for topic tracking and detection project [1].</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>Some examples of topics with high and low numbers of vital documents in the corpus.</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>Table showing topics broken out according to the number of vital documents they have in the corpus.</td>
<td>45</td>
</tr>
<tr>
<td>4.3</td>
<td>Comparison of language model based filtering systems with different initial retrieved document set size. Indri cheating runs are also provided for comparison.</td>
<td>49</td>
</tr>
<tr>
<td>4.4</td>
<td>Precision and recall comparison at different values of $K$. The step 2 of system process flow (Section 3.2) is used to return top $K$ documents as vital for a topic.</td>
<td>50</td>
</tr>
<tr>
<td>4.5</td>
<td>Precision and recall comparison at different values of $K$ for two systems differing only in use of boolean queries.</td>
<td>51</td>
</tr>
<tr>
<td>4.6</td>
<td>System comparison for two different approaches in step 2 of system process flow. BM-OR uses “OR” query to retrieve documents in step 2 and BM-Indri-mu1 uses Indri retrieval engine to retrieve 1000 documents in step 2.</td>
<td>52</td>
</tr>
<tr>
<td>4.7</td>
<td>Results for various models on short query. Phrase “st” in run names for short label query. Indri cheating run results are shown for comparison.</td>
<td>54</td>
</tr>
<tr>
<td>4.8</td>
<td>Results for various models using long abstract query. 93 topics had abstract text. Results here are averaged over 93 topics.</td>
<td>55</td>
</tr>
<tr>
<td>4.9</td>
<td>Result comparison for 93 topics for short and long queries. “lg” for long abstract query and “st” for short query.</td>
<td>59</td>
</tr>
</tbody>
</table>
4.10 Comparison of run on 1, 5 and 10 days of history indexes and on short and large queries. “lg” is for abstract queries and ’st’ for short queries(topic labels) ............................. 61

4.11 Result comparison of BM25 and Poisson Model based runs against other research groups system TREC KBA 2013. .................. 61
## LIST OF FIGURES

1.1 A screenshot from Google Trends showing interest in the query “satya nadella” over time. ........................................... 2

3.1 Average Document size on each day of collection. Notice the change in scale of two figures ........................................... 34

3.2 Average Document size on consecutive 5 days of collection. Notice the change in scale of two figures and smoothing the plot due to cumulative effect. ........................................... 35

3.3 A Generic Information Filtering System ........................................... 37

4.1 Empirical cumulative distribution function (ecdf) plot of number of vital documents per topic in the corpus. ........................................... 45

4.2 Recall and precision comparison of two system differing only for step 2 of system process flow. ........................................... 53

4.3 Comparison of BM25 Runs for short label queries and long abstract queries. X axes is indexes for 93 Wikipedia Topics ........................................... 56

4.4 Comparison of Language Model Runs for short label queries and long abstract queries. X axes is indexes for 93 Wikipedia Topics. Red color ........................................... 57

4.5 Precision and recall comparison of models on three groups of topics. Topics are grouped into three categories according to the number of vital documents in collection—1 to 20, 21 to 100, and greater than 100. Long queries were used for retrieval. ........................................... 60
ABSTRACT

We are all aware of the rise of information age: heterogeneous sources of information and the ability to publish rapidly and indiscriminately are responsible for information chaos. In this work, we are interested in a system which can separate the “wheat” of vital information from the chaff within this information chaos. An efficient filtering system can accelerate meaningful utilization of knowledge. Consider Wikipedia, an example of community-driven knowledge synthesis. Facts about topics on Wikipedia are continuously being updated by users interested in a particular topic. Consider an automatic system (or an invisible robot) to which a topic such as “President of the United States” can be fed. This system will work ceaselessly, filtering new information created on the web in order to provide the small set of documents about the “President of the United States” that are vital to keeping the Wikipedia page relevant and up-to-date. In this work, we present an automatic information filtering system for this task. While building such a system, we have encountered issues related to scalability, retrieval algorithms, and system evaluation; we describe our efforts to understand and overcome these issues.
Chapter 1

INTRODUCTION

Vannevar Bush, in 1945, noted in his visionary and influential essay “As We May Think” [8] the problem of information explosion and utilization. He envisioned a machine that he called a “Memex” that would be able to store large amounts of heterogeneous knowledge. The essay conceptualized many modern day technologies such as personal computers, hypertext, the Internet and the World Wide Web. Bush’s “Memex” was also imagined to find relevant information through association. Indeed, modern day information retrieval systems index and find documents through association of terms. Bush saw the need for information filtering in particular for the right advancement of science; even as far back as World War II, scientists were faced with the problem of enormous growth in the scientific literature. Wikipedia is a modern example of a such source of knowledge accumulation. This huge online encyclopedia is maintained and updated by an army of volunteers scraping through the vast web for meaningful information. An automated system which can filter vital information will be of immense help to users interested in certain topics.

For example, consider Satya Nadella. Nadella rose rapidly as a topic of interest when it became clear that he was in contention for appointment as CEO of Microsoft in January of 2014—Figure 1.1\(^1\) is a screenshot from Google Trends showing a spike of interest on February 4, 2014 (label “D”), when he was appointed CEO, followed by a consistent volume of interest (http://www.google.com/trends/). Before 2014, Nadella hardly makes any appearance in Google’s data, save for a small uptick in early 2011 (labeled “G” on the figure), when he was President of Microsoft’s Servers and

\(^1\) http://www.google.com/trends/explore#q=satya%20nadella
Tools Division. Similarly, Nadella’s Wikipedia page, which was created in January of 2013, has already been edited 900 times in 2014 (as of early August) after only seeing 52 edits in all of 2013. Clearly Satya Nadella has rapidly become an entity that people have a great deal of interest in.

It is interesting to look in more depth at Nadella’s Wikipedia page. The original version published on Jan 23, 2013, has only one line:

Business leader and engineer, Satya Nadella is a native of Hyderabad, India.\(^2\)

By the next day it had been revised to:

Satya Nadella is a Microsoft executive who runs the company’s Server and Tools Business division.\(^3\)

with a citation to a page on http://microsoft.com. Several days later it was updated again to give brief detail on his educational background, with a second citation from ZDNet. By July of 2013, six months after creation, the page had been revised 15 times but was not substantially different from how it appeared at the end of January.

---


\(^3\) http://en.wikipedia.org/w/index.php?title=Satya_Nadella&oldid=534582513
In January of 2014, there was a surge of edits on the article. It quickly went from having just a few sentences organized into two sections to a more complete history of his employment and education, with 11 citations to external sources. One Wikipedia editor, responding to edits that deleted supposed rumors, notes:

The sudden surge of editing on this article, which has been little visited before, is far beyond “hearsay-related rumors”.

A few days later, Nadella was officially named CEO of Microsoft. His Wikipedia page now includes an “info box” with basic information about him, details of his career and education, and 38 citations to external sources.

The key questions this example raises are:

1. How did the Wikipedia editors recognize that Nadella was to become such an important figure? What did they see or read that convinced them of that fact?

2. When the amount of information available about Nadella exploded, how did Wikipedia editors process it to distinguish what was important and what wasn’t, what was true and what wasn’t, and what rates mention on his Wikipedia page and what doesn’t?

3. Among all the information now available about Nadella, did the Wikipedia editors miss anything vital to understanding Nadella and his role at Microsoft?

Our aim in this thesis is to build and evaluate an automatic information filtering system, which can help identify vital documents about a topic like Satya Nadella. In particular, such a system should be able to process a very large, heterogeneous, incoming stream of newly created or edited documents. The system should constantly be processing new documents to identify those that contain vital information on a topic—in particular, those that would make good citations for a Wikipedia page about the topic. An example of topic would be “Satya Nadella” and its corresponding Wikipedia page is http://en.wikipedia.org/wiki/Satya_Nadella.

Apart from the focus on Wikipedia, this is hardly a new problem (as the appearance of Bush’s essay in 1945 makes clear). The need for information storage and

consumption has motivated the development of Knowledge Base Systems (KBS). Over the years various kinds of KBS have been developed; Knowledge Base Systems is a generic term referring to the set of technologies to store and query complex structured and unstructured data. KBS can be divided very loosely into two subsystems—a system for knowledge representation and storage of information, and a system providing inference and query over the stored knowledge. However, KBS development has focused almost entirely on structured representations of data, to be accessed via structured query languages. To really implement Bush’s vision, a system will need to cope with unstructured and potentially vague information needs—few people would have been able to specifically say they were interested in Satya Nadella before February of 2014, but once it became clear he was important, interest followed.

In this thesis, we are concerned with processing a large, heterogeneous stream of information to find documents that are of vital importance to understanding a topic, concept, or entity, and doing so when the available representation of an information need is unstructured. In the remainder of this chapter, we give a brief overview of attempts to bring together structured KBS and semi-structured information needs, mainly from the database and data mining communities. In Chapter 2, we discuss related work in Information Retrieval, which is highly focused on unstructured information needs. In Chapter 3, we present our filtering system, and in Chapter 4 we demonstrate its effectiveness. We conclude in Chapter 5.

1.1 Relational Knowledge Base Systems and Retrieval

Relational Knowledge Base Systems use relational databases [10] as the underlying technology for the representation and storage of knowledge. For a long time, relational databases were used to store data in a flat tabular format. This is highly structured data storage with support for numeric and string data types. Users can query KBS through various types of query language engines. SQL [9] is a popular query language which provides the ability to fetch information based on matching data
in columns of tables. Complex queries can be constructed spanning across multiple tables. Some challenges of using Relational KBS for a filtering system are as follows:

1. The database schema with appropriate columns and data types needs to be defined. For example, the table should have topic name, location, birth date, origin etc as columns. Enumerating all possible topic attributes is challenging, particularly when the attributes one is interested in may change over time.

2. When incoming data is unstructured, a suitable parser will be required to parse the data for the attributes defined in table schemata. For example, a date of birth mentioned as “29th Aug 1967” in a document needs to be extracted and converted to an appropriate data type.

3. Creation of well-defined queries to filter information is required but challenging. Several query languages such as Relational Algebra [10], Relational Calculus [11] and Mapping Oriented Languages [9] have been proposed. But for a filtering system, a query in a particular syntax needs to be predefined for a user information need, and any change in information need will require a corresponding change in the query.

On the bright side, a user information need can be satisfied with high precision and recall by using Relational KBS as long as the user is only interested in data that the schema designers thought of and that the user is able to accurately specify in a query.


Any given person can have many occupations, more than one child, more than one publication, etc. The representation of multiple values of an attribute in relational database can be achieved by storing the information as a list of values for that column. Thus the publication list of Stephen Hawking can be stored as “[A Brief History of Time, The Universe in Nutshell, God Created the Integers]”. However, no proper relational database provides list as a data type, as it violates the first and most basic normal form. The usual approach is to represent the multiple attribute value across
multiple rows. For Stephen Hawking, there will be three rows representing different books. This will create a data redundancy problem as the values in other columns (example, “Children”) of Stephen Hawking will be duplicated in each of the rows. The theory of normalization of relational database can help to solve the problem of redundancy.

To the best of our knowledge, there is no work on information filtering systems using relational databases. This is mainly due to the complex design process for creating a KB, requiring a complex parsing system for unstructured text and development of complex filtering engine. A filtering system should be generic so as to process heterogeneous sources of information. Relational databases fail to address such information sources.

1.2 XML Knowledge Base Systems and Retrieval

XML-based knowledge base and retrieval systems combine the structured storage of relational databases with unstructured text content [2, 59]. This section is motivated from the report [2]. The extensible markup language (XML) was developed by W3C consortium as a meta language for structuring documents using tags. XML has become a standard for representation and exchange of documents and data. The logical structure of XML documents is formed by elements. Elements are meaningful pieces of data enclosed by tags, similar to the schema of a relational database but far more flexible. Nesting of elements is allowed, leading to a hierarchical representation of a document, also called XML document object model (DOM). An initiative for the evaluation of XML Retrieval (INEX) [2] is the leading platform for evaluation of XML based information retrieval.

Expressive query languages have been proposed for XML search [7], ranging from simple keyword search to complex queries on the structure of the documents. There are numerous challenges to information retrieval and filtering on XML KBS. We describe some as follows:
1. The representation of tags is not homogeneous. Each XML document can have its own data definitions (DTD) or tags. An effective information filtering system will require complete knowledge of DTD.

2. Statistical Information retrieval algorithms assign a score to a document with respect to a user query. A scoring system unifying the structure and the free text content has been difficult to develop.

3. The nesting of elements and hierarchical structure pose challenge to the effective granularity level of the information to be presented to the user.

Combining unstructured text and structure in the document seems attractive, but it adds complexity at various levels (user interaction to information storage) for which universal solutions have failed to develop. For a information retrieval system, the following components are important: user interaction, query specification, information inference mechanism, and information storage. We discuss the approaches and challenges for each of the components. We do not delve into storage of XML documents as data storage is not part of this thesis.

**User Interaction:** The hierarchical structure of a XML document poses challenges regarding the presentation of granularity level of information. A document can have nested structure with each element having many children. What is the right level of element to be presented to the user? Many systems propose presenting only leaf elements (element with no children in a XML document), but giving the user only leaf element information might miss relevant information present in other parent/sibling elements. Providing the least common ancestor of all the relevant elements has been proposed as an alternative, but the least common ancestor can be at a very high level in the DOM tree which can hurt the user interaction. Systems have been proposed to provide browsing capability, where users can explore the a hierarchy of document. But again exploring capability needs to take into account the right element level to be presented.

**Query Specification:** For traditional information retrieval systems on unstructured data, the query allows free-form text. For the structured data such as relational databases, the structure of the data (columns, table names) needs to be
part of the query. The semi-structured and non-homogeneous nature of XML data makes query specification more difficult. Several approaches have been proposed with different levels of expressiveness:

1. Keyword only queries: This is similar to traditional information retrieval systems where the query is free text. XML fragments matching the queries are returned as results. Examples of such query systems are XRank [21], XKSEArch [60], and NEXI content only query [55]. Some systems also return the least common ancestor of the XML fragment of the query result [29, 18].

2. Tag and Keyword: This query specification allows associating tags with query keywords. Example query “title:star wars” will return all the XML documents with title tag relevant to “star wars”. XSEarch [12] is a system with this capability.

3. Path and Keyword Queries: Due to the rooted hierarchical structure of XML document, each element can be reached following the path from the root element. World Wide Web Consortium standard XPATH query languages can be used for sophisticated conditions on structures. Other works on such query languages are XIRQL [18], XXL [54], and NEXI Content-And-Structure queries [55].

**Scoring Mechanism:** XML IR systems exploit the structure of a document by retrieving document components (XML elements) relevant to a user query instead of the whole document. XML retrieval systems score elements instead of the whole document and return elements at the right level of granularity. Consequently, the traditional information retrieval scoring methods are not directly applicable. For example, term frequency ($tf$) and document frequency ($df$) are two well known statistics used in traditional IR systems. For XML retrieval systems, we need to use element-level term frequency ($etf$) and element frequency ($ef$). The element term frequency ($ef$) is the number of times a term appears in an element of XML document. The element frequency ($ef$) is analogous to document frequency and is the count of elements in the collection containing the term. It is not straightforward to calculate $etf$ and $ef$ because of the nesting of elements in a XML document. One approach is to compute $etf$ only for leaf elements [20, 52]. The score of a non-leaf element is a weighted combination of the score of leaf elements and is also dependent upon the distance of an element from the leaf element. Concatenating the text of elements and its descendants for
calculation of $ef$ and $etf$ is another possible approach. Other approaches are ignoring element nesting or calculating $ef$ at the document level.

An Information filtering system based on XML KBS can be built on top of well established IR system, but effective IR systems on such KBS have failed to develop. Another problem with XML KBS is a lack of data being published on the web in XML format. A complex format such as XML hinders publication of data on the web by non-technical end users.

1.3 Semantic Web and Retrieval

The semantic web is an effort to make unstructured information on the web intelligent for machines and human. We have seen in the previous section that XML allows a user to add arbitrary structure to their document without saying anything about the structure semantics. Semantic web related technologies provide standards and languages to integrate semantics of the document with its unstructured text content. The semantic content can be processed by a machine for intelligent inference. Semantic modeling is a well-known and widely-researched problem in KBS. For a nice survey on semantic modeling research and applications, please refer to [24].

We focus here on semantic web related technologies and retrieval. Two important semantic technologies for developing Semantic Web are the eXtensible markup language (XML) discussed in the previous section, and the resource description framework (RDF). RDF encodes semantic information through sets of triplets. Each triplet consists of a subject, a verb, and an object in an elementary sentence. The triplets can be written using XML tags. The subject, verb, and object of a triplet are identified by corresponding URI, which is a unique web identifier for a resource. RDF can be represented using XML tags or through other notation such as notation3 (N3). Notation3 is easy to understand with some practice; an example of RDF triplet using N3 notation is as follows:

<http://dbpedia.org/resource/Academy_Award_for_Best_Art_Direction>

<http://xmlns.com/foaf/0.1/homepage>
The above RDF is one triplet split over three lines for demonstration purpose. This triplet is in subject, predicate, object format. The statement makes an assertion about the subject “Academy_Award_for_Best_Art_Direction” regarding its homepage being “www.oscars.org”.

Subjects, objects, and predicates can be considered as concepts. A concept can have sub-concepts, for example, the concept “animal” can have different subtypes such as mammal, bird, reptile, etc. Some concepts can have certain properties; for example, a university can have students, professors, events, courses, etc. Students can then have properties such as grades, activities, etc., while events have a different set of properties. The concepts, properties of concepts, and relationships between the concepts are provided through the semantic web ontology.

The W3C (World Wide Web Consortium) standard OWL (Web Ontology Language) provides the ability to express concepts and relationships between concepts ([http://www.w3.org/standards/semanticweb/ontology](http://www.w3.org/standards/semanticweb/ontology)). We demonstrate a simple example of use of ontology. Consider a book seller publishing the information on books in RDF format on the web and refers to authors of the books as “Author”. Consider a library knowledge base system which use the term “Writer” instead of “Author” to publish information. “Author” is similar to “Writer”, and this disambiguation can be specified by semantic web ontology. An excerpt from an OWL file from [http://www.w3.org/TR/owl-xmlsyntax/apd-example.html](http://www.w3.org/TR/owl-xmlsyntax/apd-example.html) is as follows:

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:label xml:lang="en">wine</rdfs:label>
  <rdfs:label xml:lang="fr">vin</rdfs:label>
</owl:Class>
```

The above ontology defines a concept “Wine” as a sub-class of PotableLiquid. PotableLiquid is defined in the “food” ontology. It also show that “wine” is referred
to as “vin” in French and “wine” in English. A machine program can exploit such published information to infer “wind” and “vin” as the same concept.

Over the past decade, many semantic web related standards and technologies have been developed. For interesting reading on semantic web origin and recent challenges, please refer to [6, 53]. The standards and technologies related to the semantic web are promising. There has been increasing adoption of these technologies in science communities, especially life sciences. Publishing data in semantic web format is being exploited by intelligent programs and AI experts. Publishing of semantic data does not mean that the web text content should be in semantic data format. It is to complement the unstructured web content. The web page will contain the usual text and media data, but will also contain pointers to its associated semantic data and ontology definition. This meta data is meant to be consumed by machines for intelligent inference.

Some of the challenges facing the semantic web [5] are described here. Web users and content providers outside the scientific community are not, as a rule, publishing data suitable for semantic web standards. Due to a lack of data, algorithms developed over semantic web standards are restricted to specialized domains. Ontologies have been developed for many specific domains, but development and acceptability by non-technical web users is slow. Ontology matching, mapping, and merging [15, 16, 43] are some of the complex challenges that hinder its usage and acceptability. Further, annotation of web pages with semantic information has been slow. Additionally, multilingual approaches to semantic web have not yet developed; for it to be truly widespread, this aspect cannot be ignored. Scalability is another issue with semantic web. Modern day search engines have developed efficient and robust methods to store and index the unstructured text data. Storage and indexing of semantic data for efficient retrieval and inference by machines is still an open and challenging problem.
Chapter 2

PREVIOUS WORK IN INFORMATION RETRIEVAL

An alternative approach to KBS can be seen in the field of Information Retrieval (IR), which is concerned with processing, storing, and searching unstructured, heterogeneous information. In this chapter we present related work in IR. In Section 2.1 we describe classic and state-of-the-art work on modeling relevance for information retrieval and document ranking. In Section 2.2 we discuss the filtering problem and the application of retrieval models to it. Sections 2.3 and 2.4 describe two domain-specific filtering problems that are related to our work; in particular the work described in Section 2.4 is directly related to the problems we wish to solve. The problem of using past accumulated information to make decision on incoming stream for relevancy to user information need is similar to problems described in Section 2.2, 2.3, and 2.4.

2.1 Statistical Information Retrieval models

The goal of information retrieval can be broadly said to satisfy user information needs from some knowledge resource. In the classic IR problem, the knowledge resource is considered static; it does not change between the time the query is entered and the time results are returned. The aim is to find relevant information in this static knowledge resource, which could contain anything from full-text documents to short “tweet” text. Here we describe a few of the most popular statistical IR models for text retrieval; these will be the basis for our work.

Statistical IR models assign a score to each item in the resource with respect to a user query. Items (usually text documents) are ranked in decreasing order of score. The document score is an estimate of its relevance to the query—higher ranking documents are therefore thought to be more relevant to a user query. While there are
non-statistical models for IR (most notably the Boolean model and the vector space model), in this work, we have used statistical models because they have been empirically proven to produce better results [61]. In addition, scalability and performance is critical when building systems for very large corpora, and statistical models fit these criterion.

For this work, we implemented Okapi BM25 [48, 47, 17], Language Modeling approach [44], KL-Divergence retrieval model [28, 63], and Poisson Model based query generation [34]. These models have been extensively studied and researched for ad-hoc retrieval oriented projects. We use these models to make decision on incoming document regarding its relevancy to a user query. These models use past accumulated information to make this decision.

2.1.1 Okapi BM25

BM25 [51] is a popular information retrieval model based on the Probability Ranking Principle [46]. We state here the Probability Ranking Principle from [46].

If a reference retrieval system’s response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.

The model development is based on the “eliteness” of terms of the query in the document. For a document-term pair, it is assumed that there is a hidden property of eliteness. If a term in the query is elite in a document, then that document can be said to be “about” the concept denoted by term. We do not go deep into the theory behind BM25; interested readers can refer to the nice theoretical background provided by Robertson [51], who demonstrates how to derive a scoring function from this theoretical basis.
The score of a document with respect to query is the sum of the weight of the query terms in the document. The weight of query term \( t \) with term frequency \( tf \) in document \( D \) is computed as follows:

\[
\text{BM}_{25}^t = \frac{tf_t}{k((1-b) + b\frac{|D|}{avdl}) + tf_t} w_{RSJ}^t \tag{2.1}
\]

where:

- \(|D|\) is the document length in terms.
- \(avdl\) is average document length observed in the corpora.
- \(k\) and \(b\) are free parameters to be set based upon previous study and experiments.
- \(w_{RSJ}^t\) is the Robertson/Sparck-Jones weight for term \( t \), discussed in more detail below.

We used 1.75 for \(k\) and 0.5 for \(b\), as these are standard values that have proven to work well across a range of corpora and queries.

The Robertson/Sparck-Jones weight \(w_{RSJ}^t\) is a multiplier based on inverse document frequency, calculated as:

\[
w_{RSJ}^t = \log \frac{N - n_t + 0.5}{n_t + 0.5} \tag{2.2}
\]

Here \(N\) is size of the corpus (number of documents) and \(n_t\) is number of documents containing term \(t\). This weight is similar to classical inverted document frequency measure calculated for terms. Term with a high inverted document frequency have more discriminative value.

### 2.1.2 Language Modeling Approach

The Language Modeling approach [44, 23, 35], also known as query-likelihood scoring, is based on the idea of creating a model of the language used in a document. The estimated documents model is used to calculated the query-likelihood score. The query-likelihood score is calculated from this document model by assuming the query to be a sample of terms drawn from the document’s language model.
More formally, let $Q$ be query, $D$ be a document, $\theta_D$ be an estimated document model. Then the score of document $D$ with respect to query $Q$ is the probability of $Q$ given $\theta_D$.

$$\text{score}(Q, D) = p(Q|\theta_D) \tag{2.3}$$

Let $V = \{t_1, t_2, ..., t_{|V|}\}$ be the vocabulary of language of documents of the corpus, i.e. the set of all unique terms (after stemming and stopword removal) used in all documents in the corpus. The document language model $\theta_D$ can be defined as a unigram language model or more generally a multinomial word distribution [23, 35], i.e., $\theta_D = \{p(t_i|D)\}_{i=1}^{[1,|V|]}$, where $p(t_i|D)$ is the probability of term $t_i$ in the document $D$. The probability of term $t_i$ with term frequency $tf_i$ in document $D$ of length $|D|$ can be estimated as follows:

$$p(t_i|D) = \frac{tf_i}{|D|} \tag{2.4}$$

According to this model, the likelihood of query $Q = t_1, ..., t_n$ is given as follows:

$$p(\{t_i\}|\theta_D) = \prod_{i=1}^{m} p(t_i|D) \tag{2.5}$$

From the above equation, higher frequency of query words in the document will translate into a higher query-likelihood score.

There is a problem with this model, however: the absence of a query term from document will set $p(t_i|D)$ to 0, which as per equation 2.5 will reduce the score of document to 0. Note that BM25 does not have this problem; nor do other models for IR. In general, it is important to be able to retrieve documents that do not contain all the query terms. Various smoothing methods have been proposed to avoid this problem of zero probability [62]. Most of the smoothing method take into consideration $p(t|\mathcal{C})$, the probability of term in the collection to avoid zero probabilities. The size of the collection is the sum of the size of all the documents in the collection. The probability of term $t_i$ with total frequency count $cf_i$ in collection $\mathcal{C}$ of size $|\mathcal{C}|$ is given as follows:

$$p(t_i|\mathcal{C}) = \frac{cf_i}{|\mathcal{C}|} \tag{2.6}$$
We have used the Dirichlet prior smoothing [62, 31] for the language modeling approach. Other smoothing methods such as Jelinek-Mercer [25] and absolute discounting [36] are also possible, but Dirichlet has been shown to work well. The probability of term $t_i$ with frequency $tf_i$ in document $D$ using Dirichlet smoothing with Dirichlet conjugate prior ($\mu$) is as follows:

$$p(t_i|D) = \frac{tf_i + \mu p(t_i|C)}{|D| + \mu}$$  \hspace{1cm} (2.7)

The collection probability of term $p(t_i|C)$ is given by equation 2.17. In practice, $\mu$ is a free parameter and setting value equivalent to the average document length in the collection is preferable. Using equation 2.7 in equation 2.5 and taking the logarithm will provide the query-likelihood score of the document.

### 2.1.3 Poisson Query Generation Modeling

The Poisson query generation modeling is based on the same principle of query generation as the language modeling described previously in Section 2.1.2. Recall that the query generation principle assumes that query terms are drawn from the document model. For this approach, the document is modeled as a Poisson Distribution rather than based on a multinomial distribution [23, 35] or bernoulli distribution [44].

The main difference between the two is that in a multinomial distribution, the probability of all terms in a document must sum to 1. In the Poisson model, however, the frequency of each term is modeled independently. A document is scored by estimating the query-likelihood from the estimated Poisson distributions of terms in a document [34]. It is supposed to combine the advantages of multinomial distribution modeling and Bernoulli distribution modeling [34]. It also makes per-term smoothing possible, which is hard to do in the multinomial distribution.

We now briefly describe the estimation of Poisson distribution for the document. A detailed treatment is nicely described in the work of Mei et al. [34]. Let $V = \{t_1, ..., t_n\}$ be the vocabulary and let $\langle tf_1^D, ..., tf_n^D \rangle$ be the observed frequency vector of words in a document D. The frequency counts of the n unique terms are modeled
as having been generated from $n$ independent homogeneous Poisson processes. Let us suppose an author composes a document in time $\tau$. Further assume that each unique word $t_i$ is generated by an independent Poisson process with rate $\lambda_i$ in a unit time. The probability density function of such Poisson distribution is as follows:

$$P(t_f^D|\lambda_i\tau) = \frac{e^{\lambda_i\tau}(\lambda_i\tau)^{t_f^D}}{t_f^D!}$$  \hspace{1cm} (2.8)

The time $\tau$ is the duration in which author composes the text which can be set to length of document $|D|$. Let $\Lambda_D = \{\lambda_1, D, ..., \lambda_n, D\}$ be the rate of generation of terms $\{t_1, ..., t_n\}$ by $n$ independent Poisson processes in document $D$. The rate of generation of a terms $t$ with term frequency $t_f$ in document $D$ is computed as follows:

$$\lambda_t = \frac{t_f}{|D|}$$  \hspace{1cm} (2.9)

Now the likelihood of generating the document $D$ of length $|D|$ generated from $n$ independent Poisson process can be estimated as follows:

$$P(D|\Lambda_D) = \prod_{i=1}^{n} e^{\lambda_i|D|(\lambda_i|D|)^{t_f^D}} \frac{(\lambda_i|D|)^{t_f^D}}{(t_f^D)!}$$  \hspace{1cm} (2.10)

Let the frequency of term $t_i$ in query $Q$ be given by $t_f^Q$. Now the likelihood of generating query $Q$ from document model $\Lambda_D$ can be estimated as follows:

$$P(Q|D) = \prod_{t \in V} p(t_f^Q|\Lambda_D)$$  \hspace{1cm} (2.11)

Let $t_f^Q$ be term frequency of term $t$ in query $Q$; similarly let $c_f$ be its term frequency in collection $C$. Let $|Q|$ and $|C|$ be query size and collection size respectively. We used Jelinek-Mercer smoothing based scoring proposed by Mei et al. [34], to score document $D$ with respect to query $Q$. The Jelinek-Mercer parameter $\delta$ was set to 0.3 in our work. This scoring is as follows.

$$Score(D, Q) = \sum_{t \in D \cap Q} \log(1 + \frac{1 - \delta}{\delta} \frac{e^{-\lambda_t|D|}(\lambda_t|D|)^{t_f^Q}}{e^{-\lambda_t|C|}(\lambda_t|C|)^{t_f^Q}}) + \log \frac{(1 - \delta)e^{-\lambda_D|N|} + \delta e^{-\lambda_C|N|}}{1 - \delta + \delta e^{-\lambda_C|N|}}$$  \hspace{1cm} (2.12)
\[ \lambda_{D,N} = \frac{|D| - \sum_{t \in Q} t f^D}{|D|} \]  (2.13)

\[ \lambda_{C,N} = \frac{|C| - \sum_{t \in Q} c f_t}{|C|} \]  (2.14)

### 2.1.4 KL-Divergence

KL-divergence retrieval is a special case of a more general risk-minimization framework [28, 63]. A query is assumed to be generated from probabilistic query model. Similarly, a document is assumed to be generated from document model. We use the multinomial distribution (Section 2.1.2) as the underlying generative models for query and document, though other models could be used. The relevance of a document is then estimated as the similarity or distance between the query model distribution and the document model. KL-Divergence [27] is used to calculate this distance. Lower distance corresponds to higher estimated relevance.

Given two probability mass function \( p(x) \) and \( q(x) \), Kullback-Liebler (KL) divergence \( D(p||q) \) between \( p \) and \( q \) is defined as following.

\[ D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \]

Suppose the query \( Q \) is generated from model \( p(Q|\theta_Q) \) with model parameters \( \theta_Q \). Similarly document \( D \) is generated from \( p(D|\theta_D) \) with parameters \( \theta_D \). Let \( \hat{\theta}_Q \) and \( \hat{\theta}_D \) be estimated query and document model; then the score of document \( D \) with respect to query \( Q \) measured by KL-Divergence function is [28]:

\[ D(\hat{\theta}_Q||\hat{\theta}_D) = - \sum_t p(t|\hat{\theta}_Q) \log p(t|\hat{\theta}_D) + \text{cons}(Q) \]

The document independent constant \( \text{cons}(Q) \) comes from expanding the log of the ratio of distributions in Eq. 2.1.4: one of the resulting terms is \( p(x) \log p(x) \), in which \( p(x) \) corresponds to \( p(t|\hat{\theta}_Q) \) in the IR implementation. Since this term does not depend
on the document at all, it is constant for all documents, and therefore can be dropped as it does not affect the ranking of documents.

It is common to estimate the query model and document model using the same approach as language modeling as described in Section 2.1.2. This scoring method looks similar to language modeling. But it incorporates the query distribution in the scoring.

2.2 Information Filtering

An information filtering system needs to process an incoming stream of documents to find relevant documents for a standing information need. Generally, potentially relevant documents must be presented to the user immediately. The system should make a binary decision (relevant/non-relevant) on each incoming document based upon the user’s information need and past accumulated document collection. This is different from traditional so-called ad hoc search described in the previous section, in which the user need and document corpus are static: traditional ad hoc search systems return a ranked list of documents the system thinks will be useful to the user’s need, allowing the user to decide which documents to look at, but a filtering system must make a decision itself to show or not show a document to a user. Many systems work by setting a threshold score, scoring every document using models similar to those in the previous section, and judging as relevant those that score above the threshold. But determining the right threshold score remains challenging.

Before we describe the problem of threshold selection and setting, a brief overview of query adaptation will help later. Adapting user query after observing relevant documents remains an interesting research problem. A user information need can drift after observing more information. This should be reflected by updating the user query (adding/removing terms) so as to potentially retrieve more relevant documents. An ineffective query term can be dropped also. Query adaptation and threshold optimization are interesting challenges for information filtering [50]. Information Filter was
conducted as track at TREC-7 [56], TREC-8 [57] and TREC-9 [58]. We describe query adaptation and threshold setting for information filtering from [50].

2.2.1 Query Adaptation

One feature inherent to filtering is its ability to use user feedback about document relevance: since the system continues to run after presenting each document to a user, it can update its models based on the user’s response to a document. One way this is commonly done is by modifying or expanding the initial query to contain new terms, drop old terms.

Query adaptation steps can be described as follows [50]:

1. Extract all the terms from all documents judged or assumed to be relevant.
2. Rank all the terms, including the original query terms in decreasing order of some “term selection value” (described below).
3. Select all terms above a threshold on term selection value or a cut-off on the number of terms.
4. Weight the terms according to usual relevance weighting formula.

The term selection value is defined as:

\[
TSV = r \cdot \log \frac{N}{n} - \log \left( \frac{R}{r} \right) - \log V
\]  

(2.15)

where:

- \( r \) is the number of past relevant documents containing the term \( t \).
- \( R \) is the number past relevant documents known.
- \( n \) is the number of past documents containing the term.
- \( N \) is the number of documents collected so far.
- \( V \) is the vocabulary size or total unique terms in the collection.

Such approaches can be seen as adaptive learning based on user feedback.
2.2.2 Threshold Score

As mentioned above, many filtering systems work using standard IR models, delivering a document to a user if the document’s retrieval score is above some threshold. Determining the threshold is not always easy. There are two general approaches: a “model-free” approach and a model-based approach.

2.2.2.1 Model-Free Threshold Setting

We want to select and modify a threshold score so as to retrieve close to the $K$ best documents, adjusting as we go while processing the stream. We need to consider the number of documents that will be retrieved with the new threshold score. A simple approach to find the score threshold with no model of relevance is as follows:

1. Run the query against the accumulated documents so far.
2. Rank documents in decreasing order of score.
3. Predict the number of future documents by looking at the score and number of documents and adjust pro-rata.

This requires an estimate of the total size of incoming stream. We may assume a constant rate of incoming documents. If query modification is done after processing some fixed amount of documents, then we need to use the modified query on the accumulated collection. The query modification was earlier described in Section 2.2.1.

2.2.2.2 Model-based Threshold Setting

Model-based threshold setting provides a theory and framework for the probabilistic measure of relevance of a document to a query. Statistical IR models assign a score based on a probability of relevance, but the score derived is generally far from the actual probability [46, 61], since actual probabilities are not needed for ranking. To convert a score to probability of relevance, the following formula was proposed in [50].

$$\log \frac{p_d}{1 - p_d} = \beta + \gamma \frac{s_d}{s_{1%}} \quad \text{(2.16)}$$
where $p_d$ is the probability of the relevance of document $d$, $s_d$ is its computed score, and $s_{1\%}$ is the average score of the top 1% of previously retrieved documents. Initial values of $\beta$ and $\gamma$ can be estimated from training data using techniques like logistic regression. The score obtained from the above equation is on the log-odds scale and can easily be converted back to $p_d$ using the sigmoid function:

$$p_d = \frac{\exp(\beta + \gamma \frac{s_d}{s_{1\%}})}{1 + \exp(\beta + \gamma \frac{s_d}{s_{1\%}})}$$

(2.17)

As we continue to accumulate information about relevance, $\beta$ and $s_{1\%}$ can be re-estimated. Re-estimating $s_{1\%}$ is easy. It can be done over the full accumulated information or on a fixed past few days of history. Robertson et al. [50] proposed an iterative gradient descent method for re-estimating $\beta$. Assume a set of $F$ feedback document whose relevance is known, and assume a total of $R$ relevant documents. The $\beta$ estimation is based on a Bayesian prior represented by $m$ mythical documents, whose estimated probability of relevance is each assumed to be correct at 0.5. The iterative estimate of $\beta^{(n)}$ with corresponding calibrated document probability (equation 2.17) for each document is as following:

$$\beta^{n+1} = \beta^n + \frac{R - \sum_{d \in F} p_d^n + m \frac{1-\exp(\beta^n-\beta^0)}{2(1+\exp(\beta^n-\beta^0))}}{\sum_{d \in F} p_d^n (1-p_d^n) + m \frac{\exp(\beta^n-\beta^0)}{(1+\exp(\beta^n-\beta^0))^2}}$$

(2.18)

Whenever we want to modify $\beta$, we iterate over the correction until the change is less than some small constant $\epsilon$. The decision to re-estimate $\beta$ can be done after observing some fixed number of documents. At any particular iteration to estimate $\beta$, query reformulation needs to be performed before any threshold setting. Also after query modification, recalculating the scores of previous retrieved documents is a good idea. This is because the score of previously retrieved documents will be different when used with the previous query and the adaptation of $\beta$ require the new scores.

### 2.2.3 Experimental Observations

The above mentioned approaches were verified in detailed experimentation by Robertson et al. [50, 49]. Runs without adaptation (no threshold adaptation and no
query modification) showed considerable good performance from runs with either of the adaptations above (query and threshold). Robertson et al. [50, 49] concluded that the problem of threshold setting and query modification for information filtering systems is complex. It presents new challenges such as query modification which is more sensitive then when applied to traditional ad hoc retrieval systems.

2.3 Topic Detection and Tracking

Topic Detection and Tracking (TDT) was a DARPA-sponsored project [1] to build systems to follow new events in an incoming stream of news stories. Three tasks were defined to act as the test beds for research groups: (1) segmenting a stream of data into the stream of stories, (2) discovering new events in an incoming stream of news stories, (3) given a sample of events and corresponding stories, track the defined events in an incoming stream of news stories. Our work in this thesis is similar to the third task called the tracking task. We briefly describe the TDT corpus, task, and some successful systems in this section.

2.3.1 Corpus

A corpus consisting of news text and transcribed speech was collected over one year from July 1994 to June 1995. The news text was from Reuters and CNN broadcast news transcripts. The resulting 15,863 stories were arranged in chronological order. A set of 25 target events were defined for tracking, and the entire corpus was annotated with respect to the target events. Each news story was flagged into three categories: YES (the news story discusses the event), NO (the news story does not discuss the event), BRIEFLY (the event is briefly mentioned or merely references the event). Each news story belonged to only one of the events. More details about the corpus can be found in [1].

2.3.2 Event Tracking Task

The aim of event tracking is to associate each incoming story with one of the 25 events already known to the systems. Systems were allowed to use a variable number
(1, 2, 4, 8, or 16) of stories discussing an event (annotated YES in the corpora) as a training set. Thus a system using 2 stories per event as training will use the first two stories in the corpus discussing the event as training data, and the rest of corpus is considered as the test data for that event.

The event tracking is similar to the information filtering problem. Instead of tracking a user query, events are to be tracked which might have temporal information associated with them. As the test set is processed chronologically, a decision about each document must be made with respect to each event before processing any subsequent stories.

2.3.3 Approaches and Evaluation

Three research groups—University of Massachusetts, Carnegie Mellon University, and Dragon Systems (www.nuance.com)—participated in this task. The proposed approaches were variations of information retrieval/filtering methods or supervised machine learning.

University of Massachusetts: This group attacked the problem as a variation of information filtering [50]. They used the training examples (positive and negative) to build short queries containing important terms describing the event[1]. The training data was used to derive a threshold score for the query. Each test story was assigned a similarity score with respect to the query. Test stories with score greater than the threshold were said to be tracked. The research group tried two different methods. The first was based on relevance feedback where they used different amount of training stories to build 10 to 100 words describing the event. The query was run against the training examples to select the threshold. In the second approach, they used a parser to extract nouns and noun phrases (from training stories). These features were weighted using two strategies: i) a feature was given higher weight if it frequently within at least one story; ii) features were weighted according to the number of times they occurred in training stories.
Carnegie Mellon University: This group used two methods: K-Nearest Neighbor classification [13] and Decision-Tree induction classification [30]. These are supervised machine learning algorithms. They used positive and negative stories from the training data to train the learning algorithms.

Dragon Systems: Dragon systems used a segmenter to assign a set of possible background topic models to a story. The segmenter used Hidden Markov Model trained on training set from external sources to assign various background topic models to a story [1]. A language model for the event to be tracked is also constructed from the positive stories in corpus. Segmenter assign score to each story against each of the background model and the event model. It reports the score as difference between the best background model and the event model. A threshold is applied to this to determine if the story is relevant to event. The background model and event models are unigram language models (Section 2.1.2).

2.3.3.1 System Comparison and Results

The comparison of systems, results, and analysis is borrowed from a technical report describing the project [1]. The above mentioned approaches were compared using standard IR and machine learning measures—Precision, F1 score, False Alarm Rate and Miss rate. Miss rate is the percentage of relevant stories a system failed to detect for a given event. If there are 100 stories relevant to an event and a system correctly identifies 25, then its miss rate is 75%. The False Alarm Rate is the percentage of stories falsely detected relevant by the story for an event. If there are 1000 stories not relevant to an event, then a false alarm rate of 0.50% is equivalent to 5 non relevant stories marked as relevant by the system for an event. Precision is the fraction of the retrieved stories relevant to the event; recall is the fraction of relevant stories retrieved by the system. F1 Score is a measure that combines recall and precision to give a balanced measure. An ideal system will have high recall, high precision and thus a high F1 score (see also Section 4.3).
Table 2.1 compares different systems by CMU, Univ of Massachusetts and Dragon Systems. We see that systems have varying degree of error rates. CMU K-Nearest Neighbor system has a low miss rate but high false alarm rate. CMU tuned its classifier for threshold on F1 Score which balances recall and precision. It is reported that University of Massachusetts relevance feedback systems did not do well because it used a small number of features. It is also reported that minor variations in query formation can result in substantial difference in the effectiveness.

On an average, 54 stories per event could have been tracked and 8377 stories should not have been tracked. Achieving a miss rate of 50% and false alarm rate of 0.25% would mean 27 stories per event were tracked correctly and less than 20 were falsely tracked.

The following conclusions were made by the report:

1. Describing an event using 20-50 single terms seems preferable. A smaller set of terms provides higher precision but poor recall. Larger sets of terms have the opposite effect.

2. A better set of features (such as noun phrases) is more effective at producing good results.

3. Combining multiple approaches to decide relevancy of a story can be more effective.

4. An interesting observation was made to smooth event models with small stories. For example, events with only one story in the training data. To smooth such event models, use that story as a query into the training database, and use the stories retrieved as smoothing material.
Some suggested techniques for improvement were: query expansion based on pseudo relevance feedback, evidence combination using unsupervised and supervised interactive learning.

2.4 TREC KBA Track

Our work in this is thesis is specifically motivated from the TREC KBA track’s “Cumulative Citation Recommendation” (CCR) task [37, 40]. The CCR task is similar to information filtering. The topics used in the track are entities from Wikipedia and Twitter. A system performing the CCR task should propose documents from an incoming stream that a human user would want to cite in updating the Wikipedia article for a target topic. The system should iterate over documents arriving hourly in chronological order, for each making a recommendation of whether it is “vital” or not as a citation. The system should not go back to promote a document based on information that arrives after its recommendation has been made. The decision of vital/non-vital about a document for a topic cannot be postponed.

2.4.1 Corpus and Topics

The corpus creation, topic creation, and judgment of documents are described in detail in [40, 37] (http://trec.nist.gov/pubs/trec22/trec2013.html); here we provide some important details.

A corpus of size around 20 terabytes was provided by NIST for the track. This is a time-ordered corpus with over a billion of documents consisting of blog, news, and web pages collected over twelve thousand contiguous hours. The web pages were crawled starting 5 Oct 2011 and ending 13 Feb 2013. The documents are stored in subdirectories, where the directory name gives the day and hour the document was crawled from the web and stored. Each document in the corpus is also associated with metadata, specifically sentence segmentations and part-of-speech tags. Further, if a sentence mentioned any of the topics provided by the organizers, it was flagged with this information.
In order to help researchers build systems, 121 test topics were selected from Wikipedia for which around 50 thousand documents from the corpus were judged. These topics consist of Wikipedia URLs such as: http://en.wikipedia.org/wiki/Carla_Katz and http://en.wikipedia.org/wiki/Fargo-Moorhead_Symphony_Orchestra.

The documents were selected for judging by pooling all the documents which mention any term in a list of manually-created names/aliases of the topic. The following two step procedure describes the pooling of documents for judgment:

1. A list of possible alias/names of 121 Wikipedia entities were created by a human expert.

2. All the documents which mention any terms in the created aliases were pooled for judging.

Each of these pooled documents were judged by multiple human assessors for the topic it mentions. Pooled documents were judged according to the following categories by human assessors [40, 37]:

**Vital:** These documents contain very vital and timely information about the topic which would cause a human to update his existing knowledge about the topic.

**Useful:** This is citation-worthy document about a topic. However, such documents do not add new knowledge about the topic. The assessor is already up to date with the knowledge about this topic.

**Neutral:** Such documents are informative about the topic but are not citation-worthy. They provide very little information about the topic.

**Garbage:** No information about the topic. Topic names appear in the document without any context. For example spam documents about the topic.

The above judging procedure is very subjective. The assessors were asked to judge the document for a topic as if they are building a profile or dossier for the topic. The assessors were asked to start with no information about the topic. A document is “useful” when building a profile of topic from scratch. A “vital” document will cause a change in the existing profile for the topic. To reduce the subjectivity, a single
document was assessed by multiple assessors. There was 76% assessor agreement for vital document [40], which is much higher than typically observed in IR.

2.4.2 Work by Participating Groups

Several teams with different systems participated in the TREC KBA CCR task in TREC 2012 and TREC2013. We will compare our system to these in our experiments.

We first describe an “oracle” system built by the organizers to be the official baseline. This system assigns a “vital” rating to every document that matches a surface form name of a topic. It assigns the confidence score based on the length of the observed name. It is called an “oracle” because it uses hand-picked names of topics by the organizers; normally this would not be allowed for a TREC submission.

University of Illinois Team: The Graduate School of Library and Information Science at University of Illinois proposed an interesting approach [42] to the CCR task. To reduce the corpus size, they filter documents containing all the terms in the topic label. This is similar to “Boolean AND” search of topic label on the corpus. For example, if the topic is http://en.wikipedia.org/wiki/Jamie_Parsley, then the system considers only documents containing both the term “jamie” and “parsley”. By doing this, they achieve a relatively high precision at the cost of recall. The system score documents with respect to topic using a modified KL-Divergence measure which accounted for document length. The lengths of vital documents of the topic from training period are used to estimate a log normal distribution $\mathcal{N}_L$ over length with standard deviation $\hat{\sigma}$ and mean $\hat{\mu}$. The length factor for a document with size $n(D_i)$ to be used in scoring with respect to topic is given by following equation.

$$l(D_i) = \mathcal{N}_L(\log n(D_i), \hat{\mu}, \hat{\sigma})$$

$$\text{Score}(D_i, T) = \log l(D_i) - \sum_{f \in T} P(f|\theta_T) \log \frac{P(f|\theta_T)}{P(f|\theta_i)}$$

where:
• $D_i$ is document to be processed for scoring.
• $T$ is topic
• $l(D_i)$ document length factor given by equation 2.19. This is normalized length of document against the normal distribution of documents of the training data.
• $f$ is feature of topic.
• $\theta_T$ is estimated topic distribution.
• $\theta_i$ is estimated document distribution.

Documents with score greater than the threshold score $t$ are considered as vital. The threshold $t$ is determined from vital documents in the training set. The document distribution $\theta_i$ is a multinomial distribution over terms in the document. The estimation of the topic distribution is more complex. It is a combination of an estimated multinomial distribution from the training documents and an estimation of a distribution from newly predicted vital documents. The distribution is estimated as follows:

$$
\theta_iF = \lambda \frac{n(F, Tr)}{n(Tr)} + (1 - \lambda)\theta_V
$$

(2.21)

In the above equation $F$ is a feature or term in the topic. $n(F, Tr)$ is the number of occurrence of a feature $F$ in vital documents in the training data. $n(Tr)$ is the number of terms in the vital documents in the training data. $\lambda$ is a factor balancing the static distribution over training data and the dynamic distribution $\theta_V$, estimated from newly predicted vital documents for the topic. Then the dynamic distribution $\theta_V$ is estimated using following equation:

$$
\theta_t = \frac{n(f, V_t) + \mu P(f | \theta_{t-1})}{n(V_t) + \mu}
$$

(2.22)

Here $\theta_{ft}$ is the current distribution estimation, $V_t$ is the current set of vital documents predicted by the system, $\theta_{t-1}$ is the previous distribution, and $\mu$ is the extent to which we want the model to diverge from from previous model. The frequency function $n(V_t)$ is the total number of features or terms in the vital set of documents, $n(f, V_t)$ is the frequency of the feature in the vital set of documents. UIUC conducted a large
number experiments with different settings. Experiments confirm their approach is effective [42].

**BIT & MSRA Systems:** The Beijing Institute of Technology (BIT) and Microsoft Research Asia (MSRA) together proposed systems [39] for the CCR task. Their approach is first to retrieve possible relevant documents for a topic. On this set of documents, they apply supervised machine learning algorithms to filter vital documents. Related topics for a topic were found from Wikipedia pages of the topic; query expansion combines the original topic name and their related topic name to construct phrase queries. With this expanded query, the system searches for relevant documents among the likely vital documents. This reduces the search space from the full 20 TB collection. On the other hand, this could cause bias in results; by retrieving documents with the expanded query, the system is implicitly using information from the future to retrieve good documents. In the next step of processing, they apply Random Forest classification with the training data of the corresponding Wikipedia page of the topic to classify documents as vital/non-vital. This further improves their result. They used a Learning-to-Rank method implemented with RankLib which again improves their result. For training purpose they used a variety of features—document length, document source, document publishing data, document similarity with Wikipedia page of the topic etc.

**UDEL Group:** The University of Delaware group proposes a system [38, 41] based on associating related topics of the main topic and assigning weights to each of them (related topics) using training data. To reduce the search space, they filter documents containing the topic names and other possible alias names of the topic. The possible alias names of a topic are found using DBpedia (http://wiki.dbpedia.org/About). They use the following equation to assign a score to a document with respect to a topic \( T \)

\[
score(D, T) = \alpha \times mention(D, T) + \beta \sum_{t \in R(T)} occ(D, t) \times w(T, t)
\]

(2.23)

where:
- $mention(D, T)$ is binary function indicating whether document $D$ mentions $T$ or any of its name variations.

- $R(T)$ is a set of topics related to the main topic $T$.

- $w(T, t)$ is the weight of a related topic $t$ to the main topic $T$ to control influence.

- $\alpha$ and $\beta$ are coefficients to balance the impact of two scoring components in the above equation.

All of the documents with score greater than a threshold score $\gamma$ are considered as vital. The related topics $t$ for main topic $T$ are found by parsing the html text of Wikipedia page of the main topic $T$. The association weight $w(T, t)$ of a related entity is found using training data, specifically documents from the corpus with their corresponding human judgment for a topic. They use a greedy algorithm to assign weights. The algorithm maximizes the performance gain on the training set of documents by incrementally adding the related topics and incrementally increasing their association weight [41]. Coefficients $\alpha$, $\beta$ and threshold score $\gamma$ were manually set.

Most of the approaches for CCR task exploit alternate names for a topic and related topics found on the Wikipedia page of the main topic. A query consisting of these names with weights attached to each term produced good results. For most of the teams, this simple approach was a strong baseline.
Our work in this thesis is motivated from information filtering [56, 57, 58, 4, 50, 40]. The aim of an information filtering system is to help a user build and update knowledge about a topic from an incoming stream of documents. The user’s need can be a general topic such as “Israel-Palestine conflict”, a concept like “minimum wage”, an entity such as “Satya Nadella”, or a particular specific query like “ebola quarantines sierra leone”. The filtering system should have mechanisms to consume new information and accumulate the consumed knowledge. In this chapter we describe our system for accumulating information and the process of consuming new information.

3.1 Problem Definition

The problem we are trying to solve is essentially filtering a very large, streaming corpus: given a “standing query” \( Q \), determine whether each document \( D \) that appears during some time period \( Y \) is relevant or not. We intend to base our approach on the traditional IR scoring models described in Section 2.1 (and re-capped in Section 3.2 below). These approaches require statistics such as \( N \), \( df \), \( cf \), and \( avdl \) computed from documents in a corpus, and the values of those statistics may depend heavily on what information we have chosen to store from document appearing during the time period leading up to \( Y \).

In particular, since the amount of streaming data is so large, we assume that we cannot store all of it. We can only keep the most recent \( X \) days, and corpus statistics used in retrieval model scores will have to be based on data from those \( X \) days.

To see why this matters, we refer to Figures 3.1 and 3.2. These two figures show changes in average document length (which is important to BM25 scoring) over time.
Figure 3.1: Average Document size on each day of collection. Notice the change in scale of two figures.
Figure 3.2: Average Document size on consecutive 5 days of collection. Notice the change in scale of two figures and smoothing the plot due to cumulative effect.
in the TREC KBA corpus. As described in Section 2.4, the corpus consists of 20TB of data from three different sources, obtained from the time period October 5, 2011 through February 13, 2013 (a total of 498 days). For the first roughly 30 days, average document length is only slightly above 200 terms. After that, it suddenly rises to the 400-500 term range for a while, except for sudden spikes at day 150 and approximately day 190. After day 210 or so, it again drops to just over 200 for about 50 days, then jumps up to an average length of around 1500 terms. From there it declines gradually over the remaining 200 days.

If only one day of data were to be used to compute statistics like $avdl$, it could have a fairly substantial effect on BM25 score. Consider a document of length 500 that contains a query term 5 times. Suppose the stored history indicates that the term appears in 100 documents out of 10,000 stored. If average document length among those 10,000 is 200, the BM25 score is about 7.008. If average document length is 500, BM25 score rises 26% to 8.827. If average document length is 1000, BM25 score goes up to 9.664 (38% larger than 7.008; 9.5% higher than 8.827). Even if five days of data were used, as Figure 3.2 shows with $avdl$ computed in a sliding window of 5 days, which 5 days are used could have a substantial effect.

3.2 System Process Flow

Here we describe the generic process flow of our system. We will process all the documents that appear during a given day for each query and retrieve the top $K$ documents as most “vital” for the query. The main steps for processing documents each day are as follows.

For each day $Y$ (the present day) over some time period:

1. Compute corpus statistics $df$, $ctf$, $avdl$, etc from $X$ days of data prior to the present day.

2. Select a subset of $N$ documents from all documents that arrived on the present day $Y$ to be scored for relevance.

3. Score these $N$ documents for each standing query $Q$ using one of the retrieval models described in Section 3.3.
4. Pick the top $K$ from these $N$ scored documents as the “vital” documents to return.

Figure 3.3 illustrates our system. The incoming stream of documents on the present day are processed to create an inverted index. This index will be used for the retrieval in steps 2 and 3 above, and will also join the other history indexes to be used in step 1 for days in future iterations. There are few possible approaches to select $N$ documents in step 2. We describe and compare few of these in Section 4.5.1. The retrieval model component utilizes the accumulated information in history indexes to rank the documents of the present day with respect to a query. The top $K$ documents after ranking are considered as the most “vital” documents for the topic.

3.3 Retrieval Models

We experiment with four different retrieval models: BM25, Poisson Distribution Model, Language Model, and the KL-Divergence model. We plugged these models into
step 3 of the system process flow described in Section 3.2, creating different filtering systems. The retrieval model theory and scoring equations are described in more detail in Section 2.1; in this section, we briefly describe the parameter settings for these models suitable to our system. We use query and topic interchangeably in the following discussion; in general, in IR experimentation, a “topic” will have an associated keyword query.

3.3.1 BM25

BM25 is a very popular and robust model based on the Probability Ranking Principle. BM25 assign a score to each document for its relevance to a user query. The BM25 score of a document is the sum of the weight of query terms:

\[
score(Q, D) = \sum_{t \in Q} \frac{tf_t \cdot (k_1 + 1)}{tf_t + k_1 \cdot (1 - b + b \frac{|D|}{avdl})} \cdot \log \frac{N - n_t + 0.5}{n_t + 0.5}
\]

where:

- \(Q\) and \(D\) are a query and document, respectively.
- \(tf_t\) is the frequency of term \(t\) in document \(D\).
- \(|D|\) is document length in terms.
- \(avdl\) is average document length observed in the corpus.
- \(k\) and \(b\) are free parameters to be set based upon previous study and experiments.
- \(n_t\) is the number of documents (of \(N\) in the corpus) that term \(t\) occurs in.

We set parameters \(k_1\) and \(b\) manually to 1.75 and 0.5 respectively, as these values have been shown to work well across a range of retrieval scenarios.

3.3.2 Poisson Query Generation Model

Poisson based distribution is described in more detail in Section 2.1.3. Scoring of documents with respect to query is based on the query-likelihood principle [34]; the specific scoring function is:
\[ \text{Score}(D, Q) = \sum_{t \in D \cap Q} \log \left( 1 + \frac{1 - \delta}{\delta} e^{-\lambda_{t,D}|Q|} \left( \frac{\lambda_{t,D}}{|D|} \right)^{tf_t^Q} \right) \]

\[ + \log \left( 1 - \delta \right) e^{-\lambda_{D,Q}|Q|} + \delta e^{-\lambda_{C,N}|Q|} \]

with \( \lambda_{D,N} = \frac{|D| - \sum_{t \in Q} tf_t^D}{|D|} \) \hspace{1cm} (3.2)

and \( \lambda_{C,N} = \frac{|C| - \sum_{t \in Q} tf_t^C}{|C|} \) \hspace{1cm} (3.3)

where:

- \( Q \) and \( D \) are a query and document, respectively.
- \( \lambda_{t,D} \) is the ratio \( tf_t \) (the frequency of term \( t \) in document \( D \)) to the length of \( D \).
- \( |Q| \) is query length in terms.
- \( avdl \) is average document length observed in the corpus.
- \( cf_t \) is the total number of times term \( t \) appears in the entire collection. This is fetched from past history indexes.
- \( \delta \) is a free parameter to be set based upon previous study and experiments.

We set parameters \( \delta \) to 0.3.

### 3.3.3 Language Model:

Language Model based document scoring is based on same principle of query-likelihood generation. The underlying document model is considered as multinomial distribution. The language modeling approach is described in detail in Section 2.1.2.

Estimation of the probability of generating a query term from a document requires smoothing; we used Dirichlet smoothing. The probability of generating a term \( t \) with term frequency \( tf \) from document \( D \) is computed as follows:

\[ p(t|D) = \frac{tf + \mu p(t|C)}{|D| + \mu} \] \hspace{1cm} (3.4)
We tried two different values of the free parameter $\mu$, specifically we set it to 2500 and 5000 for different system runs. The probability of generating term $t$ with collection term frequency $ctf$ from collection $C$ is as follows:

$$p(t|C) = \frac{ctf}{|C|} \quad (3.5)$$

Finally, the score of document $D$ with respect to query $Q$ can be now computed as follows:

$$score(D, Q) = \sum_{t \in Q} \log p(t|D) \quad (3.6)$$

### 3.3.4 KL-Divergence

KL-Divergence described in Section 2.1.4, is based on the idea of similarity of two probability distribution. The user query is assumed to be generated from some estimated probability distribution. Similarly, the document is generated from different probability distribution. The score of document with respect to query is similarity between the two probability distribution (query and document). The similarity is measured using KL-Divergence measure [27]. The score of document $D$ with respect to query $Q$ is as follows:

$$score(D, Q) = -\sum_{t \in Q} p(t|Q)\log p(t|D) + cons(q)$$

The probability of generating a query term from document ($p(t|D)$) is computed using equation 3.4, described previously in Section 3.3.3. We used the Dirichlet prior $\mu$ set to 2500 to smooth the estimation of term probability. The probability of generating a term from the query ($p(t|Q)$) was computed similarly.

### 3.4 System Design

To score documents using the statistical IR models described in Section 2.1, we need to compute corpus statistics $df$ (document frequency), $cf$ (collection term frequency), $avdl$ (average document length), etc. These statistics can be pre-computed and stored in an inverted index [3]. There are many open-source software tools which
create inverted indexes—Lucene (http://lucene.apache.org/), Indri (http://www.lemurproject.org/), and Terrier (http://terrier.org/), to name a few. We used Indri to index the corpus, as it has been used frequently in our lab’s research.

An inverted index is a term-oriented mechanism for indexing a text corpus to speed up an information retrieval task. An inverted index consists of two components: vocabulary and posting lists. The vocabulary is the unique terms in the text corpus; the posting lists contain the list of documents containing the term of vocabulary, along with frequency counts. A posting list would also include the document frequency and collection term frequency of the term. Other kinds of metadata can be stored in a posting list as well, for example the positions of terms within a document. The creation of an inverted index makes it possible to quickly collect term statistics such as term frequency, document frequency, etc, for rapid computation of retrieval model scoring functions.

In order to be able to use $X$ days of prior history without using any “future” data—that is, data that our system should not be aware of because it has not yet arrived—we create a separate inverted index for each day of a streaming corpus. Creating indexes on a per day basis gives us flexibility in terms of scalability and efficiency. We have the option of reading just one day of history or all available history depending upon the allowable scalability of our system. Increasing the number of history indexes increases the memory consumption of our system. Creating inverted index for each day allowed us to vary the amount of history used to compute term statistics.

However, we observed that the Indri engine became unreliable and its performance degraded after increasing history index size beyond 25. We are not yet sure why this is the case.

The document text was stemmed using the Krovetz stemmer [26] and a common set of stopwords were removed before feeding the corpus into the Indri indexing engine. A total of 498 indexes were created, each representing a single day of the corpus. Before processing a documents on a particular day, we read inverted indexes of previous days to collect various corpus level statistics. We call the previous $X$ days of inverted index

---

1 However, we observed that the Indri engine became unreliable and its performance degraded after increasing history index size beyond 25. We are not yet sure why this is the case.
the *history index* of size $X$.

To re-cap, our system works as follows: For each day $Y$ (the *present day*) over some time period:

1. Compute corpus statistics $df$, $cf$, $avdl$, etc from $X$ days of data prior to the present day, using the inverted indexes that have been created from the documents that arrived on those days.

2. Select a subset of $N$ documents from all documents that arrived on the present day $Y$ to be scored for relevance. This could be done with any of the models described in Section 3.3; we used a language model for this step for reasons described in the next chapter.

3. Score these $N$ documents for each standing query $Q$ using one of the retrieval models described in Section 3.3. By varying the model used in this step, we create different filtering systems to compare.

4. Pick the top $K$ from these $N$ scored documents as the “vital” documents to return.

Since these steps will be performed over the course of hundreds of days for a single experiment, computation time could potentially be quite long. Step 2 is meant to increase efficiency by only requiring that a small subset of documents be scored on any given day.
Chapter 4

EXPERIMENT AND RESULTS

We conducted an experiment on the TREC KBA track data using the filtering system framework described in Chapter 3. We use different IR models into the scoring and ranking step (step 3 in Section 3.2). Each of these IR models will be different system runs. We compare these models on different types of queries and accumulated information (history indexes) in Section 4.4.

4.1 Corpus

We used the TREC KBA 2013 corpus (http://trec-kba.org/trec-kba-2013.shtml) as a testbed for our filtering system [40]. The corpus is described in Section 2.4.1; here we recap some of its properties and also describe in more detail the properties that are relevant to our work.

The corpus is a time-ordered collection of over a billion documents fetched from the web and other sources. The primary sources of documents were from news, social, blog, and publication sites. The documents were crawled from the web over 498 days, starting in October of 2011 and ending February 2013. Each document in the corpus has a timestamp reflecting the time it arrived in the stream.

The compressed size of the corpus is around 5.5 terabytes. On disk, the corpus is stored in separate directories, each containing every document that arrived during a particular hour on a particular day. For example, “2011-10-05-02” contains documents crawled on 5th Oct 2011 between 0200 hrs and 0300 hrs. There are $498 \cdot 24 = 11,952$ total directories. The stored documents have been processed by the designers of the collection with a variety of tools for different types of text content. Each document has the following contents associated with it:
4.2 Topics

One of the goals of KBA CCR task is to find sources of information (documents) that would make good citations for facts presented on a Wikipedia page. To that end, the KBA track provided 121 Wikipedia pages to be used as user needs (and the source of queries) for a filtering system. Each topic consists of a Wikipedia URL, e.g. http://en.wikipedia.org/wiki/Red_River_Zoo. Each of the URL’s can be processed to get human readable topic label. For example topic label for http://en.wikipedia.org/wiki/Red_River_Zoo is “Red River Zoo”.

For each of the topics, possible alias names (possible topic labels) were constructed manually by TREC KBA organizers (see Section 2.4.1). For example, a possible alias for topic Using these aliases, the organizers retrieved documents to be judged from the full corpus. These documents were judged with respect to the Wikipedia topic on four relevancy levels—vital, useful, neutral or garbage. Of these, “vital” is most important; a “vital” document provides timely information about a topic which would cause a human to update his or her existing knowledge on the topic. The different relevancy levels and judging process is described in more detail in Section 2.4.1. In this work we consider only the vital documents as relevant documents for a topic.

Vital documents are distributed unevenly: some of the topics have a large number of vital documents and some have very few. Table 4.1 show some topics with a
**Figure 4.1:** Empirical cumulative distribution function (ecdf) plot of number of vital documents per topic in the corpus.

<table>
<thead>
<tr>
<th>Wikipedia Topic Name</th>
<th>Total Vital Docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edgar Bronfman, Jr.</td>
<td>318</td>
</tr>
<tr>
<td>Benjamin Bronfman</td>
<td>194</td>
</tr>
<tr>
<td>Hjemkomst Center</td>
<td>189</td>
</tr>
<tr>
<td>Austral Group</td>
<td>1</td>
</tr>
<tr>
<td>Bernard Kenny</td>
<td>1</td>
</tr>
<tr>
<td>Chiara Nappi</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 4.1:** Some examples of topics with high and low numbers of vital documents in the corpus.

<table>
<thead>
<tr>
<th>No. of Vital Documents</th>
<th>Number of Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>1–20</td>
<td>66</td>
</tr>
<tr>
<td>21–100</td>
<td>24</td>
</tr>
<tr>
<td>&gt; 100</td>
<td>17</td>
</tr>
</tbody>
</table>

**Table 4.2:** Table showing topics broken out according to the number of vital documents they have in the corpus.
large number of vital documents and some with very few. Figure 4.1 is the empirical cumulative distribution function (ecdf) plot for number of vital documents per topic. From this plot, we can estimate that 80% of topics have fewer than 100 vital documents, and 60% have fewer than 20 vital documents.

Table 4.2 show the break up of topics according to number of vital documents in the corpora. The message here is that there is a large number of topics with fewer than 50 vital documents in the whole corpus. This is an example of the challenge of an information filtering system. We have a large amount of information being published every day, and a filtering system has to process this information to find a very small amount of relevant knowledge regarding a topic.

4.2.1 Topic Query Generation

An information filtering system requires a profile (e.g. terms describing topic) to be associated with a topic; the profile is what is used to score documents. In our system, the profile is text representing information about the topic. The text profile for a topic will be fed as a query to our system.

In particular, the Wikipedia topic URL can be associated with human readable label. For example, the URL http://en.wikipedia.org/wiki/Fargo-Moorhead_Symphony_Orchestra could be mapped to “Fargo Moorhead Symphony Orchestra” by simply removing the domain and top-level directory, then segmenting the remaining terms by punctuation marks. Each topical URL will be processed this way to be used as short queries to our system. The labels will be converted to lower case letters and stop words removed before using it as a query.

Short queries provide a relatively small amount of “profile” information for estimating the relevance of documents. We would also like to test what happens when a larger amount of information is available. To do that, we used DBpedia to get additional text associated with each Wikipedia URL. DBpedia (http://wiki.dbpedia.org/About) is a community-built semantic database, consisting of RDF triplets derived from semi-structured Wikipedia pages.
We specifically used DBpedia to get a Wikipedia page’s abstract. This text is a short description about the topic in the abstract section of its corresponding page on Wikipedia. An example abstract from the Wikipedia page http://en.wikipedia.org/wiki/Corn_Belt_Power_Cooperative is:

Corn Belt Power Cooperative is a generation and transmission electric cooperative in northern Iowa. It currently supplies to 12 of its member cooperatives in 28 counties in northern Iowa. It owns and maintains a coal-fired plant and a gas combustion turbine generator and is one of the owners of the Duane Arnold Energy Center. The Cooperative also has access to renewable energy sources such as wind and hydroelectric power.

Therefore, with each topic we associate two types of query text—a short label query and a long abstract query. All 121 topics had a short label, and 93 of them had associated abstract text.

4.3 Evaluation Measures

We use common information retrieval measures—recall, precision, and F1 score—for comparison of different models.

**Recall:** Recall is the fraction of relevant documents retrieved:

\[
Recall = \frac{|\{\text{Relevant Documents}\} \cap \{\text{Retrieved Documents}\}|}{|\{\text{Relevant Documents}\}|}
\]

**Precision:** Precision is the fraction of retrieved documents that are relevant:

\[
Precision = \frac{|\{\text{Relevant Documents}\} \cap \{\text{Retrieved Documents}\}|}{|\{\text{Retrieved Documents}\}|}
\]

As mentioned earlier, pooled documents in the corpus were graded on four levels of relevancy to a topic. For comparison of retrieval models, we consider only vital documents as relevant. We calculate these measures for the complete set of documents retrieved over the 498 days of the corpus, i.e. the set of “retrieved documents” in the above expressions is the union of all documents retrieved in the top \( K \) for all 498 days of the corpus.
**F1 score:** Ideally, a high recall and high precision system is desired. But, empirically speaking, as recall increases precision tends to decrease. The F1 score is the harmonic mean of recall and precision, meant to give a balanced view of the system. Good retrieval models exhibit high F1 score. It is calculated from recall and precision as follows:

\[
F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}
\]

4.4 Retrieval Models

The system described in Section 3.2 has two separate steps at which retrieval and/or relevance scoring is performed. The first (step 2) involves selecting a subset of the full set of documents for day \( Y \) to be scored. The second (step 3) involves scoring and ranking those selected in step 2. We did not vary the approach used in step 2: the initial set of documents are retrieved using a query-likelihood language model with the \( \mu \) smoothing parameter set to 1. Setting \( \mu \) to 1 is the minimum smoothing possible with Indri; it makes the effect of corpus statistics on retrieval almost minimal. We use this value of \( \mu \) for two reasons:

1. To ignore, as much as possible, “future evidence” from the same day. If we are scoring the very first document that arrives on day \( Y \), we should not use corpus statistics derived from documents that arrive later in the day. Using a very low level of smoothing in step 2 ensures that corpus statistics from day \( Y \) will have negligible effect on the documents to be scored in step 3.

2. To reduce the set of documents that would need to be scored for each topic. This is important because it reduces the total number of documents that need to be scored over 498 days from hundreds of millions to only about 500,000.

Furthermore, with such low smoothing, the documents retrieved will be dominated by those that contain *all* of the query terms.

For step 3, we experimented with different retrieval models, specifically those listed in Section 3.3. In reporting results, we use short names to describe the systems: LM for language model, PM for Poisson model, KL for KL-Divergence and BM for BM25 model. For some models, we tested different values of \( \mu \) for Dirichlet smoothing.
<table>
<thead>
<tr>
<th>Run Name</th>
<th>Precision</th>
<th></th>
<th>Recall</th>
<th></th>
<th>F1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td>$K = 10$</td>
<td>$K = 30$</td>
</tr>
<tr>
<td>LM-mu1500 ($N = 10,000$)</td>
<td>0.20</td>
<td>0.20</td>
<td>0.52</td>
<td>0.62</td>
<td>0.29</td>
<td>0.31</td>
</tr>
<tr>
<td>LM-mu1500 ($N = 1,000$  )</td>
<td>0.20</td>
<td>0.20</td>
<td>0.53</td>
<td>0.62</td>
<td>0.29</td>
<td>0.31</td>
</tr>
<tr>
<td>Ind-cheat</td>
<td>0.20</td>
<td>0.22</td>
<td>0.57</td>
<td>0.69</td>
<td>0.30</td>
<td>0.33</td>
</tr>
<tr>
<td>Ind-cheat-Fb</td>
<td>0.20</td>
<td>0.21</td>
<td>0.61</td>
<td>0.72</td>
<td>0.31</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Table 4.3: Comparison of language model based filtering systems with different initial retrieved document set size. Indri cheating runs are also provided for comparison.

In these cases, the system name will include the word “mu” followed by its value. For example, LM-mu1500 will mean experiment with language modeling and $\mu$ set to 1500.

In our discussion below, a reference to “present day” means the current processing day for the system.

Two experimental system which we call “cheating runs” are also described here. We performed two experiments that use information from future documents; these are referred to as “Ind-cheat” and “Ind-cheat-Fb”. “Ind-cheat” used the Indri search engine with language modeling ($\mu = 2500$), the Wikipedia URL as query, and the full collection of documents on the present day. This experiment does not use Indri indexes from previous days. The returned result is a ranked list of documents for a topic from the present day, but based on “future evidence” from the same day. “Ind-cheat-Fb” is the same as the “Ind-cheat” but further includes pseudo-relevance feedback from the top 10 documents to re-rank the documents.

4.5 Results

In our experiments, we will vary the following:

- $X$, the size of the history to use in computing corpus term statistics.
- $N$, the size of the initial set of documents to retrieve.
- $K$, the size of the set to return as “vital”.
- Retrieval model, using the four models described in Section 3.3.
- Query type, either short Wikipedia URL-derived label or long abstract text.
\begin{table}[ht]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{$K$ ($N = 1000$)} & \textbf{Indri-stp2} & \\
\hline
\textbf{$K = 10$} & 0.20 & 0.64 & 0.30 \\
\textbf{$K = 30$} & 0.21 & 0.72 & 0.33 \\
\textbf{$K = 100$} & 0.21 & 0.79 & 0.33 \\
\textbf{$K = 200$} & 0.21 & 0.82 & 0.34 \\
\textbf{$K = 300$} & 0.21 & 0.82 & 0.34 \\
\hline
\end{tabular}
\caption{Precision and recall comparison at different values of $K$. The step 2 of system process flow (Section 3.2) is used to return top $K$ documents as vital for a topic.}
\label{tab:precision_recall}
\end{table}

- Per-topic vital document count (in the TREC KBA gold standard data).

Throughout the subsections below, we report results for $K = 10$ and $K = 30$ and for a selection of the retrieval models. Section 4.5.2 reports on the effect of selecting $N$. Section 4.5.3 reports on the effect of query type and vital document count. Section 4.5.4 reports on the effect of history size $X$. Finally, in Section 4.5.5, we compare our results to those of systems submitted to the TREC KBA track.

### 4.5.1 Effect of Reducing Search Space

In step 2 of the system process flow (Section 3.2), we retrieve a set of $N$ documents from the current day inverted index. The set of $N$ documents is then scored by a retrieval model in step 3 to suggest top $K$ documents as vital for the topic. We used Indri language model retrieval engine to fetch $N$ documents. We provide some other approaches possible for step 2.

We perform a experiment with Indri as retrieval engine in the step 2, as described in Section 3.4. We call this system “Indri-stp2”. This system retrieve $N = 1000$ documents using the topic label as query and pick top $K$ documents as vital for the topic. Table 4.4 show precision and recall comparison for “Indri-stp2” at different values of $K$. The recall of the system reaches maximum value of 0.82 at $K = 200$. This suggest that the system was able to retrieve 80% of the vital documents for the topics at $K = 200$. 

50
<table>
<thead>
<tr>
<th>$K$</th>
<th>AND-stp2</th>
<th>OR-stp2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>$K = 10$</td>
<td>0.18</td>
<td>0.23</td>
</tr>
<tr>
<td>$K = 30$</td>
<td>0.18</td>
<td>0.24</td>
</tr>
<tr>
<td>$K = 100$</td>
<td>0.18</td>
<td>0.25</td>
</tr>
<tr>
<td>$K = 300$</td>
<td>0.18</td>
<td>0.25</td>
</tr>
</tbody>
</table>

**Table 4.5:** Precision and recall comparison at different values of $K$ for two systems differing only in use of boolean queries.

There are other possible approaches for step 2. To reduce the search space, the system can consider only documents containing all the terms of the topic label. This is similar to “AND” query of the terms over the current day documents. Similarly, we can do “OR” query of the terms, i.e. consider only documents containing any of the terms of the topic label. We replace the Indri retrieval in step 2 with “AND” query of terms. We call this system “AND-stp2”. Similarly, we create “OR-stp2” system. Table 4.5 show precision, recall and F1 for “AND-stp2” and “OR-stp2” at different values of $K$. The recall and precision in Table 4.5 for “AND-stp2” quickly reaches maximum because there are very few documents which contain all the query terms. So this system will drastically reduce the search space. The recall value of “OR-stp2” increases with increase in $K$.

We design a experiment to show that the selection of $N$ documents in step 2 of the process flow using Indri retrieval engine is similar to selecting documents which contain any of the terms in topic label. We use the topic label as the user query to the experiment. We replace the Indri retrieval in step 2 with “OR” query of terms of the topic label to retrieve documents from current day inverted index. The filtering system will only consider the documents from the present day which contain any of the terms of the topic label. We use BM25 retrieval model in step 3 of the system process flow. We call this system “BM-OR”. We compare “BM-OR” with a system using Indri as retrieval engine in step 2 and BM25 as retrieval model in step 3. We call this system “BM-Indri-mu1”. This system retrieve $N = 1000$ documents in step 2 of the system process flow. Thus the two systems only differ for step 2 of system.
<table>
<thead>
<tr>
<th>Run Name</th>
<th>Precision</th>
<th></th>
<th>Recall</th>
<th></th>
<th>F1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td>$K = 10$</td>
<td>$K = 30$</td>
</tr>
<tr>
<td>BM-OR</td>
<td>0.20</td>
<td>0.21</td>
<td>0.43</td>
<td>0.52</td>
<td>0.27</td>
<td>0.30</td>
</tr>
<tr>
<td>BM-Indri-mu1</td>
<td>0.21</td>
<td>0.21</td>
<td>0.46</td>
<td>0.54</td>
<td>0.29</td>
<td>0.31</td>
</tr>
</tbody>
</table>

Table 4.6: System comparison for two different approaches in step 2 of system process flow. BM-OR uses “OR” query to retrieve documents in step 2 and BM-Indri-mu1 uses Indri retrieval engine to retrieve 1000 documents in step 2.

process flow. We compare the results at $K = 10$ and $K = 30$ values. Table 4.6 shows the two system comparison. It is clear from the table that the two systems are similar. Figure 4.2 shows topic wise recall and precision of the two systems at $K = 10$. We see good correlation for recall and precision for two systems.

4.5.2 Effect of Retrieved Set Size

Table 4.3 shows a comparison of language model-based systems (including our two “cheating” runs) with different sizes of initial retrieved set size $N$. We note two things:

1. The cheating runs provide substantially higher recall than the non-cheating runs at roughly the same level of precision. The result is that their F1 scores are slightly higher. This indicates that using future evidence in calculating corpus statistics does make a difference in retrieval.

2. Between retrieving 1,000 or 10,000 documents in the initial set to be scored, there is virtually no difference in effectiveness. The only measurable difference is that recall when $K = 10$ drops 0.01 when $N$ increases to 10,000.

From these results we conclude that it is indeed necessary to specifically avoid using corpus statistics from the current day, but that among all documents retrieved during the present day, whether we score 1,000 or 10,000 makes little difference. This is likely because the proportion of vital documents is so low that if we have not found them in the top $K$ in a set of 1,000 we are unlikely to find them in the top $K$ in a set of 10,000.
Figure 4.2: Recall and precision comparison of two system differing only for step 2 of system process flow.
<table>
<thead>
<tr>
<th>Run Name</th>
<th>Precision K = 10</th>
<th>Precision K = 30</th>
<th>Recall K = 10</th>
<th>Recall K = 30</th>
<th>F1 K = 10</th>
<th>F1 K = 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM-st</td>
<td>0.21</td>
<td>0.21</td>
<td>0.46</td>
<td>0.54</td>
<td>0.29</td>
<td>0.31</td>
</tr>
<tr>
<td>PM-JenSmooth-st</td>
<td>0.19</td>
<td>0.20</td>
<td>0.53</td>
<td>0.61</td>
<td>0.28</td>
<td>0.30</td>
</tr>
<tr>
<td>LM-mu2500-st</td>
<td>0.20</td>
<td>0.21</td>
<td>0.53</td>
<td>0.62</td>
<td>0.29</td>
<td>0.31</td>
</tr>
<tr>
<td>LM-mu5000-st</td>
<td>0.20</td>
<td>0.20</td>
<td>0.53</td>
<td>0.62</td>
<td>0.29</td>
<td>0.31</td>
</tr>
<tr>
<td>Ind-cheat</td>
<td>0.20</td>
<td>0.22</td>
<td>0.57</td>
<td>0.69</td>
<td>0.30</td>
<td>0.33</td>
</tr>
<tr>
<td>Ind-cheat-Fb</td>
<td>0.20</td>
<td>0.21</td>
<td>0.61</td>
<td>0.72</td>
<td>0.31</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Table 4.7: Results for various models on short query. Phrase “st” in run names for short label query. Indri cheating run results are shown for comparison.

4.5.3 Effect of Query Type

4.5.3.1 Short Queries

First we used the Wikipedia topic labels as a query. The results of various models for different values of $K$ are shown in Table 4.7. (Note that we do not conduct KL-Divergence experiments here. With short queries, the KL-Divergence model reduces to the language model.) The performance of Poisson Modeling, Language Modeling, and BM25 are all roughly comparable to each other, and lower than the performance of the cheating runs. BM25 performs very slightly better on precision then other filtering system runs, but its recall is lower. Overall, this translates into very little difference in F1 scores.

It has been observed that these models perform similarly for many IR tasks, despite their differing theoretical backgrounds. This confirms those observations.

4.5.3.2 Long Queries

Next we used abstracts derived from Wikipedia pages as queries. Note that the abstracts were taken from past versions of the page to ensure that the abstract was not using any information that the system should not have access to. Table 4.8 shows results for four retrieval models. Unlike previous results, here we see substantial differences between the recall performance of different models. BM25 easily achieves the highest recall, and since it achieves slightly higher precision than other models as
<table>
<thead>
<tr>
<th>Run Name</th>
<th>Precision</th>
<th></th>
<th></th>
<th>Recall</th>
<th></th>
<th></th>
<th>F1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BM-lg</td>
<td>0.21</td>
<td>0.22</td>
<td>0.65</td>
<td>0.80</td>
<td>0.32</td>
<td>0.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM-JenSmooth-lg</td>
<td>0.20</td>
<td>0.21</td>
<td>0.56</td>
<td>0.70</td>
<td>0.30</td>
<td>0.33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KL-mu2500-lg</td>
<td>0.22</td>
<td>0.21</td>
<td>0.38</td>
<td>0.45</td>
<td>0.29</td>
<td>0.29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LM-mu2500-lg</td>
<td>0.18</td>
<td>0.19</td>
<td>0.34</td>
<td>0.46</td>
<td>0.24</td>
<td>0.27</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.8: Results for various models using long abstract query. 93 topics had abstract text. Results here are averaged over 93 topics.

well, it obtains the highest F1 scores. The Poisson modeling approach had the second-highest recall and therefore second-highest F1. The KL-Divergence model, which differs from the basic language model in how it models the query, is more effective than that model.

Figure 4.3 shows a comparison of BM25 precision and recall using long and abstract queries for each of the 93 topics with a long query. While precision does not vary much over topics, recall varies a great deal; while the recall achieved with a long query is usually higher than the recall achieved with a short query, this is not always the case.

Contrast this with the same comparison for language models in Figure 4.4. The correlation of precision between long and short queries is not as strong as with BM25; some queries achieve much higher precision, though most achieve lower. For recall, too, most queries achieve much lower recall with long queries than with short.

We actually did not expect BM25 to perform well on long queries due to the previously-mentioned observed empirical similarity between BM25 and LM. Nevertheless, it was clearly the best model for long queries. We did expect KL-divergence to perform better because of long queries, since with a long well-formed query, the estimated query model will be better (Section 2.1.4). While KL-Divergence did better than Language Model, it did not out-perform BM25 or the Poisson Model.

The following is an example for abstract query used for topic [http://en.wikipedia.org/wiki/Frank_Winters](http://en.wikipedia.org/wiki/Frank_Winters) for which all of our models had zero recall and precision.
Figure 4.3: Comparison of BM25 Runs for short label queries and long abstract queries. X axes is indexes for 93 Wikipedia Topics
Figure 4.4: Comparison of Language Model Runs for short label queries and long abstract queries. X axes is indexes for 93 Wikipedia Topics. Red color
Frank Mitchell Winters (born January 23, 1964 in Hoboken, New Jersey) is a former American football center in the National Football League for the Cleveland Browns, New York Giants, Kansas City Chiefs, and the Green Bay Packers. He played college football at Western Illinois University and was drafted in the tenth round of the 1987 NFL Draft. Winters was the Packers’ starting center serving for eight straight seasons (1993-2000). He played in the Pro Bowl and also earned USA Today All-Pro honors in 1999. His nickname was “Frankie Baggadonuts” or “Old Bag of Donuts”. On July 18, 2008, Winters was inducted into the Green Bay Packers Hall of Fame. His ceremony was marked by heightened media interest because quarterback Brett Favre gave the induction speech amidst the developing saga regarding Favre’s status the Packers. On May 20, 2009, Winters got an internship with the Indianapolis Colts. He has part ownership in a popular Missouri bar and grill, Frankie &amp; Johnny’s.

The above mentioned topic had only 7 vital documents in the full corpus. Many of the topics with low recall and precision had few vital documents. Following is an example of abstract of topic [http://en.wikipedia.org/wiki/Hjemkomst_Center](http://en.wikipedia.org/wiki/Hjemkomst_Center) with high precision and recall. It had 189 vital documents.

The Heritage Hjemkomst Interpretive Center, commonly known as the Hjemkomst Center, is an interpretation center museum in Moorhead, Minnesota. The building opened in 1985 and serves as a home to Hjemkomst Viking Ship, Hopperstad Stave Church replica, quarterly museum exhibits, and county archives. It is also occupied by the Historical and Cultural Society of Clay County. The Red River Valley exhibit is a permanent display of the geologic and cultural history of the valley and its settlers.

Finally, we compared effectiveness of models between short and long queries. Table 4.9 shows results. Interestingly Language Model based systems with long queries hurt system performance compared to LM with short queries; LM was the only model for which that happened. It is not clear why this is the case.

4.5.3.3 Effect of Vital Document Counts

In Table 4.2, we showed the counts of topics with various numbers of vital documents in the TREC KBA data. It may also be interesting to see if there is any empirical performance difference due to these counts. Figure 4.5 compares the average
### Table 4.9: Result comparison for 93 topics for short and long queries. “lg” for long abstract query and “st” for short query.

<table>
<thead>
<tr>
<th>Run Name</th>
<th>Precision</th>
<th></th>
<th>Recall</th>
<th></th>
<th>F1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Precision</td>
<td></td>
<td>Recall</td>
<td></td>
<td>F1</td>
</tr>
<tr>
<td></td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td>$K = 10$</td>
<td>$K = 30$</td>
</tr>
<tr>
<td>BM-lg</td>
<td>0.21</td>
<td>0.22</td>
<td>0.65</td>
<td>0.80</td>
<td>0.32</td>
<td>0.35</td>
</tr>
<tr>
<td>BM-st</td>
<td>0.21</td>
<td>0.21</td>
<td>0.46</td>
<td>0.54</td>
<td>0.29</td>
<td>0.31</td>
</tr>
<tr>
<td>PM-JenSmooth-lg</td>
<td>0.20</td>
<td>0.21</td>
<td>0.56</td>
<td>0.70</td>
<td>0.30</td>
<td>0.33</td>
</tr>
<tr>
<td>PM-JenSmooth-st</td>
<td>0.19</td>
<td>0.20</td>
<td>0.53</td>
<td>0.61</td>
<td>0.28</td>
<td>0.30</td>
</tr>
<tr>
<td>LM-mu2500-lg</td>
<td>0.18</td>
<td>0.19</td>
<td>0.34</td>
<td>0.46</td>
<td>0.24</td>
<td>0.27</td>
</tr>
<tr>
<td>LM-mu2500-st</td>
<td>0.20</td>
<td>0.21</td>
<td>0.53</td>
<td>0.62</td>
<td>0.29</td>
<td>0.31</td>
</tr>
</tbody>
</table>

recall and precision of our models with vital document counts in the three ranges in Table 4.2: 1–20, 21–100, and more than 100. Systems generally have lower recall and higher precision on topics with a large number of vital documents and vice versa.

### 4.5.4 Effect of History Size

All of the previous experiments used $X = 5$ days of history to fetch corpus statistics. Table 4.10 shows a comparison of the BM25 and KL-Divergence models for $X = 1$, $X = 5$, and $X = 10$ for both short and long queries. We observed differences in performance due to history size on BM25 based runs using long queries to filter vital documents. The performance of BM25 system with one day of history data is lower as compared to BM25 with five and ten days of history. We did not observe any difference between BM25 runs for shorter queries. The performance of KL-Divergence with 10 days of accumulated data was slightly better than with 5 days; recall improved some (though that did not translate into an improvement in F1). More experiments are required to study the effect of history. The initial results in Table 4.10 are interesting.

### 4.5.5 Comparison Against KBA teams

The work in this thesis is similar to the work that others have done towards the TREC KBA track described in Section 2.4. Table 4.11 compares our two systems “BM25-lg” (BM25 with long abstract queries) and “PM-JenSmooth-st” (Poisson modeling with short queries) against other good reported results from the TREC 2013
Figure 4.5: Precision and recall comparison of models on three groups of topics. Topics are grouped into three categories according to the number of vital documents in collection—1 to 20, 21 to 100, and greater than 100. Long queries were used for retrieval.
Table 4.10: Comparison of run on 1, 5 and 10 days of history indexes and on short and large queries. “lg” is for abstract queries and ’st’ for short queries (topic labels).

<table>
<thead>
<tr>
<th>Run Name</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K = 10$</td>
<td>$K = 30$</td>
<td>$K = 10$</td>
</tr>
<tr>
<td>BM-st ($X = 5$)</td>
<td>0.21</td>
<td>0.65</td>
<td>0.32</td>
</tr>
<tr>
<td>BM-st ($X = 10$)</td>
<td>0.19</td>
<td>0.53</td>
<td>0.28</td>
</tr>
<tr>
<td>BM-lg ($X = 1$)</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
</tr>
<tr>
<td>BM-lg ($X = 5$)</td>
<td>0.22</td>
<td>0.54</td>
<td>0.31</td>
</tr>
<tr>
<td>KL-mu2500-lg ($X = 5$)</td>
<td>0.22</td>
<td>0.48</td>
<td>0.29</td>
</tr>
<tr>
<td>KL-mu2500-lg ($X = 10$)</td>
<td>0.21</td>
<td>0.49</td>
<td>0.29</td>
</tr>
</tbody>
</table>

Table 4.11: Result comparison of BM25 and Poisson Model based runs against other research groups system TREC KBA 2013.
KBA track. The works of the research teams mentioned in Table 4.11 are described in more detail in Section 2.4.2. Reported precision, recall and F1 score are averaged across Wikipedia topics. As evident from Table 4.11, our two systems “BM25-lg” and “PM-JenSmooth-st” match or exceed the effectiveness of other research team’s systems.

These systems all have different approaches. All of them do some preprocessing filtering to reduce the corpus size. The system proposed by University of Illinois [42] (system uiucGSLIS-bayes02 in Table 4.11) is based on statistical information retrieval and dynamic updating of the topic profile (Section 2.4.2). The University of Delaware system udel_fang-UDInfoK_Wiki1 (Table 4.11) [41] was based on finding related topics of the main topic and assigning appropriate weights to them. The score of a document is the sum of weighted topics found in document (Section 2.4.2). The system proposed by Beijing Institute of Technology [39] (system BIT-RFClassStrict in Table 4.11) is based on multi-step processing of a document (Section 2.4.2).

One important difference between our system and other systems mentioned in table is the strict real-time decision on documents about relevancy to topic. Our system pools documents for the present processing day and returns the top $K$ documents, whereas other systems in Table 4.11 make decision after processing each document. In some sense this could mean that our systems are actually using less information than others, since whether we are scoring the first document arriving on a day or the last, we are using the same information (the $X$ days of history prior to that), whereas other groups would be using all documents that arrived during the day to score the final document on that day.
Chapter 5

CONCLUSION

We introduced a framework for filtering a very large streaming corpus for vital documents and performed experiments suggesting that it equals or outperforms other such systems on the TREC KBA data. We performed a variety of experiments—short query, long query, accumulated index size, and retrieval set size. Among the models we tested, we expected BM25 and Poisson model to not do well on long queries, but surprisingly the results were good; in particular, using BM25 with abstract queries resulted in very high recall. On the other had, we were expecting good performance from KL-divergence because of its better estimation of the query model for long queries; although its performance was not bad, it was lower than other systems apart from the language model. Interestingly, the Poisson model and the Language Model are based on same principle of query-likelihood [34, 44], so since the Poisson model outperformed the Language Model, we believe that the Poisson Model provides a better distribution to model query generation from documents. Our experiments confirmed the effectiveness of statistical IR methods for filtering task.

The implemented retrieval models were able to filter vital documents reasonably well, but there remain many experiments that could be performed. The decision about the vitality of document to a topic was not in real time; in practice those decisions could be used to update the profile and potentially make better decisions in the future. (Although it may also be the case that strict real-time decision-making might not be a requirement for a practical system used by generic user.) One simple approach for real-time decision-making could be the use of the score of the top ranked document for a topic previously observed to filter documents which are above that score. This score needs to be updated regularly after observing new set of documents. We did not use
training data or the past judgment of the documents for a topic, but we believe this can definitely improve the results. The work of Robertson et. al on threshold setting and optimization [49] is very interesting and relevant. All of the challenges described in their work remain relevant to our task.

In addition, we definitely feel the need for evaluation methodologies specifically suited for real-time data and filtering. Our systems reached a performance peak for F1, but we feel that the evaluation measures we used (which were official measures for the TREC KBA track) did not reflect the real difference between systems. A good evaluation measure should take into account the burstiness of the topic. A topic may be alive only for certain period of time, with many vital documents in that period, while it may have very few vital documents in other periods of its life cycle. We should calculate the evaluation measurements based on the life cycle of the topic. For example, we could compute two sets of measurements—one over time periods when the topic has many vital documents and another when it does not. Another possible evaluation measure is the ratio of relevant to non-relevant documents being proposed by the system on each day of the corpus.

Information filtering remains an interesting and challenging problem even after more than 40 years of research. The work in this thesis can be expanded on many fronts. We did not use supervised machine learning techniques. It will be very interesting to combine signals from machine learning and statistical IR. Recent trends in commercial search engines have proved that combining different signals and feedback improve system performance. In this work, we have studied the challenges of statistical IR techniques. Applying machine learning technique for filtering can be another interesting problem. The problem of thresholding, profile modification (query changes), over fitting, types of learning algorithm are interesting research projects. We have already mentioned about the need of right evaluation measures. We did not use meta data associated with documents in our model. Using simple meta data such as title and anchor text can be important feedback to the system. We also did not update the user query after observing new information. There is lot of scope in information
filtering projects and we touched few challenges and methods.
BIBLIOGRAPHY


