

***Retrieval using Document Structure and Annotations***

Paul Ogilvie

CMU-LTI-10-012

Language Technologies Institute  
School of Computer Science  
Carnegie Mellon University  
5000 Forbes Ave., Pittsburgh, PA 15213  
[www.lti.cs.cmu.edu](http://www.lti.cs.cmu.edu)

**Thesis Committee:**

Jamie Callan (chair)  
Christos Faloutsos  
Yiming Yang  
W. Bruce Croft (University of Massachusetts at Amherst)

*Submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy  
In Language and Information Technologies*

© 2010, Paul Ogilvie

# *Abstract*

---

Successful retrieval of information from text collections requires effective use of the information present in a collection. The structure of documents in the collection and the relationships between elements within a document and other documents contain important information about the meaning of these elements. For example, the words present in the title of a web page may contain important clues about that page’s content. The text of a link to the web page may also be an important indicator of the page’s content.

Researchers have long recognized that structure can be an important indicator of relevance. Yet the majority of prior work is limited to experiments on small test collections and evaluated on a single retrieval task. These limitations hamper the generality of the conclusions. The recent construction of large and diverse test collections provides us the opportunity to reconsider the general task of retrieval in collections with structure.

This dissertation draws on three retrieval tasks to identify important properties of retrieval systems supporting the use of structure and annotations. We investigate known-item finding of web pages, retrieving elements from XML articles, and the retrieval of answer-bearing sentences as a component of a question-answering system. The retrieval model, an adaptation of the Inference Network model, clarifies the query language and simplifies the process of smoothing using multiple representations. The experiments in this dissertation show state-of-the-art results for these tasks and also provide novel insights to the shape of the parameter space when using mixtures of language models. Our experiments with question-answering further show how semantic predicates automatically annotated on a collection can be used to improve a system’s ability to retrieve answer-bearing sentences.

# *Acknowledgments*

---

My time as a graduate student has spanned many years so it is likely that my fallible memory will not do justice to all who have shaped my work.

I should first recognize James Allan, for without his mentorship while I was an undergraduate I would not have pursued a doctoral degree. My committee chair, Professor Jamie Callan, has of course had immense impact on my work; we never had a shortage of ideas we wished to explore. Professor W. Bruce Croft, my external committee member, has helped my work to be practical and useful for a wider audience than it would have been otherwise. I also thank Professors Yiming Yang and Christos Faloutsos; their presence on my committee has helped me to keep my work rigorous and practical.

My conversations with Benjamin van Durme regarding semantic retrieval led to my experiments with Matthew Bilotti. Without their collaboration the scope of my dissertation would have been much narrower. Professor Jaap Kamps and the other researchers across the pond using language models have naturally influenced my work; Jaap has additionally provided me with helpful advice and perspective on several occasions. Over the years I have had affecting conversations with Yi Zhang, Victor Lavrenko, ChengXiang Zhai, Luo Si, Kevyn Collins-Thompson, Don Metzler, Trevor Strohman, Le Zhao, and Jonathan Elsas.

I owe extra thanks to Christine Mahady, who provided me with emotional support and motivation during a difficult few years of working full-time at a startup and part-time on my dissertation. My parents have also been a great pillar of support.

# Table of Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Known-Item Finding . . . . .	1
1.2	Retrieving Document Parts . . . . .	4
1.3	Retrieval of Linguistic Structure . . . . .	5
1.4	Properties of Structured Retrieval Models . . . . .	7
1.5	Dissertation Overview . . . . .	8
1.6	Outline . . . . .	10
<b>2</b>	<b>Document Structure</b>	<b>11</b>
2.1	Representing Structure as a Graph . . . . .	16
2.1.1	Document Structure . . . . .	17
2.1.2	Multiple Representations . . . . .	17
2.1.3	Annotations . . . . .	19
2.2	Summary . . . . .	21
<b>3</b>	<b>Related Work</b>	<b>23</b>
3.1	Exact Match Boolean Retrieval . . . . .	23
3.2	Vector Space Model . . . . .	25
3.2.1	Applications to Structured Retrieval . . . . .	26
3.3	Okapi . . . . .	29
3.3.1	Applications to Structured Document Retrieval . . . . .	30
3.4	Statistical Language Models . . . . .	33
3.4.1	Ranking Models . . . . .	34
3.4.2	Model Estimation and Smoothing . . . . .	35
3.4.3	Multiple Bernoulli Model . . . . .	36
3.4.4	Applications to Structured Document Retrieval . . . . .	37
3.5	The Inference Network Model . . . . .	41

3.5.1	Query Operators . . . . .	43
3.5.2	Applications to Structured Document Retrieval . . . . .	47
3.6	Other Approaches . . . . .	51
3.7	Summary . . . . .	52
<b>4</b>	<b>Task-Oriented Retrieval Models</b>	<b>54</b>
4.1	Language Models for Known-Item Finding . . . . .	54
4.1.1	Evaluation Methodology . . . . .	56
4.1.2	Results . . . . .	57
4.2	Language Models and Inference Networks for Element Retrieval . . . . .	58
4.2.1	Evaluation Methodology . . . . .	60
4.2.2	Grid Search for Parameter Estimation . . . . .	61
4.2.3	Results . . . . .	64
4.3	Inference Networks for Annotation Retrieval . . . . .	72
4.3.1	Experimental Methodology . . . . .	74
4.3.2	Results . . . . .	76
4.4	Summary . . . . .	83
<b>5</b>	<b>An Extended Inference Network Model</b>	<b>85</b>
5.1	Structure-Aware . . . . .	86
5.1.1	Motivation . . . . .	86
5.1.2	The Representation Layer . . . . .	89
5.1.3	Creating Representation Nodes . . . . .	91
5.1.4	Estimating Model Nodes . . . . .	92
5.1.5	Observations . . . . .	95
5.2	Structure-Expressive . . . . .	98
5.2.1	Motivation . . . . .	98
5.2.2	Scope Operator . . . . .	99
5.2.3	Observations . . . . .	101
5.3	Result-Universal . . . . .	102
5.3.1	Motivation . . . . .	103
5.3.2	Prior Probabilities Belong on Scope Operations . . . . .	103
5.4	Annotation-Robust . . . . .	105
5.4.1	Annotation Boundary Errors . . . . .	106

---

5.4.2	Annotator Confidence Estimates . . . . .	108
5.5	Summary . . . . .	108
<b>6</b>	<b>Experimental Results</b>	<b>111</b>
6.1	Known-Item Finding . . . . .	111
6.2	Element Retrieval . . . . .	116
6.2.1	Results . . . . .	118
6.2.2	Discussion . . . . .	127
6.2.3	Summary . . . . .	135
6.3	Annotation Retrieval . . . . .	136
6.3.1	Better Representations . . . . .	136
6.3.2	Probabilistic Observations . . . . .	141
6.3.3	Summary . . . . .	144
6.4	Chapter Summary . . . . .	145
<b>7</b>	<b>Conclusions</b>	<b>147</b>
7.1	Summary . . . . .	147
7.2	Contributions . . . . .	149
7.3	Future Work . . . . .	151
<b>A</b>	<b>NEXI query language</b>	<b>163</b>
<b>B</b>	<b>IEEE Elements Retrieved</b>	<b>165</b>

# Introduction

---

It is generally understood that structure in both collections of documents and queries should be beneficial for a wide range of retrieval tasks. Yet the field of Information Retrieval has created a much larger body of research focusing on supporting unstructured keyword queries and document retrieval methods that do not make much use of the structure. This focus is in large part a result of the test collections available. While many have document structure, it has been difficult for researchers to find significant benefit from the structure when investigating traditional document retrieval tasks.

Research in the last ten years has challenged earlier findings, resulting from a better understanding of a wider range of users' information needs and test collections developed with those information needs in mind. This dissertation focuses on three tasks where document structure is important. These are the retrieval of known documents from a collection (known-item finding), ranking the relevance of document parts (element retrieval), and the retrieval of sentences that may contain the answer to a question.

To illustrate these tasks, we consider a few information needs for an example news article. Figure 1.1 shows this article, which has internal structure, such as the title and paragraphs. It also has metadata: the author, publication date, and a link to an audio version of the report. There may also be alternative representations of content external to the document. For example, news publishers often provide summaries of recent articles in RSS (really simple syndication) feeds. Figure 1.2 show a sample of an RSS feed containing an item describing the news article. The news article is described by the fourth item in the feed. This feed item contains the title of the article, a link to it, and a short description of the content in the article.


## 1.1 Known-Item Finding


When users search the Web, they are often searching for documents they know exist. Broder [6] estimates that these navigational searches may be 20%-25% of all Web search queries. Some example information needs are the search for automobile manufacturer Nissan's homepage or the instructions for filling out a tax form.

Typical navigational Web search systems do not assume expert users with knowledge of a complex query language or the structure of documents in the collection. The

## UN Report Says Suicide Bombings on Rise in Afghanistan

By Daniel Schearf  
Islamabad  
09 September 2007

[Schearf report \(mp3\) - Download 536k](#) 

[Listen to Schearf report \(mp3\)](#) 

**A United Nations report says the number of suicide bombings in Afghanistan is rising fast and would this year likely reach a record high. As Daniel Schearf reports for VOA from Islamabad, the report says most Afghani suicide bombers were trained, supported, and many recruited, in neighboring Pakistan.**



**Investigators check the scene after a suicide attack in Kabul, 31 Aug 2007**

The U.N. report issued Sunday says the number of suicide bombings by insurgents in Afghanistan has rocketed from 17 attacks in 2005 to 123 in 2006.

The report also says that record high number is likely to be surpassed this year. In the first eight months there have already been 103 suicide attacks, with more expected.

The report says most suicide bombers were Afghani nationals but received training or support in neighboring Pakistan's tribal region where many were recruited from Islamic theology schools called madrassas.

Figure 1.1: A news article posted on the Internet.



## **VOA News: War and Conflict**

---

Up to the minute news from Voice of America

### **[ETA Vows to Continue Bombings in Spain](#)**

ETA claims responsibility for a small explosion on July 25

### **[Democrats Question Petraeus Report on Iraq](#)**

Senator Feinstein says she does not think the report's author is an 'independent evaluator' of the Iraq situation

### **[Israeli Police Arrest Neo-Nazi Gang](#)**

Police say a group of neo-Nazis is responsible for a wave of attacks on Orthodox Jews, synagogues and gays

### **[UN Report Says Suicide Bombings on Rise in Afghanistan](#)**

Suicide attacks reached 123 in 2006 and 103 in first eight months of 2007

### **[Palestinians Observe Strike Called by Fatah in Hamas-Ruled Gaza](#)**

Fatah-dominated PLO called strike after Hamas militiamen used clubs against Fatah activists who held outdoor prayer rallies in Gaza on Friday

Figure 1.2: An RSS news feed on the Internet that contains a description of the article in Figure 1.1.

queries are short and only contain query terms, as in the queries `nissan` or `irs 1040 instructions`. If a user has read the example news article and would like to find it again, they may remember that the report was published by the VOA (Voice of America) and about suicide bombings in Afghanistan. From this memory, a reasonable query would be

```
1.1    VOA suicide bombings Afghanistan.
```

Despite the unstructured nature of the query, document structure is still important for effective ranking of documents for known-item searches. In the news article, most of the query terms are contained in the article's title, and it could be an effective technique to place more emphasis on terms in the title when estimating the relevance of documents. The alternative representation found in the RSS feed may also be helpful. `VOA` occurs in the title of the RSS feed, and the query term `suicide` also occurs in the text of the article's image caption.

## 1.2 Retrieving Document Parts

In many cases, it may be preferable to rank parts, or elements, of documents. An element corresponds directly to explicitly encoded structure in documents, such as the text of a title, paragraph, or section. If users know this structure, they may wish to create complex queries that better match their information needs. For example, many articles published by Voice of America have audio segments. Rather than retrieving the whole article, a query such as

```
1.2    RETURN audio OF (article MATCH suicide bombings)
```

specifies a user's request that only the audio of articles that match the query terms `suicide bombings` be returned.

However, the right unit of retrieval may vary by document or information need. For example, a user with a general information need on the subject of suicide bombings and may submit the query

```
1.3    suicide bombings
```

to the search engine. As the example news article is entirely on the subject of suicide bombings, it would be appropriate to return the entire news article. On the other hand, if the user's information need is more specific and the user is only interested in text about where suicide bombers are trained, they may use the query

```
1.4    training suicide bombers.
```

In this case, it may be more appropriate for the retrieval system to return only the fourth paragraph in the document. Automatically identifying the relevant elements can be more important in collections with long documents, such as journal articles or books.

Regardless of whether the user has precise expectations about the structure of relevant documents or is ambiguous about which elements should be retrievable, the document structure may be useful during estimation of relevance. Estimating whether an element is relevant may be particularly difficult when the element contains very little text. This small sample problem is more pronounced in element retrieval than in document retrieval; in many cases a system may have to rank elements containing only a few words.

As with known-item finding, knowledge of where keywords occur within the structure of an article may benefit ranking. With Query 1.3 it may be helpful for the system to use the information that both query terms occur in the title of the article. This may be good evidence that the system should return the entire article, and not just a single element. When ranking elements for Query 1.4, the system may also use the evidence that `suicide` and `bombers` occurs throughout the article when estimating the relevance of the paragraphs. A paragraph which contains all three keywords in an article about training counter-terrorism forces may not have as frequent use of the terms `suicide` and `bombers` in the article, providing evidence to the retrieval system that the paragraph about training suicide bombers is more likely to be relevant than the entire article.

### 1.3 Retrieval of Linguistic Structure

Finally, a retrieval system may also have access to annotations of the text in the news article. For example, the text of the article may be annotated with PropBank style semantic role labels:

```
1.5    [ARG1 most Afghani suicide bombers] were [TARGET trained],
        supported, and many recruited, [ARGM-LOC in neighboring
        Pakistan].
```

This bracket notation for annotations is typical of many linguistic annotation tools. The verb `trained` is labeled as the *target* of the semantic predicate. The meanings of the ARG clauses depend on the target verb. In this case, the ARG1 clause corresponds to the *recipient* of the training, and the ARGM-LOC clause indicates the location of the training.

The third motivating application directly studied in this dissertation is the retrieval component of a question answering system. Modern question answering systems use a large text corpus to extract or compose answers for natural language questions, such as

```
1.6    Where are suicide bombers trained?
```

The typical process such a system performs consists of several steps.

The first step is question processing. In question processing, the system analyzes the question to extract keywords and identify candidate answer types and also possibly answer templates. For the question above, the question answering system might identify

that a semantic predicate with a target verb `trained` that has an ARG1 clause containing `suicide bombers` might contain the answer to the question in an ARGM-LOC clause.

Given the question analysis, a question answering system will then formulate queries to search for passages that may contain the answer to the question. A question answering system using a keyword-only information retrieval system may submit the query

```
1.7    suicide bombers trained
```

to the search engine. This query is not a very precise representation of the question answering system's information need. For example, a retrieval system ranking sentences could return both of the following sentences with a high estimate of relevance:

```
1.8    Al Qaeda has trained many suicide bombers.
```

```
1.9    Hamas trained many of the suicide bombers used in Israel.
```

The first sentence can be easily ruled out as a candidate answer because it does not have a location. Many retrieval systems provide support this directly in their query languages. However, a retrieval system that simply supports the presence of locations in the retrieved passages will still retrieve the second sentence.

If the retrieval system supports the indexing and querying of semantic annotations, then the question answering system could issue the query

```
1.10   RETURN target (MATCH trained) AND (suicide bombers IN
        ./arg1) AND (HAS ./argm-loc) .
```

Here we assume that predicates are indexed by their target verb with the ARG clauses as direct "children" of the target verb. The query requests that predicates be ranked by how well the target verb matches the keyword `trained`, a child ARG1 clause (with child-hood indicated by `./`) of the target match `suicide bombers`, and that the target have a child that is of the type ARGM-LOC.

Sentence 1.9 may be annotated with the semantic predicates

```
1.11   [ARG0 Hamas] [TARGET trained] [ARG1 many of the suicide
        bombers used in Israel]
```

```
1.12   Hamas trained many of [ARG1 the suicide bombers] [TARGET
        used] [ARGM-LOC in Israel]
```

Neither semantic predicate for Sentence 1.9 match the constraint in the query that the ARGM-LOC clause be a child of the target verb matching `trained`. However, the example predicate in Annotation 1.5 that answers the question does match the query.

In addition to the in-depth example provided above, there may be other important challenges to retrieval systems. For example, should the results only contain elements that exactly match all constraints, or should the result list contain partial matches, with

those matching the query better placed in positions near the top of the result list? As the linguistic annotations are produced by tools prone to error, the search system should be robust to these errors, returning the best matches first and gracefully degrading with less exact matches further down the result list. Furthermore, annotation tools may provide confidence values and alternative annotations. The search system should be capable of incorporating confidence estimates and alternative annotations during indexing and retrieval. Finally, the search system should have a query language and retrieval model expressive enough to handle information needs of these forms.

## 1.4 Properties of Structured Retrieval Models

The previous sections provided concrete examples of when the use of structure may be important. Each example highlighted some properties we believe are helpful for effective retrieval in these environments. This section draws on those observations and states some properties we believe are necessary for effective retrieval of structured documents.

**result-universal:** The model can return as results any document or any other structural element.

Many of the traditional retrieval approaches have focused on document retrieval. However, as published information has become more structured, it has become apparent that users' information needs may be satisfied not only by documents but also by elements of the documents in a collection. Evidence of this need to rank parts of documents can be seen in product search on the Web (a single document may have many products listed), retrieval of elements, book search, and retrieval of passages for question answering systems.

**structure-aware:** The model leverages query term occurrences found in related documents, representations, elements, or annotations.

When ranking structural elements or annotations, the system may be faced with very small samples of text. We hypothesize that in order to achieve good estimates of relevance, it will be important to place some weight on term occurrences from related elements, such as the containing document or alternative representations. We also hypothesize that it may be beneficial to place extra emphasis on elements of certain types (such as titles) contained within the element being ranked. We feel that a strong retrieval model for information retrieval on structured information sources will be able to make use of these related elements when estimating relevance.

**structure-expressive:** The model supports a rich query language so that users can express constraints on related elements.

It should not be surprising that with more structured information comes a greater need for structured representations of users' information needs. Consider the complexity of queries in databases, where very little of the information is unstructured. We expect that it may be difficult for a non-expert user to formulate effective structured queries. However, there is a strong tradition within Information Retrieval of automatic query-reformulation which could be leveraged to enrich a query with more structural constraints. Additionally, there may be many retrieval environments where the users may be expert searchers. Librarians and users of legal information systems are frequently capable of forming very complex but effective queries.

**annotation-robust:** The model is robust to errors in annotations.

The automatic creation of annotations on text is an error-prone process and will always be so due to the inherent ambiguity of natural language. We believe that the most effective retrieval systems leveraging annotations will not treat the annotations as "ground truth" but instead the result of a process that may make errors. Annotators may make many kinds of errors: missing/extra annotations, incorrect boundaries or labels, and incorrect links between annotations.

## 1.5 Dissertation Overview

This dissertation is unique because it examines a wider range of retrieval tasks making use of document structure and annotations. We first consider the tasks of known-item finding and the retrieval of document parts in isolation, presenting experiments on these tasks with retrieval models developed specifically for them (Chapter 4). While these retrieval models are effective, they suffer the same flaw as the bulk of prior work. Development in isolation of other retrieval tasks limits their generality.

Prior work, reviewed in Chapter 3, focused on developing retrieval approaches supporting some of the properties in Section 1.4, but not all. To this date, no comprehensive work has been done examining a unified retrieval approach supporting this a wide range of retrieval tasks. This dissertation corrects this, proposing adaptations of the Inference Network model that supporting the properties. This dissertation also investigates similarities and differences of successful model configurations across the three retrieval tasks presented in this chapter.

The retrieval model proposed in this dissertation uses a graph-based representation of the structure to improve system's ability to estimate the subject of the text in an element, annotation, or representation, making the retrieval model *structure-aware*. It does this by using properties of the graph structure to identify related text to improve the estimation of which topics are discussed. For example, a word in the title element of a news article

may be more indicative of the subject of the document than other words in the document. The text of an alternative description in RSS of the article may also serve as a good representation of the article's content.

The estimates of content formed using the approach described above can be used within a probability framework known as the *Inference Network model* for retrieval. An inference network is constructed from the query and document collection to estimate whether a user's information need is satisfied by the structural elements in the collection. Queries written in an expressive query language specify how the probabilities should be estimated in the inference network. The queries can be complex, allowing the use of Boolean operations: one can say that all query terms or constraints should be matched (AND), one of several terms or constraints should be matched (OR), or a term or constraint should not be matched (NOT). Users can also express phrasal constraints (e.g. the phrase `Voice of America` should occur) and proximity constraints (e.g. `suicide` and `bombings` should appear within ten words of each other).

In Chapter 5 we extend the retrieval model's query language to make it more *structure-expressive* and *result-universal*. For example, we may wish to retrieve articles with an image caption containing `Kabul`. It supports the information needs of both human users and other applications that use retrieval systems. While doing so, the retrieval model uses established statistical modeling techniques to formally justify the rankings presented to the users.

The probabilistic Inference Network model has several important motivating properties which encourage its application to structured document retrieval tasks. Prior work with Inference Network models has demonstrated that it may be easily adapted to model metadata or structured documents. Inference networks can be used to model complex information needs, providing a way to probabilistically model relevance for users (human or computer) through expressions of information needs more sophisticated than those expressed in simple keyword queries. By using statistical estimation techniques, we may be better able to estimate for which queries the elements and annotations in a collection are relevant. The statistical estimation techniques may also be easily adapted to robustly handle uncertainty and errors in annotations in a principled manner, making the model *annotation-robust*.

Chapter 6 demonstrates the effectiveness of the modeling and ranking approaches presented in this dissertation on the tasks of element retrieval, ranking Web documents for navigational search, and the retrieval of passages for question answering systems. The analyses in these chapters show the strength and flexibility of the approach presented in this dissertation. These chapters also provide insight into the individual tasks examined, analyzing assumptions about related elements on these collections and the techniques necessary for effective retrieval performance.

## 1.6 Outline

The rest of the dissertation is organized as follows. Chapter 2 more explicitly describes the retrieval problem and some motivating retrieval tasks. Chapter 3 presents the related work on structured retrieval and provides some important background material. Chapter 4 introduces task-oriented models developed separately for known-item finding, element retrieval, and the retrieval of answer-bearing sentences for question answering systems. Drawing from observations regarding the limitations of the task-oriented models, Chapter 5 proposes modifications to the Inference Network model. Evaluation of the approach for the tasks of navigational Web search, element retrieval, and answer-bearing sentence retrieval are presented in Chapter 6. Chapter 7 concludes the dissertation by summarizing the contributions and their significance, and presents open problems and potential research applications.



# Document Structure

---

A retrieval system may make use of two forms of document structure: *annotations* and *structural markup*. This chapter defines structural markup and annotations; they will be important for our discussion for retrieval models in Chapters 4 and 5.

Annotations typically encode linguistic or semantic information and document markup is commonly used to represent general layout of documents. The difference between markup and annotations can be viewed as the source of the structure. We define our usage of the phrase *structural markup* and the term *annotations* below.

**structural markup:** explicitly encoded information about the content of the document present within the document. Structural markup is created as a part of the document authoring and editorial process. The structure may encode information about layout, as in title tags or paragraph tags, or citations and hyperlinks.

**annotations:** information about the document assigned by a process or human after the original creation of the document, such as information provided by a named-entity tagger or an automatic layout analysis tool. This information may be less certain than structural markup, as the process or person creating the annotations may not know the author's original intent. Annotations are usually added for linguistic or semantic information such as part-of speech tags, parse trees of sentences, named-entities, reading difficulty, and language.

So that the system may know how to render the presentation in Figure 1.1, the structure in the document must be explicitly represented; Figure 2.1 shows how this structure can be encoded using the structural markup language of XML. Just as the news article can be represented in XML, so can the RSS feed shown in Figure 1.2. Figure 2.2 shows a sample of the XML in the RSS feed.

Table 2.1 reviews some key concepts related to XML and structural markup. Each metadata and structural *field* or *element* is encoded in the document by surrounding it in a begin and an end *tag*. For example, the `title` field of the report is identified in the document by surrounding the title text with `<title>` and `<\title>`. The article element is the *parent* of each `paragraph` element. Each of the `paragraph` elements is a *child* of the `article` element.

```
<article>
  <front_matter>
    <title>
      UN Report Says Suicide Bombings on Rise in Afghanistan
    </title>
    <author> Daniel Schearf </author>
    <location> Islamabad </location>
    <date> 09 September 2007 </date>
    <audio>
      <link> http://www.voanews.com/medi.....Bombers-Mp2.Mp3 </link>
      <caption> Schearf report </caption>
    </audio>
  </front_matter>
  <intro_paragraph>
    A United Nations report says the number of suicide bombings in
    Afghanistan is rising fast and would this year likely reach a
    record high. As Daniel Schearf reports for VOA from Islamabad,
    the report says most Afghani suicide bombers were trained,
    supported, and many recruited, in neighboring Pakistan.
  </intro_paragraph>
  <image>
    <link> http://www.voanews.com/englis.....g07_eng_195.jpg </link>
    <caption>
      Investigators check the scene after a suicide attack in Kabul,
      31 Aug 2007
    </caption>
  </image>
  <paragraph>
    The U.N. report issued Sunday says the number of suicide
    bombings by insurgents in Afghanistan has rocketed from 17
    attacks in 2005 to 123 in 2006.
  </paragraph>
  <paragraph>
    The report also says that record high number is likely to be
    surpassed this year. In the first eight months there have
    already been 103 suicide attacks, with more expected.
  </paragraph>
  <paragraph>
    The report says most suicide bombers were Afghani nationals but
    received training or support in neighboring Pakistan's tribal
    region where many were recruited from Islamic theology schools
    called madrassas.
  </paragraph>
</article>
```

Figure 2.1: The news article encoded in XML.

```
<rss>
  <channel>
    <title> VOA News: War and Conflict </title>
    <link> http://www.voanews.com/ </link>
    <description>
      Up to the minute news from Voice of America
    </description>

    .....

  <item>
    <title> Israeli Police Arrest Neo-Nazi Gang </title>
    <link> http://www.voanews.com/english/20070909-voa13.cfm </link>
    <description>
      Police say a group of neo-Nazis is responsible for a wave of
      attacks on Orthodox Jews, synagogues and gays
    </description>
  </item>
  <item>
    <title>
      UN Report Says Suicide Bombings on Rise in Afghanistan
    </title>
    <link> http://www.voanews.com/english/20070909-voa16.cfm </link>
    <description>
      Suicide attacks reached 123 in 2006 and 103 in first eight
      months of 2007
    </description>
  </item>
  <item>
    <title>
      Palestinians Observe Strike Called by Fatah in Gaza
    </title>
    <link> http://www.voanews.com/english/20070909-voa18.cfm </link>
    <description>
      Fatah-dominated PLO called strike after Hamas militiamen used
      clubs against Fatah activists who held outdoor prayer rallies
      in Gaza on Friday
    </description>
  </item>
</channel>
</rss>
```

Figure 2.2: A sample of the RSS news feed in XML.

Concept	Definition
<i>markup</i>	in-line specification of structure
<i>field, element</i>	a part of a document
<i>tag</i>	marks the beginning or end of an element
<i>link</i>	a reference to another element or document
<i>parent</i>	the element which directly contains another
<i>child</i>	an element directly contained by another
<i>ancestor</i>	an element that is a parent of the element or is an ancestor of the element's parent
<i>descendant</i>	an element that is a child of the element or is a descendant of one of the element's children

Table 2.1: Key concepts in structural markup.

There are many formats for the specification of document structure other than XML. Web pages are typically encoded in HTML, many documents were encoded using SGML before the introduction of XML, and word processors such as Microsoft Word have their own formats for encoding the structure of a document. However, the concepts in Table 2.1 are generally applicable other forms of structural markup.

The Introduction showed one example annotation. However, the semantic predicates may overlap in complex ways. For example, the text in Annotation 1.5 also contains the predicates:

- 2.1 [ARG1 most Afghani suicide bombers] were trained, [TARGET supported], and many recruited, [ARGM-LOC in neighboring Pakistan].
- 2.2 [ARG1 most Afghani suicide bombers] were trained, supported, and many [TARGET recruited], [ARGM-LOC in neighboring Pakistan].

There is one predicate for each of the target verbs `trained`, `supported`, and `recruited`. This example shows that the structure of information in language is often more complex than the simple hierarchical nesting that in-line annotation such as the square-brace notation above or the markup of XML within the document can represent.

A natural solution to encoding linguistic annotations is to use offset, or standoff, annotation. In offset annotation, an index of structure separate from the original text of the document contains cross-references to the original text. The cross-references are called offsets, and may be in terms of byte locations or token (word) positions. To keep our example

simple, we will use token positions and assume the original text has been tokenized in a uniform manner for all processing components. Below are the token identifiers for a sample of the news article:

...	(57) were	(62) recruited
(53) most	(58) trained	(63) in
(54) Afghani	(59) supported	(64) neighboring
(55) suicide	(60) and	(65) Pakistan
(56) bombers	(61) many	...

The three overlapping predicates may be represented in offset annotation in a separate file:

```
<semantic-roles>
  <target begin='58' length='1'>
    <arg1 begin='53' length='4' />
    <argm-loc begin='63' length='3' />
  </target>
  <target begin='59' length='1'>
    <arg1 begin='53' length='4' />
    <argm-loc begin='63' length='3' />
  </target>
  <target begin='62' length='1'>
    <arg1 begin='53' length='4' />
    <argm-loc begin='63' length='3' />
  </target>
</semantic-roles>
```

Each annotation has a label such as `target`, a `begin` *attribute* which indicates the token where the annotation begins, and a `length` attribute which specifies the number of tokens contained within the annotation. As every semantic predicate has a single target verb, we have chosen to make the `target` annotations the parents of the other arguments in the predicate. This enables us to encode the relationship between the arguments and target verb of each predicate in a simple way.

The concepts of *parent*, *child*, *ancestor*, and *descendant* are defined by the hierarchical relationships present in the offset annotation file and not on the source text. With inline encoding of structure, the text of the child must be contained within the text of its parent. However, with offset annotation it is possible to express hierarchical relationships among annotation elements where the parent text does not overlap with the text of its children. For example, the text of the parent target `trained` does not contain the text of its child `most Afghani suicide bombers`.

Offset annotations allow for natural representation of the different annotations within the text of a document. The approach is not limited to storing a single type of annotation or structure. For example, a system may also identify named-entities, such as people or locations. A named-entity recognizer run on the tokens labeled with locations above might identify that `Afghani` is a nationality and that `Pakistan` is a country. The offset annotation description of the document structure can be extended to encode this information by including:

```
<named-entities>
  <nationality begin='54' length='1' />
  <country begin='65' length='1' />
</named-entities>
```

The news article and the related RSS feed concretely demonstrate encoding of structural markup, alternative representations, and linguistic annotations. The news article and the feed items are not artificial; they are actual examples found on the Internet. While the actual format of the presentation of structure may differ, these forms of document structure are not uncommon. We believe it is essential that modern information retrieval systems must effectively represent and make use of this structure to achieve the best possible retrieval quality. In order to discuss how a system may use the structure, Section 2.1 presents how the structure, annotations, and representations can be modeled using a graph.

## 2.1 Representing Structure as a Graph

One can represent the relationships between elements and annotations in a document collection using a graph. A graph is defined by a set of vertexes and edges between vertexes. The vertexes correspond to the structural elements, which may be typed, such as `title`, `paragraph`, and so on. Formally, given alphabets  $\Sigma_V$  and  $\Sigma_E$  specifying the possible labels on vertexes and edges, a graph is defined as:

$$G = (V, E, \ell_V)$$

where  $V$  is a finite set of vertexes (or nodes),  $E$  is a set of labeled directed edges between vertexes denoted by pairs of the form  $(v_i, l, v_j)$  such that  $v_i, v_j \in V$ ,  $l \in \Sigma_E$  and  $\ell_V$  is a function of the form  $\ell : V \rightarrow \Sigma_V$  specifying a labeling of the vertexes. The rest of the section describes how document structure, multiple representations, and annotations may all be represented.

### 2.1.1 Document Structure

For the simple document structure formed by in-line tagging, the links in the graph are used to represent the parent/child relationship explicitly coded in the text. To represent structural markup one needs only a single edge type  $p$  (for parent) in  $\Sigma_E$  to represent documents with internal structural markup.

The structure of the news article can be represented visually as in Figure 2.3 or more formally using the graph notation:

$$\begin{aligned} V_s &= \{v_1, v_2, \dots, v_{16}\} \\ E_s &= \{(v_1, p, v_2), (v_1, p, v_{10}), (v_1, p, v_{11}), (v_1, p, v_{14}), \\ &\quad (v_1, p, v_{15}), (v_1, p, v_{16}), (v_2, p, v_3), (v_2, p, v_4), \\ &\quad (v_2, p, v_5), (v_2, p, v_6), (v_2, p, v_7), (v_7, p, v_8), \\ &\quad (v_7, p, v_9), (v_{11}, p, v_{12}), (v_{11}, p, v_{13})\} \end{aligned}$$

and

$$\begin{array}{lll} \ell_V(v_1) = \text{article} & \ell_V(v_7) = \text{audio} & \ell_V(v_{12}) = \text{link} \\ \ell_V(v_2) = \text{front\_matter} & \ell_V(v_8) = \text{link} & \ell_V(v_{13}) = \text{caption} \\ \ell_V(v_3) = \text{title} & \ell_V(v_9) = \text{caption} & \ell_V(v_{14}) = \text{paragraph} \\ \ell_V(v_4) = \text{author} & \ell_V(v_{10}) = \text{intro\_paragraph} & \ell_V(v_{15}) = \text{paragraph} \\ \ell_V(v_5) = \text{location} & \ell_V(v_{11}) = \text{image} & \ell_V(v_{16}) = \text{paragraph} \\ \ell_V(v_6) = \text{date} & & \end{array}$$

This graph-based representation depicts the structural relationships between the elements in the news article. The vertexes and their labels correspond directly to the elements and element types in Figure 2.1 in the order of the begin tags within the text. The parent/child relationships present in the XML markup are explicitly represented in the edges between vertexes. For example, the edge  $(v_1, p, v_2)$  exists in  $V$  because the `article` element is the parent of the `front_matter` element.

### 2.1.2 Multiple Representations

A simple way to extend the graph to capture representational relationships is to introduce new edge types that provide connections across representations. Each structural element may have multiple representations, and we can connect these representations through the introduction of an additional edge type  $r$  to indicate that a vertex corresponds to an alternative representation.

Returning to our news article, we can represent the structure of the RSS feed and the alternative representation relationship by adding vertexes  $\{v_{17}, \dots, v_{41}\}$  and the following

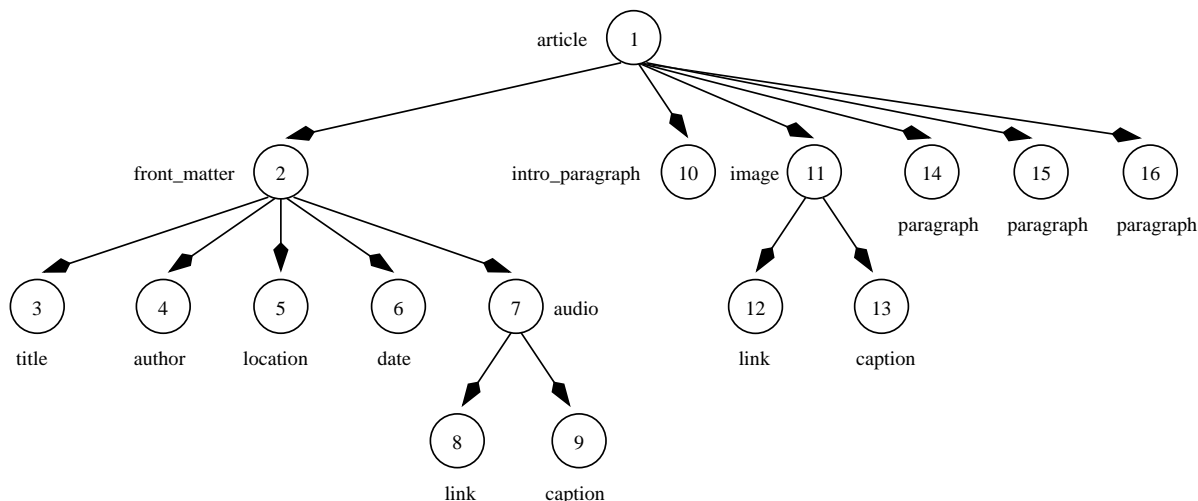


Figure 2.3: The graph representing document structure for the example news article.

edges to the graph:

$$\begin{aligned}
 V_r &= V_s \cup \{v_{17}, v_{18}, \dots, v_{41}\} \\
 E_r &= E_s \cup \{(v_{17}, p, v_{18}), (v_{18}, p, v_{19}), (v_{18}, p, v_{20}), (v_{18}, p, v_{21}), \\
 &\quad (v_{18}, p, v_{22}), (v_{22}, p, v_{23}), (v_{22}, p, v_{24}), (v_{22}, p, v_{25}), \\
 &\quad (v_{18}, p, v_{26}), (v_{26}, p, v_{27}), (v_{26}, p, v_{28}), (v_{26}, p, v_{29}), \\
 &\quad (v_{18}, p, v_{30}), (v_{30}, p, v_{31}), (v_{30}, p, v_{32}), (v_{30}, p, v_{33}), \\
 &\quad (v_{18}, p, v_{34}), (v_{34}, p, v_{35}), (v_{34}, p, v_{36}), (v_{34}, p, v_{37}), \\
 &\quad (v_{18}, p, v_{38}), (v_{38}, p, v_{39}), (v_{38}, p, v_{40}), (v_{38}, p, v_{41}), \\
 &\quad (v_1, r, v_{34})\}
 \end{aligned}$$

and

$$\begin{array}{lll}
 l_V(v_{17}) = \text{rss} & l_V(v_{26}) = \text{item} & l_V(v_{34}) = \text{item} \\
 l_V(v_{18}) = \text{channel} & l_V(v_{27}) = \text{title} & l_V(v_{35}) = \text{title} \\
 l_V(v_{19}) = \text{title} & l_V(v_{28}) = \text{link} & l_V(v_{36}) = \text{link} \\
 l_V(v_{20}) = \text{link} & l_V(v_{29}) = \text{description} & l_V(v_{37}) = \text{description} \\
 l_V(v_{21}) = \text{description} & l_V(v_{30}) = \text{item} & l_V(v_{38}) = \text{item} \\
 l_V(v_{22}) = \text{item} & l_V(v_{31}) = \text{title} & l_V(v_{39}) = \text{title} \\
 l_V(v_{23}) = \text{title} & l_V(v_{32}) = \text{link} & l_V(v_{40}) = \text{link} \\
 l_V(v_{24}) = \text{link} & l_V(v_{33}) = \text{description} & l_V(v_{41}) = \text{description} \\
 l_V(v_{25}) = \text{description} & & 
 \end{array}$$

The final edge,  $(v_1, r, v_{34})$ , connects the fourth RSS feed item to the article. This edge indicates that the feed item is another representation of the article. Figure 2.4 shows how



the graph may be drawn.

We choose to introduce the new link type rather than adding an additional node to the graph grouping representations in order to preserve the property that each graph node is anchored in a textual element present in the collection. In Chapter 5 we will show how the graph structure can be “queried” to group related elements together during model estimation. Furthermore, the introduction of the additional link type  $r$  allows us to distinguish between links within a document and links across a document. These link types will be used when grouping related elements in Chapter 5.

### 2.1.3 Annotations

The introduction to this chapter includes several examples of linguistic annotations on the text. Recall that the relationships between annotations and their locations in text are typically stored external to the original text, such as in offset annotation. These externally defined annotations explicitly contain the parent-child relationships between annotations, which may not correspond to the textual containment relationships common to structural markup. However, the relationships between annotations often have a similar graph-like nature, making it is easy to encode these annotations within the same graph framework. This section will not explicitly list the graph notation to represent the hierarchical relationships between annotations; the approach is the same as for inline structural markup.

The following example demonstrates how annotations may have parent-child relationships, even though the text for an annotation may contain the text of its descendant annotations. Recall the overlapping semantic predicates:

- 1.5 [ARG1 most Afghani suicide bombers] were [TARGET trained], supported, and many recruited, [ARGM-LOC in neighboring Pakistan].
- 2.1 [ARG1 most Afghani suicide bombers] were trained, [TARGET supported], and many recruited, [ARGM-LOC in neighboring Pakistan].
- 2.2 [ARG1 most Afghani suicide bombers] were trained, supported, and many [TARGET recruited], [ARGM-LOC in neighboring Pakistan].

These three separate annotations can be represented in a graph in the context of the text in Figure 2.5. This figure shows how the annotations may be cleanly represented within the collection graph structure, even though the text of the argument annotations are not contained within the text of their parent `target` annotation.

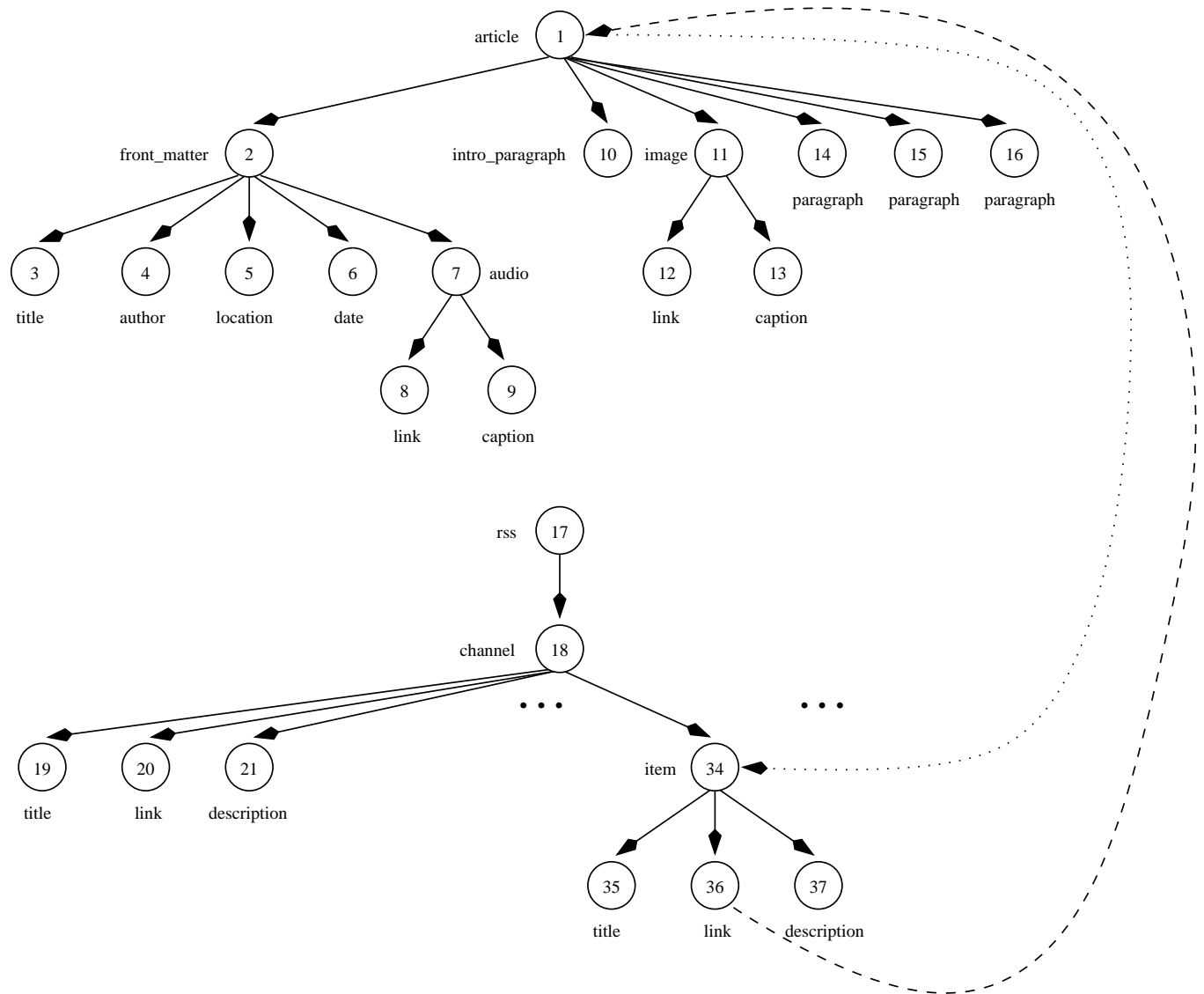


Figure 2.4: The graph representing the structure of the news article and part of the RSS feed structure. The dashed arrow from  $v_{36}$  to  $v_1$  represents the explicit link from the RSS item to the news article. The dotted arrow from  $v_1$  to  $v_{34}$  represents our view that the `item` element of  $v_{34}$  is an alternative representation of the `article` element.

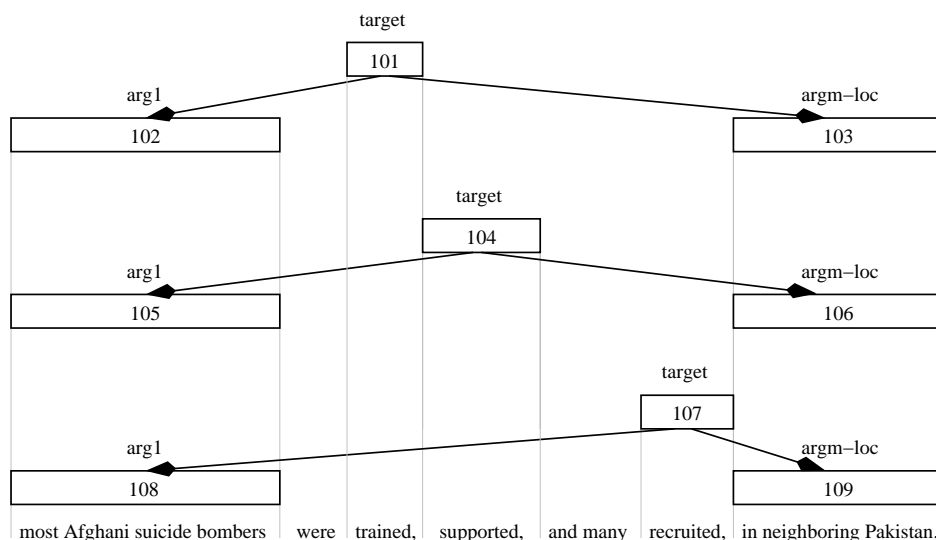


Figure 2.5: The relationship between keywords and the annotations can be clearly conveyed by drawing the graph in the context of the text. The vertexes for the annotations are drawn with square boxes above the terms contained within each annotation.

## 2.2 Summary

This chapter discussed encoding document structure, multiple representations, and annotations. Section 2.1 presented a way to represent the structural relationships between document components, representations and annotations in a graph.

There are distinct similarities in structural markup and annotations; they can both be represented using a graph. The vertexes in a graph represent the elements and annotations. Aggregated across all documents in a collection, which we refer as the *collection graph structure*. Yet there are differences between explicit structural markup and annotations. Structural markup is created during authorship and is typically quite hierarchical in nature, with elements containing the text of descendant elements. Annotations of text are performed post authorship and may have complex relationships between annotations which do not correspond to nesting of text. Annotations may be imprecise and imperfect, which can have impact on their use in retrieval tasks.

For a retrieval system to be *result-universal*, it must be able to retrieve and provide reasonable rankings for any structural element or annotation in the collection. As this chapter shows, some annotations and elements may contain very small amounts of text, making meaningful ranking difficult. Thus a system must be able to make use of related and surrounding elements and annotations to provide good estimates of relevance, highlighting the need for a system to be *structure-aware*.

Just as a system may need to make use of containment and the collection graph structure to do a good job of representing the elements and annotations in a collection, any query language supporting retrieval of structured collections must be able to express the relationships between elements, annotations, and keywords. Without this support, a retrieval system cannot fully be *structure-expressive*.

Finally, the imperfection of annotations calls for systems to be *annotation-robust*. A system supporting the use of annotations should be able to make effective use of annotations in its result-universality, structure-awareness, and structure-expressivity, but also be robust to the errors introduced during annotation.

The formalization of structural markup and annotations is also important because it gives us a framework for the discussion of related work in Chapter 3. Chapter 4 provides more detailed examples of models adapted to use structure and annotations in retrieval for some retrieval tasks, presenting experiments and discussing results. However, few of the techniques discussed in Chapters 3 and 4 address the similarities and differences between structural markup and annotations. The extended Inference Network model presented in Chapter 5 draws extensively on the definitions in this chapter to provide a more complete and direct integration of structural markup and annotations.

# Related Work

---

Given the large volume of prior work on structured document retrieval problems, it is no mean task to provide a thorough and concise summary. Nevertheless, this is precisely what this chapter aims to do. Much of the research focuses on the adaptation of the popular *vector space*, *probabilistic*, *inference network*, and *statistical language model* retrieval models for information retrieval. This chapter has sections covering the research for each of these models. The discussions throughout the chapter also note which of the five properties presented in Section 1.4 are addressed in the related work. In addition to the detailed text summary, Figure 3.1 shows a timeline of major research on structured document retrieval.

One of the common themes of adapting existing retrieval models to support structured document retrieval is what we will call the *small document* model. In the small document model, elements or alternative document representations are treated in the same way as documents within the retrieval model. Using this approach, one can directly apply existing ranking models to the document elements. This chapter uses the notion of the small document model throughout to simplify the discussion of related work. It is also common to layer score combination on top of the small document model to combine evidence from multiple representations or elements to produce a single score for an element, which this chapter also notes.

This chapter first presents examples of the earliest research of structured document retrieval. Sections 3.2 through 3.5 present each of the highly adapted retrieval models and their applications to structured document retrieval. Section 3.6 reviews other techniques. Section 3.7 summarizes the related work while reviewing the properties desirable for structured retrieval.

## 3.1 Exact Match Boolean Retrieval

Researchers have long recognized that document structure is important to effective retrieval. Some of the earliest systems supporting structure focused on providing support for querying of the fielded information. This research had strong connections to the library sciences, where fielded search of citations is important, such as those in the MACHine-Readable Cataloging (MARC) format. A common approach to handling fielded queries was to treat constraints literally, perhaps providing a ranking of the query cor-

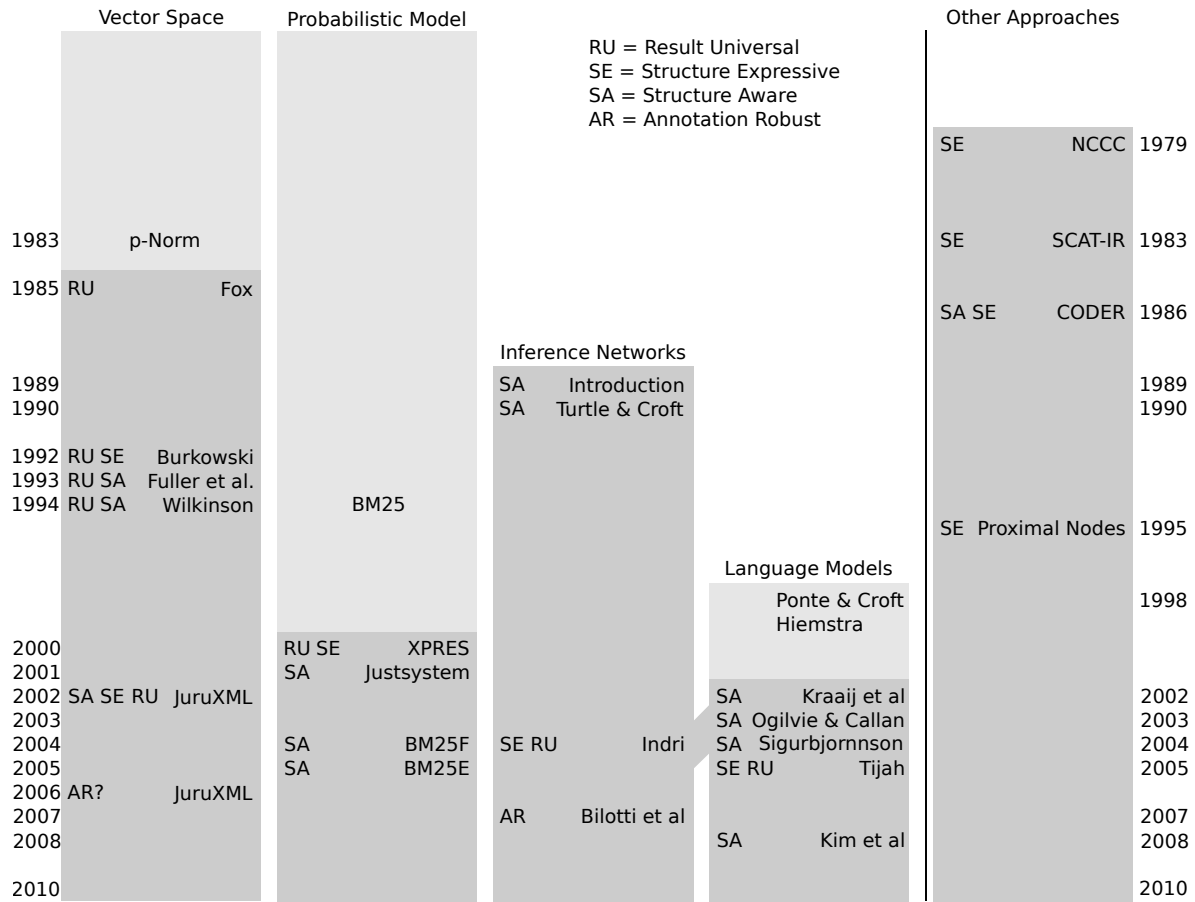


Figure 3.1: A timeline of research related to structured document retrieval. Dark shading indicates research in structured document retrieval, while light shading shows prior work in the model.

responding to how well the multiple constraints are met. Two examples are the Norton Cotton Cancer Center (NCCC) On-Line Personal Bibliographic Retrieval System and the SCAT-IR citation search system. Both systems provided a *structure-expressive* and Boolean query language, although exact match Boolean retrieval was the result; issues of ranking were not considered.

## 3.2 Vector Space Model

In the vector space retrieval model [74, 76, 69], documents and queries are represented as vectors. The dimensions in the vector could be any feature of the document, but they are typically words in the vocabulary. The length, or weight, on each dimension is often a variant of  $tf \cdot idf$ . Term frequency ( $tf$ ) is a function of the number of occurrences of the term in the represented text. It captures the intuition that frequent terms in a document are more representative of the document's content. Inverse document frequency ( $idf$ ) is a function of the frequency of the term in the collection and represents the intuition that occurrences of common words are less informative than occurrences of rare words. In addition, vectors may have a normalization component, which helps control document length biases.

A summary of vector components commonly used in the vector space model for structured document retrieval is in Table 3.1. For example, the notation  $lnc.ltc$  indicates that the query vector components  $lnc = I(w \in q)(1 + \log tf(w, q)) \cdot 1 \cdot 1$  has  $q_w = (1 + \log tf(w, q))$  for terms in the query and 0 otherwise. The document vector components are

$$d_w = \frac{(1 + \log tf(w, d)) \cdot \log \frac{N}{df(w)}}{\sqrt{\sum_{i=1}^{|W|} ((1 + \log tf(w, d)) \cdot \log \frac{N}{df(w)})^2}} \quad (3.1)$$

for terms in the document and 0 otherwise. For structured document retrieval, a few variants components  $N$  and  $df(w)$ . Exceptions to using global, document-based  $idf$  computation will be noted in the descriptions of systems below. This version of the vector space model is representative of applications of the vector space model to the retrieval of documents with structure in that it does not make use of pivoted length normalization (Singhal et al [81]).

Documents and queries may be compared by using the dot product of the two vectors:

$$q \cdot d = \sum_{w=1}^{|W|} q_w d_w \quad (3.2)$$

where  $|W|$  is the number of words in the vocabulary. The closer the vectors representing the query and document are to each other, the greater  $q \cdot d$ .

Term frequency component		Normalization component	
$b$	$I(w \in x)$	$n$	1
$n$	$\text{tf}(w, x)$	$c$	$1 / \sqrt{\sum_{i=1}^{ W } x_i^2}$
$l$	$I(w \in x)(1 + \log \text{tf}(w, x))$	$l$	$1/ x $
Term importance component		$L$	$1/\log( x )$
$n$	1		
$t$	$\log \frac{N}{\text{df}(w)}$		

Table 3.1: Commonly used components of the vector space model in structured document retrieval.

### 3.2.1 Applications to Structured Retrieval

**Fox** Fox performed some of the earliest research in applying the vector space model to structured retrieval using a small document approach [19]. Fox compared paragraph retrieval to section retrieval; these early experiments were *result-universal* in the sense that a fixed non-document unit of retrieval could be specified in advance, but the research did not consider rankings of mixed element types.

The  $p$ -norm model introduced by Salton et al. [75] is a generalization of exact match Boolean retrieval approaches to provide more flexible ranking of documents. The approach is *expressive* as queries can be combinations of ANDs and ORs of query terms, while providing ranked result lists of documents and also allowing the user to specify weights on query terms. The ranking model has an adjustable parameter which specifies the trade-off between matching the Boolean constraints and high similarity in the query clauses.

In the  $p$ -norm model, the query clauses can be nested, resulting in an expressive query language. Fox [19] suggests that this approach can be naturally extended for use in structured document collections, but stops short of proposing actual extensions to the model.

**Burkowski** Burkowski [7] presents a model for searching and retrieving components of a hierarchically structured text database using the small document and vector space models. The *structure-expressive* query language allows the user to specify the unit of retrieval and supports enforcement of components being contained within others, as well as providing support for standard Boolean operations. As any element could be retrieved, the model is *result-universal*. Burkowski shows particular foresight in supporting multiple hierarchical structures on the text. However, the structured queries in the model are



treated as exact match results, with ranking provided only on the results that satisfy an exact match query. Burkowski discarded the structure of the elements and documents for ranking, using *nnn.ltL* with standard global, document-based *idf*.

**Fuller et al** Fuller et al [26] present a *result-universal* model for retrieval of elements that combined the score of child elements with the score for the element itself. This *structure-aware* ranking formula is:

$$\text{RSV}(q, e) = \alpha_{t(e)}(q \cdot e) + \sum_{i=1}^{|c(e)|} \frac{q \cdot c_i}{\beta^{i-1}} \quad (3.3)$$

where  $c_i \in c(e)$  are the children of the element  $e$  sorted in order of descending  $q \cdot c_i$  and  $\alpha_{t(e)}$  is a weight for  $t(e)$ , the type of the element.  $\beta$  is a constant greater than one that reduces the bias toward elements with more children. Fuller et al experimented with both *btc.ntc* and *btn.btl* with standard global, document-based *idf* values.

We call the approach for creating a *structure-aware* retrieval model in Equation 3.3 *score combination*; the final ranking formula is a combination of the retrieval scores from each of the representations deemed important for estimating relevance. An advantage of *score combination* is that it is easy to compute; one need only use a pre-existing retrieval model to rank each of the representations and then combine the scores.

**Wilkinson** Wilkinson [91] performed the first systematic evaluation of many of the hypotheses previously posed about structured document retrieval. Wilkinson’s work focused on four general hypotheses: (1) document retrieval can be done just using retrieval of their elements, (2) document retrieval can be enhanced by using scores for documents and elements, (3) element retrieval can be done just using retrieval of documents, and (4) element retrieval can be enhanced by using both scores for documents and elements. In his experiments, Wilkinson used the small document and vector space models to rank documents. By using score combination Wilkinson was able to replicate the same effectiveness of document retrieval:

$$\text{RSV}(q, d) = \sum_{e_i \in d} \alpha_{t(e)} \frac{q \cdot e}{\beta^{i-1}} \quad (3.4)$$

where the section elements in the summation are sorted by decreasing  $q \cdot e$ . Wilkinson used  $\beta = 2$  and hand-tuned the  $\alpha$  weights on element types. For similarity computation in  $q \cdot e$ , Wilkinson used *nnc.ntc*. The *idf* values for documents were computed using document counts while the *idf* values for sections were computed where  $N$  equaled the number of sections in the corpus and  $\text{df}(w)$  counts were based on the number of sections containing the term.

Using a linear combination of Equation 3.4 normalized by the maximum  $RSV(q, d)$  and  $q \cdot d$  normalized similarly resulted in a slight boost to early precision, providing evidence that *structure-aware* retrieval algorithms may do better than standard document retrieval algorithms for some tasks.

These two experiments verify Wilkinson’s first two hypotheses. Ranking using only document scores did not perform well when compared to directly ranking elements, thus rejecting the Wilkinson’s third hypotheses. By linearly combining normalized scores of the element and its document, Wilkinson found that incorporating the document’s context during ranking improved retrieval effectiveness, confirming his fourth hypothesis.

Wilkinson’s work is important as it provides some of the earliest thorough experiments demonstrating that *structure-aware* systems can provide effective retrieval. This work is also the first using thorough assessments on elements, providing an early empirical evaluation of *result-universal* techniques.

**JuruXML** A more recent application of the vector space model to structured document retrieval is implemented in the JuruXML system [49, 46, 47, 8]. Carmel et al apply the vector space model to ranking XML elements against queries represented as *structure-expressive* XML fragments. For example, the query

```
3.1 <article> <title> bombings </title> </article>
```

expresses the information need that `article` elements with `title` elements about `bombings` should be returned. Note that a strict interpretation of this query would not return the example article in Chapter 2 as the `title` element in that article is not a child of the `article` element but rather a descendant. However, Carmel et al treat the XML fragment as a vague information need, where an exact match of the structure is preferred, but other approximate matches are also returned. They do so through the inclusion of  $cr(c_i, c_j)$ , a context resemblance function which models the user’s tolerance to imperfect matches. The context of the term is defined by its path from the root element of the document or query to the term. Carmel et al define a partial match context resemblance function as

$$cr(c_i, c_j) = \begin{cases} \frac{1+|c_i|}{1+|c_j|} & \text{iff } c_i \text{ is a subsequence of } c_j, \text{ and} \\ 0 & \text{otherwise.} \end{cases} \quad (3.5)$$

JuruXML applies this context resemblance function in a variant of cosine similarity:

$$RSV(q, d) = \frac{\sum_{(w, c_i) \in q} \sum_{(w, c_j) \in d} q_{w, c_i} d_{w, c_j} cr(c_i, c_j)}{u(q) \cdot u(d)} \quad (3.6)$$

where  $u(q)$ ,  $u(d)$  are the number of unique terms in the query and document, and  $d_w = \log(\text{tf}(w, d) + 1) \cdot \log(N/\text{df}(w))$  and  $q_w = \log(\text{tf}(w, q) + 1)$ . The retrieval model is *structure-aware* as terms from other contexts in the document may match a specific query term and context pair.

Mass and Mandelbrod [46] apply the small document model to ranking elements for ranking unstructured queries in JuruXML. For  $N$  and  $df(w)$  they used counts for the element type and the element frequency, rather than standard document frequency. They found it important to normalize the scores in Equation 3.6 by dividing the highest achievable score by hypothetical element of the same type, because the *idf* statistics and average element lengths can be quite different across the element types in a collection.

Carmel et al [8] consider the impact of the context resemblance function and explore variants of context sensitive *idf* which accumulates counts where  $cr > 0$ . For element retrieval with strict interpretation of the requested result type, they found that using the standard, global document-based *idf* and discarding structural constraints on query terms worked as well as other more complex approaches.

Mass and Mandelbrod [47] consider a linear score combination of an element's score with that of the document. This work also extends the XML fragment query language to support Boolean operations.

Chu-Carroll et al [11] later applied the XML fragments query language to retrieval of documents leveraging annotations for the support of a question answering system. Unfortunately, the work does not thoroughly investigate how the quality of the annotations affects the degree to which the model is *annotation-robust*.

Despite the strengths of the JuruXML system, we feel that the vector space model in JuruXML is not as *structure-expressive* as one might like. For example, when ranking articles against the query

```
3.2    <article> <paragraph> suicide bombings </paragraph>
        </article>
```

the retrieval model does not recognize the user's desire that the terms `suicide` and `bombings` occur in the same `paragraph` element; the ranking function in Equation 3.6 does not reward articles where both terms are present in the same paragraph.

### 3.3 Okapi

The Okapi retrieval model and related probabilistic retrieval models have been extensively applied to the retrieval of structured documents. These probabilistic models are based on the *probability ranking principle* [71], which asserts that the best ordering of results is by descending probability of usefulness to the user.

The applications of the probabilistic retrieval models presented in this section are mostly variants of Okapi, which is an approximation to the two Poisson model [28]. The two Poisson model directly models the probability ranking principle, but in practice has too many parameters to estimate. Robertson and Walker [72] approximate the two Pois-

son model with:

$$\text{bm25}(q, d) = \sum_{w \in W} \frac{(k_3 + 1) \cdot \text{tf}(w, q)}{k_3 + \text{tf}(w, q)} \frac{(k_1 + 1) \cdot \text{tf}(w, d)}{k_1 \left( (1 - b) + b \frac{|d|}{\text{avg}l} \right) + \text{tf}(w, d)} \log \frac{N - \text{df}(w) + 0.5}{\text{df}(w) + 0.5}. \quad (3.7)$$

where  $k_1$ ,  $k_3$ , and  $b$  are constants that must be tuned to a retrieval task and collection. While the applications of the probabilistic models may use a different approximation, Equation 3.7 is a representative example.

### 3.3.1 Applications to Structured Document Retrieval

While the applications to structured document retrieval presented below may use different approximations of the probability ranking principle, the general techniques remain the same. As with the vector space model, many of the techniques use the small document model and score combination.

**XPRES** Wolff, Flörke, and Cremers [92] present the XPRES system for retrieval of structured documents. It is a *result-universal* retrieval model, allowing any element in a structured document to be ranked. The query language is also *structure-expressive*; queries consist of a set of terms and contexts (contexts are called *roles* in their work). Elements are ranked using the small document model; when a more specific context is requested for a query term, the element's contribution to the retrieval status value for that query term is computed by forming a small document from the descendent elements that match the requested context. As with the JuruXML system, XPRES does not allow the user to express the desire that two query terms occur in the same element. Wolf et al use *role* specific counts for  $N$  and  $\text{df}(w)$ .

**Justsystem** The earliest adaptations of the probabilistic model to create *structure-aware* approaches performed *score-combination*. Fujita [25] of Justsystem combines the scores from several different representations to perform document retrieval of web pages for the task of homepage finding. The model uses a weighted linear combination of small document scores estimated on numerous document representations and query independent predictors of relevance. Example representations for web document retrieval include the title of the document and a small document formed by concatenating the text of all elements with large fonts. The choice of  $N$ ,  $\text{df}(w)$ , and average length for different representations was not specified in [25].

**Beitzel et al** Many others have used the *score combination* approach for *structure-aware* applications of the probabilistic model. For example, Beitzel et al [1] combine the Okapi

scores from title, in-link, and full text indexes using the combMNZ (combine, multiply by non-zero) result fusion algorithm for web document retrieval. CombMNZ [21] combines rankings of the top  $n$  results across several search engines (one for each representation type) by summing the scores observed in the ranked documents and multiplying the sum by the number of rankings where a representation of the document was returned. The formula can be written as

$$\text{combMNZ}(q, d) = \left( \sum_{r \in R} I(r_r(q, d) \leq n) \right) \cdot \left( \frac{1}{|R|} \sum_{r \in R} I(r_r(q, d) \leq n) \cdot s_r(q, d) \right) \quad (3.8)$$

where

$$\begin{aligned} R & \text{ is a set of rankings from search engines,} \\ s_r(q, d) & \text{ is the score of } d \text{ from the search engine } r, \\ r_r(q, d) & \text{ is the rank of } d \text{ for the search engine } r, \text{ and} \\ I & \text{ returns 1 when the argument is true and 0 otherwise.} \end{aligned} \quad (3.9)$$

Beitzel et al [1] apply the combMNZ fusion algorithm to rankings returned by scoring the different document representations of title, in-link text, and full document text. The scores across the different representations are often not very comparable. To improve the comparability of the scores, Beitzel et al use a variant of the ZNORM score normalization proposed by Montague and Aslam [57]:

$$s_z(q, d) = \frac{s(q, d) - \text{mean}(s(., q))}{\text{var}(s(., q))} \quad (3.10)$$

where  $\text{mean}(s(., q))$  and  $\text{var}(s(., q))$  are the mean and variance of the scores for query  $q$  over the top  $n$  results. Beitzel et al's modification is to apply the exponential function to the BM25 scores from each of the representations,  $s(q, d) = \exp(\text{bm25}(q, d))$ , before applying the ZNORM normalization. As they create a separate index for each of the representations, the average document length and inverse document frequency components are conditioned on the representation type. Lu et al [45] also apply combMNZ to the output of several Okapi rankings for Web search.

**Vittaut et al** In addition to applications of Okapi to web document retrieval, it has also been adapted to element retrieval. Vittaut et al [88] use the small document model for ranking elements. Vittaut and Gallinari [87] propose a *structure-aware* approach to ranking XML elements by using a weighted linear combination of small document scores for the element, its parent, and the element's document. The weights in the linear combination are estimated on training topics and judgments by minimizing exponential loss over ranking preferences on pairs of elements. Vittaut and Gallinari use global, document-based values for *idf* but use element type specific estimates for average length.

**BM25F** However, Robertson et al [73] openly criticize the *score combination* approach to adapting the probabilistic model to be *structure aware*. They list several problems with the approach. One concern is that score combination breaks the nonlinear aspects of the term frequency normalization present in the original Okapi BM25 algorithm. 2.1. Suppose we rank documents by

$$\sum_{i=1}^{|R|} \alpha_i \cdot \text{bm25}(q, r_i(d)) \quad (3.11)$$

where  $R$  is a list of document representations and  $r_i \in R$ . If we assume  $\text{df}_{r_i}(w) = \text{df}(w)$ ,  $\alpha_i = \alpha_j$  for all  $i, j$ , and  $|r_i(d)| = \text{avdl}_i$ , then a document's score  $a$  for single term query is equivalent to ranking by

$$\sum_{i=1}^{|R|} \frac{(k_1 + 1) \cdot \text{tf}(w, r_i(d))}{k_1 + \text{tf}(w, r_i(d))} \geq \sum_{i=1}^{|R|} \text{I}(\text{tf}(w, r_i(d)) > 0) \frac{(k_1 + 1) \cdot 1}{k_1 + 1} = \sum_{i=1}^{|R|} \text{I}(\text{tf}(w, r_i(d)) > 0). \quad (3.12)$$

This linear increase in score when matching additional representations is contrary to the spirit of Okapi. A linear increase in term matches in a document causes a non-linear increase in the  $\text{bm25}(q, d)$  score, and Robertson et al assert that this property should also hold for matches across multiple representations.

Robertson et al [73] propose to address this concern by reweighting the document vector input to Okapi BM25 through the use of a linear weighted combination of the document vectors from each of the fields or representations:

$$\text{tf}_R(w, d) = \sum_{i=1}^{|R|} \alpha_i \cdot \text{tf}(w, r_i(d)). \quad (3.13)$$

The document length and average document length components of BM25 are also updated to reflect the modified term weights:

$$\begin{aligned} |d|_R &= \sum_w \text{tf}_R(w, d) \quad \text{and} \\ \text{avdl}_R &= \frac{1}{N} \sum_d |d|_R. \end{aligned} \quad (3.14)$$

$\text{tf}_R$ ,  $|d|_R$ , and  $\text{avdl}_R$  are substituted in place of  $\text{tf}$ ,  $|d|$ , and  $\text{avdl}$  in Equation 3.7. Robertson et al use standard global, document-based values for  $N$  and  $\text{df}(w)$  for the computation of the *idf* component of Okapi. This approach is referred to in the literature as Okapi BM25F. We call the general approach of combining representations within the model *in-model combination*. In-model combination of representations preserves many of the properties of the original retrieval model, while still allowing a weighted combination of evidence from alternative representations or related elements.

**BM25E** Lu, Roberston, and MacFarlane [44] adapt Okapi BM25F for XML element retrieval. Their approach, Okapi BM25E, arises from the hypothesis that Okapi BM25F is naturally applicable to ranking elements in addition to documents. They further state the desire to allow for the text of other elements to be considered during estimation of the term occurrences for an element. The resulting *structure-aware* and *result-universal* Okapi BM25E is a simple element based modification of BM25F, where the documents in BM25F are substituted with elements in BM25E. For BM25F,  $avdl_R$  is defined over documents, while for BM25E, the average length is defined over all elements in the collection. However, Lu et al approximate the proposed average with the BM25F  $avdl_R$  presented in Equation 3.14, stating concerns about the efficiency of averaging the element length over all elements in the collection. As with BM25F, BM25E uses document-based values for  $N$  and  $df(w)$ .

### Summary

As can be seen by the above examples, the Okapi BM25 approximation of the two Poisson model for information retrieval is a popular choice for adaptation to structured retrieval tasks. It has been extensively applied to Web retrieval tasks and XML element retrieval. The recent introduction of Okapi BM25F allows researchers to flexibly rank elements and documents leveraging multiple representations of the text while still preserving the non-linear attributes of the original Okapi BM25 weighting function. We agree with Robertson et al [73] that the *in-model combination* of fields, representations, or elements is a more principled approach to adapting the probabilistic model than earlier work using *score combination*.

The major goals in research of adapting the probabilistic model for structured document retrieval have been to create *structure-aware* and *result-universal* systems. The adaptations of the probabilistic model have not extensively addressed *structure-expressiveness* and *annotation-robustness*.

## 3.4 Statistical Language Models

Statistical language modeling for information retrieval has recently gained popularity due to its ability to make use of common statistical estimation methods (Ponte and Croft [67] and Hiemstra [30]). In the language modeling approach, the text of the document is viewed as a sample from an unknown latent model of the information present in a document. Generally, the latent model of the information in the document is represented by a unigram language model. A unigram language model specifies a multinomial probability distribution over words in a vocabulary. We use the symbol  $\theta$  to represent a language model. The probability of observing a word  $w$  given the language model  $\theta$  is written as

$P(w|\theta)$ . For a multinomial probability distribution,  $\sum_{w \in W} P(w|\theta) = 1$ . These models are based on combinations of maximum likelihood or Bayesian estimators, leaving few parameters that must be tuned by the researcher. This section first reviews the approaches to ranking the latent document models then presents some options for the estimation of the latent models for documents and queries.

### 3.4.1 Ranking Models

The most common approaches to using language models for ranking in information retrieval have been the generative query-likelihood model [31, 90, 96], the model comparison approaches of Kullback-Leibler divergence [93], and the generative relevance models [41]. The most widely applied of these to structured document retrieval is the query-likelihood model.

The generative query-likelihood approach was first introduced by Ponte and Croft [67]. In Ponte and Croft's approach, queries are represented as a binary vector over the vocabulary. However, later work by Hiemstra [31], Westerveld et al. [90], and Zhai [96] treats queries as a list of words rather than a binary vector. This allows for placing more weight on frequently occurring query terms, as in Zhai [96]:

$$P(q|\theta_d) = \prod_{w \in q} P(w|\theta_d)^{\text{tf}(w,q)} \stackrel{\text{rank}}{\equiv} \sum_{w \in q} \text{tf}(w,q) \log P(w|\theta_d) \quad (3.15)$$

Documents more likely to have produced the query are ranked higher.

Lafferty and Zhai [39] discuss the role of relevance in language modeling, proposing that one rank by

$$\log P(Q = q|D = d, R = 1) + \log \frac{P(R = 1|D = d)}{P(R = 0|D = d)} \quad (3.16)$$

where  $Q$  and  $D$  represent the random variables for the query and document and  $R = 1$  indicates relevance. Lafferty and Zhai [39] assert that one may reasonably estimate

$$P(Q = q|R = 1, D = d) \approx P(q|\theta_d). \quad (3.17)$$

Kraaij et al [38] instead propose an alternative interpretation where rankings use

$$P(Q = q|R = 1, D = d)P(R = 1|D = d). \quad (3.18)$$

For many retrieval tasks, we do not have prior information about what documents may be relevant. In these cases it is common to assume that  $P(R = 1|D = d)$  is a constant that can be omitted during ranking, resulting in ranking solely by  $P(q|\theta_d)$ . Upon reflection of Equation 3.17, one can see that a good language model  $\theta_d$  is one which accurately represents the queries for which the document  $d$  is relevant.



### Prior Probability of Relevance

One advantage of using the query-likelihood model for ranking documents is that it is compatible with prior probabilities of relevance. That is, we may have a query-independent estimate of relevance for documents in a collection for a specific retrieval task. In order to make use of the characteristics of relevant documents, one can assume that  $P(R = 1|D = d)$  is not a constant.

Westerveld et al [90] used URL types to inform document priors for homepage search. The URL types corresponded to root pages, sub-root (a single directory deep), deeper directory pages, and URLs corresponding to other files.

Kamps et al [33] applied prior probabilities to XML element retrieval. The priors were assumed to be proportional to the length of the element raised to the power  $\beta$ :

$$P(R = 1|D = d) = c \cdot |d|^\beta \quad (3.19)$$

where  $c$  is an unknown constant that can be neglected during ranking. This allows the introduction of a bias toward retrieving longer elements, where larger  $\beta$  values bias the scores toward ranking longer elements near the top of the result list for a query.

### 3.4.2 Model Estimation and Smoothing

To estimate  $\theta_d$  for document retrieval in the statistical language modeling framework, it is common to linearly interpolate the language model estimated from the document with a collection model. The maximum-likelihood estimate (MLE) of the document model, assuming an underlying multinomial model, is given by

$$P(w | \text{MLE}(d)) = \text{tf}(w, d) / |d|, \quad (3.20)$$

where  $|d|$  is the length in terms of the observed document. Linear interpolation is a simple approach to combining estimates from different language models:

$$P(w | \theta) = \sum_{m=1}^M \lambda_m P(w | \theta_m), \quad (3.21)$$

where  $M$  is the number of language models we are combining, and  $\sum_{m=1}^M \lambda_m = 1$ . An equivalent shorthand to Equation 3.21 is

$$\theta = \sum_{m=1}^M \lambda_m \theta_m. \quad (3.22)$$

Method	$\lambda_d$	$\lambda_C$
Jelinek-Mercer	$1 - \lambda$	$\lambda$
Dirichlet priors	$\frac{ d }{ d  + \mu}$	$\frac{\mu}{ d  + \mu}$
Two-stage	$\frac{(1-\lambda) d }{ d  + \mu}$	$\frac{\lambda d  + \mu}{ d  + \mu}$

Table 3.2: Common interpolation parameters for smoothing a document with a collection model.

Applying Equation 3.22 to the maximum likelihood estimate of the document’s language model and use a collection language model, we may write

$$\hat{\theta}_d = \lambda_d \text{MLE}(d) + \lambda_C \hat{\theta}_C. \quad (3.23)$$

The most common choices for setting the  $\lambda_C$  and  $\lambda_d$  are outlined shown in Table 3.2. These include Jelinek-Mercer smoothing, smoothing using Dirichlet priors [95, 96, 62, 90], and two-stage smoothing [96]. The two-stage smoothing weights shown in the table reflect the choice to model the background query model using the same collection model used for documents.

### 3.4.3 Multiple Bernoulli Model

Metzler et al [53, 84] proposes an approach (Model B) which views a document as a collection of samples from a multiple Bernoulli distribution. The model assumes that the multiple Bernoulli distribution is sampled once for each word in the document; a sample is associated with a word  $w_i \in W$  in a vector  $r$  where  $r_i = 1$  and 0 at all other locations in the vector. As with the other language modeling approaches, the features  $r_j$  in the vector are assumed to be independent. This approach uses a Bayesian approach to the parameter estimation of the model  $\theta$ , imposing a multiple beta prior over the parameters in  $\theta$ . Doing so results in estimates where

$$P(r_i = 1|d, \alpha, \beta) = \int_{\theta} P(r_i = 1|\theta)P(\theta|D, \alpha, \beta) = \frac{\#(r_i = 1, d) + \alpha}{|d| + \alpha + \beta}. \quad (3.24)$$

In the above,  $\#(r_i = 1, d) = \text{tf}(w_i, d)$ , or the number of observation vectors  $r$  in  $d$  where  $r_i = 1$ .

Setting  $\alpha = \mu P(w_i|\theta_C)$ ,  $\beta = \mu(1 - P(w_i|\theta_C))$ , and  $\theta_C = \text{MLE}(C)$  results in a model that produces identical probability estimates to the multinomial model using the Dirichlet smoothing estimate. This approach formalizes a connection between the multiple-Bernoulli model and the multinomial model. This formulation is particularly important as it is used in modern applications of the Inference Network model, which is presented in Section 3.5.

### 3.4.4 Applications to Structured Document Retrieval

Statistical language modeling techniques have been applied to web document retrieval as well as element retrieval.

#### Web Document Retrieval

Research in Web document retrieval has largely focused on *structure-aware* adaptations of statistical language models.

**Kraaij, Westerveld, and Hiemstra** The earliest applications of statistical language modeling to structured document retrieval were for the task of known-item Web search tasks. Kraaij, Westerveld, and Hiemstra [38] propose a *structure-aware* approach which uses a linear interpolation (Equation 3.21) of smoothed language models estimated from the title, document, and in-link text. Kraaij et al propose to use this as the representation of the document in the generative ranking model (Equation 3.15). However, they do not test this model, but instead linearly combine the query-likelihood scores resulting from each of the models evaluated separately:

$$\sum_r \lambda_r P(q|\theta_r) \quad (3.25)$$

where  $P(q|\theta_r)$  is estimated by linearly interpolating the maximum likelihood estimate from the representation with a representation type specific collection model. The representation type specific collection model is estimated using the maximum-likelihood estimation from all document representations of the same type. As this is a *score combination* approach, it is open to the same criticism the Okapi score combination approaches presented in Section 3.3.1.

**Ogilvie and Callan** Ogilvie and Callan [63] implement and test the *structure-aware* mixture of language models proposed but untested by Kraaij et al [38] on the task of known-item Web search. The work was the first experimental comparison of the *in-model* and *score* combination approaches. Ogilvie and Callan form a mixture model for each document from language models formed from document representations, such as the title text, text of the entire document, in-link text, and image alternate text. They do so by first estimating a language model using Dirichlet prior smoothing for each of the representations and then linearly interpolating these models (Equation 3.21). Like Kraaij et al, Ogilvie and Callan smooth the representation language models with representation-type specific collection models. A side-by-side comparison of the mixture model to the CombMNZ score combination approach (Equation 3.8) and other common score and rank combination approaches demonstrate that the mixture model is at least as effective and robust as

score and rank combination approaches. The mixture modeling approach also has the appeal that the evidence combination happens within the generative framework, and not as a post-retrieval ad-hoc combination of scores. Robertson et al [73] cite this work as in the same spirit as the Okapi BM25F retrieval model.

Ogilvie and Callan [65] later apply the mixture of multinomials approach to investigating known-item finding of emails in email discussion lists. They considered representations of the email message text, the subject of the thread, the text of the entire message thread, and the text of replies to the email. They also optionally applied a prior probability of relevance estimated based on the depth of the email in the thread, placing a strong bias on retrieving emails that started a discussion thread. This work reinforced earlier experiments finding that the in-model combination of mixture models was again at least as robust as score combination. Apart from the text of the email itself, the most beneficial representation was the subject of the email thread.

### XML Element Retrieval

The statistical language modeling techniques have also been applied to XML element retrieval, creating models that are both *structure-aware* and *result-universal*.

**Sigurbjörnsson, Kamps, and de Rijke** Sigurbjörnsson et al [78] rank XML elements using a *score combination* of the element's query-likelihood, the document's query-likelihood, and a prior probability of relevance estimated using elements in place of documents in Equation 3.19 and  $\beta = 1$ . Later work by Sigurbjörnsson et al [77] uses an in-model combination of language models estimated from the collection, element text, and document text. Kamps et al [34] thoroughly investigate of the use of element length prior probabilities, considering a wider range of values for  $\beta$ . For smoothing, they use a single language model estimated from all elements in the collection.

Sigurbjörnsson et al [80] also investigate *structure-expressive* applications of statistical language modeling techniques. They decompose content-oriented XPath (NEXI, Appendix A) queries into pairs of location paths and keyword queries. While similar to the contexts of JuruXML (Section 3.2.1), these pairs are more expressive as they allow the user to request that multiple query terms occur in the same element.

The system first estimates  $P(q_i|\theta_e)$  for each of the keyword queries  $q_i$ , preserving only the elements that match the corresponding path  $p_i$ . Elements matching the target restrictions of the query are then combined by multiplying the scores of the elements matching the sub-queries. In a case where multiple elements match a sub-query, then the maximum scoring element is used. The final score for an element is a weighted geometric mean of the scores from each of the subqueries.

This construction of rankings is a fairly strict interpretation of the NEXI query, although they do ignore the distinction between AND and OR operations. Sigurbjörnsson et

al [80] discuss other ways that the NEXI queries could be interpreted. Some alternatives include the following query reconstructions.

- *full content query*: the query is converted into a simple flat text query with only the target element's path preserved.
- *partial term propagation*: query terms may be propagated up several levels, relaxing the context restraints.
- *full term propagation*: the structure of the query is preserved, but each clause is augmented to contain all query terms.

These reconstructions vary the degree of how strict the constraints in the original query are interpreted. All are still strict in the sense that the returned elements are all of the type requested in the original query. Sigurbjörnsson et al find that the *full term propagation* reconstruction improves the recall of elements in queries that have multiple subqueries. When the notion of relevance is relaxed to allow elements that do not match the expressed desired target element path, Sigurbjörnsson et al [79] find that a simple keyword query constructed from all query terms evaluated against elements in the collection outperforms the above reconstructions.

While Sigurbjörnsson et al propose the *structure-expressive*, *structure-aware*, and *result-universal* retrieval model detailed above, the authors do not describe how the *structure-expressive* combination of evidence and query reformulations fit within the language modeling framework.

**Tijah** List et al [43] present the Tijah system, which ranks elements using the generative query-likelihood model (Equation 3.15) with Jelinek-Mercer smoothing and prior probabilities of relevance estimated proportional to the length of the element. The collection language model used for smoothing is a single, document-based collection model. List et al focus more on providing a *structure-expressive* retrieval model leveraging statistical language modeling techniques. They extend a region algebra by including probabilities. A region algebra provides operators such as containment and Boolean operations over regions of text. Regions are defined by a begin location, end location, and a probability. The probabilistic region algebra allows implementations that leverage database techniques for query optimization.

List et al describe how their model can support queries written in the NEXI query language. For combination of probabilities resulting from multiple `about` filters, List et al preserve the minimum belief for Boolean `AND` operators and the maximum probability for Boolean `OR` operators.

**Ogilvie and Callan** Ogilvie and Callan [61] propose a *structure-aware* and *result-universal* approach that explicitly models the hierarchical structure of the documents. They estimate a language model  $\hat{\theta}''$  for each element by smoothing the maximum likelihood estimates up then down the hierarchical structure:

$$\begin{aligned}
 &1. \text{ Estimate base model: } \hat{\theta}_e = (1 - \lambda_C) \cdot \text{MLE}(e) + \lambda_C \cdot \hat{\theta}_C \\
 &2. \text{ Smooth up: } \hat{\theta}'_e = \lambda_c(e) \cdot \hat{\theta}_e + \sum_{c \in c(e)} \lambda'_c(c) \cdot \hat{\theta}'_c \\
 &3. \text{ Smooth down: } \hat{\theta}''_e = \begin{cases} \hat{\theta}'_e & \text{if } p(e) = \emptyset \\ (1 - \lambda_p) \cdot \hat{\theta}'_e + \lambda_p \hat{\theta}''_{p(e)} & \text{otherwise} \end{cases}
 \end{aligned} \tag{3.26}$$

where  $c(e)$  returns the children of  $e$  and  $p(e)$  returns the parent of  $e$ . One difference between this model and others is that the text of the element  $e$  used in the maximum likelihood was taken to be the text in that element not contained in any of its children. However, in experimentation Ogilvie and Callan chose  $\lambda_c(e)$  and  $\lambda'_c(c)$  such that  $\hat{\theta}'_e$  would be identical to  $\hat{\theta}_e$  if its maximum likelihood were estimated from all of the text in that element and its descendants.

This model allows flexible combination of evidence from the direct children and parent language models. However, the model leaves many smoothing parameters unspecified per document, so this model was only ever investigated using simple length-based weighting schemes for  $\lambda_c(e)$  and  $\lambda'_c(c)$ , sacrificing much of the potential power for simplicity and efficiency. Despite the flexibility in smoothing there are some weaknesses to this hierarchical smoothing approach. For example, it is not possible in this model to directly control the weight on a document's language model in  $\hat{\theta}''$ . As an element is deeper in the tree, the weight on the document language model decreases rapidly.

In experiments at INEX, Ogilvie and Callan ranked elements using the generative likelihood model with the element length based prior probabilities. They did not find that setting  $\lambda_p > 0$  significantly improved ranking effectiveness.

Ogilvie and Callan also discuss using the NEXI query language. They combine the evidence from multiple subqueries through multiplication of the probabilities (using a probabilistic AND). When there are multiple elements matching a subquery, they take the probability of the maximum matching element. Rather than rewriting queries by copying terms to multiple about clauses as in Sigurbjörnsson et al's *full term propagation* strategy, Ogilvie and Callan hope that the richer hierarchical smoothing will have a similar effect while preserving the original intent of the user.

Ogilvie and Callan's approach has the intuitive appeal that there is extensive *in-model combination* of the evidence present within the document. However, setting the  $\lambda_c$  parameters differently or the use of a non-zero  $\lambda_p$  requires that much of the document struc-

ture be loaded into memory during ranking. For efficient retrieval, Ogilvie and Callan found that this may require efficient in-memory storage of the entire collection’s structure or replicating much of the the structure in the inverted index term postings, which can be difficult in a large-scale information retrieval system. Nevertheless, we feel that this model has intuitive appeal, and we draw inspiration from this work in Chapter 5.

**Kim, Xue, and Croft** Kim, Xue, and Croft [36] draw inspiration from Ogilvie and Callan [61] but hypothesize that the best weights for different elements may vary across query terms. Their model closely resembles the mixtures of representations presented in Ogilvie and Callan [63], but replaces the static  $\lambda_r$  weights with  $P_M(E_j|w)$  values dependent upon the query terms:

$$P_M(E_j|w) = \frac{P_M(w|E_j)P_M(E_j)}{\sum_{E_k \in E} P_M(w|E_k)P_M(E_k)}$$

where  $E_j$  is the type of an element or representation.  $P_M(E_j)$  allows a prior probability to be placed upon an element type and the authors used the maximum likelihood estimator for  $P_M(w|E_j)$ . For most experiments, the authors chose  $P_M(E_k)$  to be equal for all representations, but they did perform some experiments suggesting non-equal priors for element types may provide slight boosts.

The estimates of  $P_M(E_j|w)$  result in weights roughly proportional to the relative frequency of a query term in each of the element types in the collection. Although the set of training queries used in evaluation is quite limited, their results suggest that query-term dependent weights on representations may improve results over document retrieval and mixtures of representations.

### 3.5 The Inference Network Model

The Inference Network Model is based on Bayesian networks and is implemented in the InQuery [86] and Indri [84] retrieval systems. Inference networks have been successfully applied to many retrieval tasks over the years. The presentation of the model will closely follow the presentation of inference networks for information retrieval used in the Indri retrieval system [84]. The work with inference networks dates back to Turtle and Croft’s [86] presentation, which included an expressive query language for the creation of inference networks to represent users’ information needs through query operators representing Boolean relationships, synonymy, and structural relationships. The resulting model is *structure-aware*. This retrieval model has more recently been adapted to be *result-universal* and *structure-expressive*.

A Bayesian network is a directed acyclic graph where the nodes are random variables and the edges specify conditional dependencies between the variables. In the application

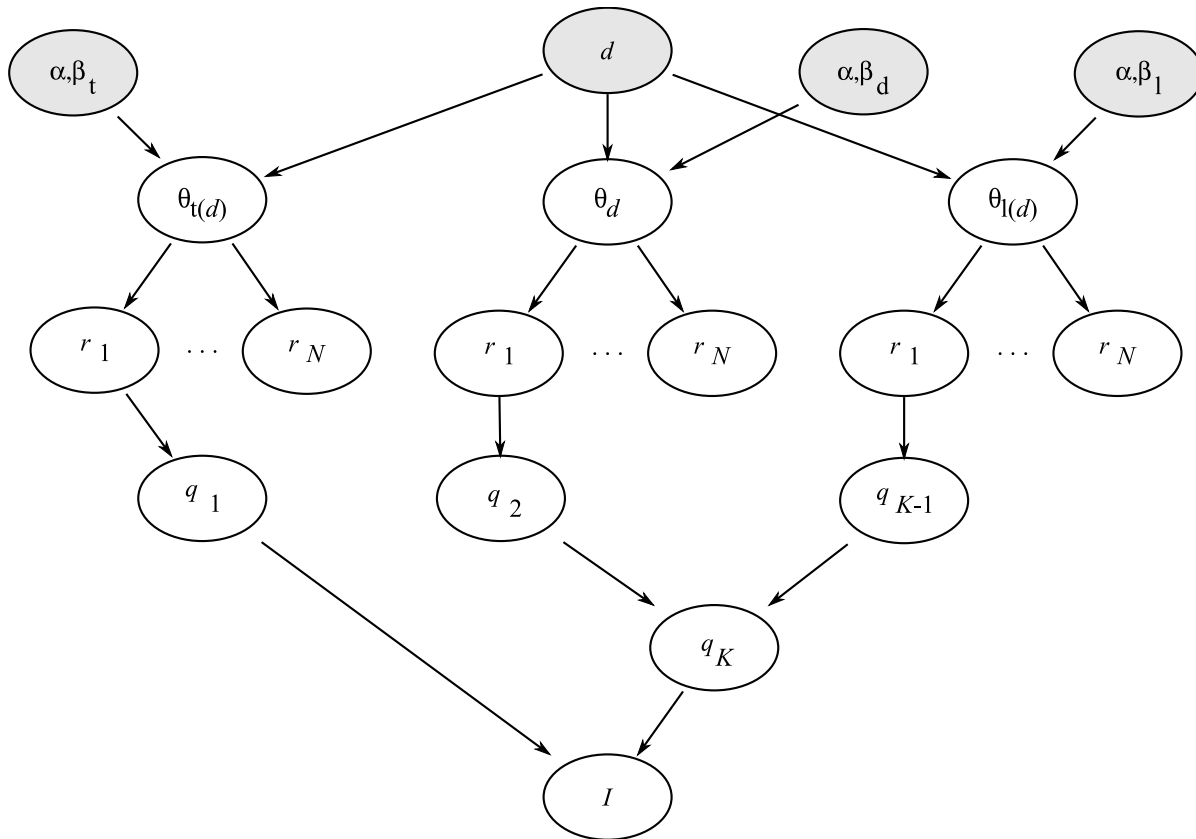


Figure 3.2: An example of Indri's Inference Network model.

of Bayesian networks to the Inference Network model of information retrieval, a Bayesian network is instantiated for each document and query. Queries are considered to be composed of concepts which represent terms, phrases, or other complex features. Documents are supposed relevant when the concepts in the query are present. Note that a term query concept being present in a document is not the same as the term occurring in the document; a document may contain the term 'surfing' without being about the concept of surfing. Yet an occurrence of 'surfing' does provide some evidence that the document may be about the concept.

Figure 3.2 depicts the general structure of the networks created using the Inference Network model. The shaded nodes correspond to observed nodes, such as the document  $d$  and model parameters  $\alpha$  and  $\beta$ . The language models  $\theta$  are estimated from the observations in the document and the prior parameters. Documents are ranked according to  $P(I|d, \alpha, \beta)$ , the belief that the information need is met given our observation of  $d$  and the parameters  $\alpha$  and  $\beta$ .

The representation nodes, labeled  $r_i$ , correspond to estimates of the belief that the con-



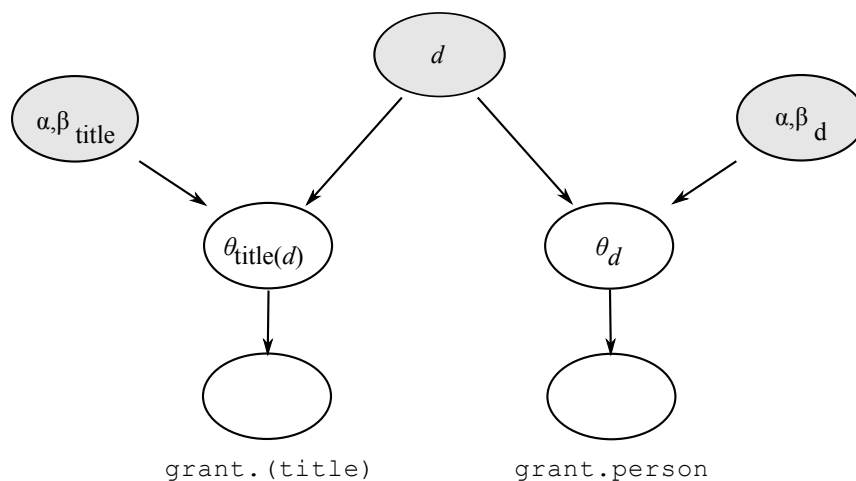


Figure 3.3: Belief estimation for field evaluation (left) and field restriction (right).

cept  $r_i$  was observed. These concepts are represented as a vector of binary values, where the location corresponding to feature  $r_i$  is 1 and all other values equal to 0. Given the structure of the network, the probability of observing concept  $r_i$  is dependent on  $d$ ,  $\alpha$ , and  $\beta$ . The probabilities  $P(r|\theta)$  in this model are estimated using the multiple-Bernoulli model presented in Section 3.4.3. Indri uses Equation 3.24 to estimate the beliefs for these representation nodes. The earlier formulations of the Inference Network model implemented in the InQuery system [86] used a linear combination of Okapi BM25 with a default belief.

### 3.5.1 Query Operators

The internal query nodes  $q_i$  correspond directly to the query operators expressed in the query that we will refer to as belief combination operators. These operators may include Boolean operations or weighted combinations of term/feature representation nodes or other query nodes. This section outlines the Indri query language, focusing on operators of specific interest for the material presented in this dissertation. There are several components to Indri queries: terms, feature representation operators, belief combination operators, and weights.

Terms in Indri refer directly to words in the vocabulary observed in a collection. Terms occurrences may be limited to a field type using *field restriction*: `grant.person` will only count occurrences of the term `grant` that occur in `person` fields during belief estimation. Alternative representations of the document can be created from fields using *field evaluation*, for example `grant.(title)`. This creates a language model from the `title` fields in the document. This is referred to as context restriction. Figure 3.3 shows how the two different representation beliefs are estimated in an inference network.

One would typically use the *field evaluation* syntax (`grant . (title)`) to create representations for use in ranking and use the *field restriction* syntax (`grant . person`) in cases where a term may be ambiguous and the annotation may help disambiguate its use in the document. For example, when searching for documents about people with the last name Grant, `grant . person` would be the appropriate way to restrict the count of occurrences for the term `grant` to named-entities indexed with the `person` type. If we wished to create representations from title elements present in a document, then field evaluation is appropriate.

The exact formula for the belief corresponding to `grant . (title)` is

$$\begin{aligned}
 P(r_1 = 1|d, \alpha, \beta) &= P(r_1 = 1|\theta_{\text{title}(d)}) \\
 \hat{P}(r_1 = 1|\theta_{\text{title}(d)}) &= \frac{\#(r_1, \text{title}(d)) + \alpha_{\text{title}}}{|\text{title}(d)| + \alpha_{\text{title}} + \beta_{\text{title}}}
 \end{aligned}
 \tag{3.27}$$

where

$r_1$	refers to the concept <code>grant</code> ,
$\text{title}(d)$	is a representation formed from all <code>title</code> elements in $d$ ,
$\#(r_1, \text{title}(d))$	is the number of observed occurrences of <code>grant</code> in $\text{title}(d)$ ,
$\alpha_{\text{aml}}$	is set to $\mu P(r_1 MLE(\text{title}(C)))$ , and
$\beta_{\text{title}}$	is set to $\mu(1 - P(r_1 MLE(\text{title}(C))))$ .

In contrast, the formula for the belief estimate corresponding to `grant . person` is

$$\begin{aligned}
 P(r_2 = 1|d, \alpha, \beta) &= P(r_2 = 1|\theta_d) \\
 \hat{P}(r_2 = 1|\theta_d) &= \frac{\#(r, d) + \alpha_d}{|d| + \alpha_d + \beta_d}
 \end{aligned}
 \tag{3.28}$$

where

$r_2$	refers to the concept <code>grant . person</code> ,
	which are occurrences of <code>grant</code> in <code>person</code> elements,
$\#(r_2, d)$	is the number of observed occurrences of <code>grant . person</code> in $d$ ,
$\alpha_d$	is set to $\mu P(r_2 MLE(C))$ , and
$\beta_d$	is set to $\mu(1 - P(r_2 MLE(C)))$ .

The difference in belief estimation for `grant . (title)` and `grant . person` is two-fold. For the former, the combined `title` elements is used as the context. Therefore, the length used in estimation is the length of the combined `title` elements. Also, the smoothing is not with a general collection model, but one estimated from only `title` elements in the collection. For the *field restriction* `grant . person`, the context is preserved as the

Operator	Name	Description
#ODN ( $f_1 f_2 \dots f_n$ )	ordered window	features separated by no more than $N$ tokens
#UWN ( $f_1 f_2 \dots f_n$ )	unordered window	features in a window of $N$ tokens in any order
#SYN ( $f_1 f_2 \dots f_n$ )	synonym	treat all terms/features as the same feature
#ANY: <i>type</i>	any	matches occurrences of an element type
#PRIOR ( <i>name</i> )	prior	include beliefs from prior <i>name</i>

Table 3.3: Feature representation operators in the Indri query language.

document, and the smoothing uses the standard collection model estimated from all text in all documents.

Feature representation operators allow the creation of complex features such as proximity and order constraints on terms and synonyms. Table 3.3 outlines the feature representation operators. The ordered window and unordered window operators allow the expression of phrasal and proximal constraints on features and terms. For example, #OD1(suicide bombers) specifies a feature where the words *suicide* and *bombers* occur in that order directly adjacent to each other. The synonym operator allows the terms and features to be aggregated together as a single feature. The “any” operator creates a feature representation node that matches occurrences of elements of the requested type.

As with terms, complex features can be restricted to occurrences within fields. A term/feature representation node is created for each term and feature representation operator not nested within another feature representation operator. Only feature representation operators not nested within other feature representation nodes have a corresponding inference network node in the query network. For example, in the query #OD1(suicide bombers), the query network does not have representation nodes for *suicide* or *bombers*; there is only a single representation node created for the concept #OD1(suicide bombers).

The prior operator allows the inclusion of features of the document independent of query terms to be a factor in the model. The beliefs produced by the inference network node for a prior are externally computed and stored within the system for run-time access. One example may be an estimate of the probability that a web document will satisfy the user’s information need conditioned on the observed length of the document’s URL. Unlike the other feature representation operators, the prior operator cannot be nested within other feature representation query operators. The interpretation and use of prior probabilities in the Inference Network model depends on how the prior operator is used in the query; we will return to this point later.

The belief combination operators combine the belief estimates of the nested terms,

Operator	Combination Function	Description
#AND ( $b_1 \ b_2 \dots b_n$ )	$\prod_{i=1}^n bel(b_i)$	probabilistic and
#NOT ( $b$ )	$1 - bel(b)$	probabilistic not
#OR ( $b_1 \ b_2 \dots b_n$ )	$1 - \prod_{i=1}^n (1 - b_i)$	probabilistic or estimate
#WAND ( $w_1 \ b_1 \dots w_n \ b_n$ )	$\prod_{i=1}^n bel(b_i)^{w_i}$	weighted and
#MAX ( $b_1 \ b_2 \dots b_n$ )	$\max(bel(b_1), bel(b_2), \dots, bel(b_n))$	takes the maximum belief
#WSUM ( $w_1 \ b_1 \dots w_n \ b_n$ )	$\sum_{i=1}^n w_i bel(b_i)$	weighted and

Table 3.4: Belief combination operators in the Indri query language.

feature representation, and belief operators. The terms and operators specified in the parameters to the operator indicate conditional dependencies in the inference network. The choice of the belief operator specifies how the distributions on which the inference network node is dependent are combined to form the belief estimates for the inference network node. The belief combination operators are presented in Table 3.4. We use  $bel(q_i)$  as a shorthand notion for the belief  $P(q_i|d, \alpha, \beta)$ . The belief combination operators of #AND, #OR, and #NOT provide probabilistic applications of basic Boolean operators. The #WAND operator provides a weighted version of the #AND operator. For example, #WAND(0.3 terrorist 0.7 #OD1(suicide bomber)) ranks documents on how well they match the concepts terrorist and #OD1(suicide bomber), placing more emphasis on matching #OD1(suicide bomber).

In addition to the construction of inference networks for ranking documents, the Indri query language supports *extent retrieval*. Extent retrieval allows the ranking of document elements of a specific type. Any belief operator can be modified by placing the desired element type  $\sigma_v$  in square braces prior to the list contained in parentheses. For example, if the user desires that only the paragraph elements be ranked against the query terms “suicide bombings”, then the corresponding query would be:

```
3.3 #AND[paragraph] ( suicide bombings )
```

This specification of result element types changes the inference network constructed. In this network, the  $d$  node is replaced by a node for a paragraph element. A separate inference network is constructed for each paragraph element. The smoothing parameters  $\alpha$  and  $\beta$  are updated in this case to use a background paragraph element models.

*Extent retrieval* is used instead of *field restriction* and *field evaluation* when there is a need to retrieve elements instead of documents. Extent retrieval can also be applied to belief operators nested within another belief operator. The nesting of extent retrieval has legitimate use cases (some examples of which we outlined in Chapter 1), but its use can be confusing. We will discuss these uses some in Chapter 5.

### 3.5.2 Applications to Structured Document Retrieval

Despite the long history of support for structured document retrieval in the Inference Network model, there is a surprisingly small amount of literature surrounding applications and experiments on applications to structured document retrieval.

**Myaeng et al** One application of the Inference Network model to structured document retrieval is provided by Myaeng et al [58]. This model, predating the current approach used in Indri described above, replaces the document or element nodes in Figure 3.2 with the entire structure of the document. Edges are created from parents to their children. In contrast to the current Indri retrieval model, a term occurrence is assumed to only be present in its leaf node. As a result, the representation nodes are connected to each of the leaf nodes.

Myaeng et al also describe how the model can be *structure-expressive* through restricting the connections of representation nodes to subsets of the structural element nodes. These subsets are constructed through the query specification. For example, a query requesting articles with “suicide bombings” matching paragraphs in the article could be converted to an inference network by connecting representation nodes for `suicide` and `bombings` to only the `paragraph` nodes in the inference network that are contained within the `article` element being ranked.

One challenge of this model is how the conditional probabilities in the inference network must be estimated. The authors choose to restrict belief estimation to the positive events when conditioning on the parent, resulting in computationally more simple model that has fewer conditional probabilities that must be specified. For inference network nodes corresponding to elements in the document, the authors consider two options. The first assumes that the probability of the event corresponding to that node given the observation of its parent is set to be proportional to the length of the element. This approach treats all term occurrences the same, regardless of the leaf element length or depth in the hierarchical structure of the document. The other approach assumes that some elements are more representative of its parent than others. They consider two ways of setting weights; the first weights the element by a type specific weight and its similarity to the parent, while the other weights the element proportionally to its similarity to other sibling elements.

In experiments, Myaeng et al use the former method, weighting the belief proportional to its similarity to the parent element. The authors find that this approach improves the retrieval effectiveness for keyword retrieval of patent documents, suggesting that the approach may be able to filter out the effect of “noisy” elements. They also find that when placing extra emphasis on the `BSUM` element type, which is the “brief summary” of the patent, improves retrieval effectiveness even more.

Through its use of element type weights and the similarity of an element to its parent

during estimation, the model achieves *structure-aware* properties. As it can rank any element and users can express structure constraints, the model is also *result-universal* and *structure-expressive*. However, one cannot incorporate context during ranking, such as the term occurrences in the document when ranking an element.

**Indri** The presentation of Indri and its inference network approach to modeling has built-in *structure-expressive* properties. Metzler et al [54] performs known-item retrieval of Web documents using the query template `#WAND( 0.15 ql 0.25 qt 0.10 qh 0.50 q )`. Each sub-query uses terms restricted to the corresponding language models using the `term.(type)` field evaluation syntax. The types they consider are a representation formed from in-link text, the `title` elements, html header elements, and the text of the Web page. Metzler et al find that a combination of this approach and phrasal expansions improves retrieval effectiveness over the rankings given by using a simple keyword query.

**Annotation Retrieval in Indri** Our work in Bilotti, Ogilvie, Callan, and Nyberg [5] reports on experiments using Indri to rank and retrieve answer-bearing sentences to factoid questions. They indexed semantic predicates found in sentences by the ASSERT semantic role labeler [68]. The semantic predicates identified by ASSERT are in the style of PropBank [37] predicates, where each predicate has a *target* verb indicating an action and multiple *arguments* and modifiers which specify the use of the target in the sentence. The target of a predicate were indexed as the parent of the arguments and modifiers of the predicate. They also indexed named-entities identified in the text by BBN Identifier [2].

Bilotti et al investigated the ability to retrieve answer-bearing sentences using two retrieval approaches. The most basic approach converted questions into a keyword + named entity query by identifying keywords in the questions and dropping stopwords. For questions where a specific named-entity type is expected as the answer, they added an `#ANY: type` operator to that query. These queries can be easily constructed automatically from the questions themselves. For example, the question

What year did Wilt Chamberlain score 100 points?

was converted into the Indri query

```
3.4 #AND[sentence]( #ANY:date wilt chamberlain score 100
    points ) .
```

The authors considered another approach which converted known answer-bearing sentences into structured-queries. A sentence is said to be *answer-bearing* if it contains the answer to the question without requiring inference using knowledge present in other parts of the document. These sentences were converted into queries by converting the

semantic predicates identified in the sentence into query clauses. The first step in the conversion was the identification of keywords and answer named-entity shared by the question and the answer-bearing sentence. For example, the answer-bearing sentence

3.5      On March 2, 1962, Chamberlain scored a record 100 points  
          in a game against the New York Knicks.

shares date named-entity containing the answer and the keywords `chamberlain`, `score`, `100`, and `point`.

These keywords were located within the named-entities present in the sentence, wrapping the terms of each named-entity with a query clause restricted to that named-entities type. The answer type was converted into an `#ANY` operator. For Sentence 3.5, Identifier's annotations were

3.6      On [DATE March 2, 1962,] [PERSON Chamberlain] scored  
          record 100 points in a game against the [ORG New York  
          Knicks.]

The named-entity conversion step resulted in the following query components: `#ANY:date`, `#MAX( #AND[arg0]( chamberlain ) )`, `score`, `100`, and `point`.

The next step located components in semantic predicates and added this structure to query components. For our example sentence, ASSERT identified the semantic predicate

3.7      [ARGM-TMP On March 2, 1962,] [ARG0 Chamberlain] [TARGET  
          scored] [ARG1 record 100 points] [ARGM-LOC in a game  
          against the New York Knicks.] ,

which was used to add structure to the partial query:

```
3.8      #MAX( #AND[target](
           score
           #MAX( #AND[argm-tmp]( #ANY:date ) )
           #MAX( #AND[./arg0](
                 #MAX( #AND[person]( Chamberlain ) )
               ) )
           #MAX( #AND[./arg1]( 100 point ) )
         ) )
```

Indri's query language was modified for these experiments to support the `./` operation in extent retrieval in order to represent the relationships between a target and its arguments.

The final structured query was constructed by wrapping all query components, including keywords and named-entities not present in a semantic clause, with a sentence retrieval operation:

```

3.9 #AND[sentence] (
      #MAX( #AND[target] (
          score
          #MAX( #AND[argm-tmp] ( #ANY:date ) )
          #MAX( #AND[./arg0] (
              #MAX( #AND[person] ( Chamberlain ) )
          ) )
          #MAX( #AND[./arg1] ( 100 point ) )
      ) )
    ) .

```

These queries were constructed for all known answer-bearing sentences in the collection for a given question.

The experiments compared the ability of Indri to retrieve answer-bearing sentences for the simple keyword + named-entity approach and the structured retrieval approach. The intent of the structured queries was to measure the ability of the retrieval system in the presence of a question-answering system that would formulate queries matching a semantic template for answers to the question. In the absence of such a system, Bilotti et al were forced to emulate an ideal question-answering system by constructing these queries from the answer-bearing sentences in the collection. Because the structured queries were constructed directly from answer-bearing sentences, the evaluation was optimistic, measuring the ability of the system to re-retrieve these sentences using the keywords shared between the question and answer-bearing sentence and the structure present in the answer-bearing sentence. The experiments found that the inclusion of the semantic structure in the queries did result in improved precision.

Further experiments explored the role of annotation quality. For these experiments, the authors explored sentence re-retrieval using hypothetical questions. The PropBank [37] corpus used had manually annotated semantic predicates on roughly one million words of news text. The queries were automatically constructed from the manually annotated semantic predicates. Retrieval was then performed on the manually annotated corpus and the corpus annotated by ASSERT, which correctly identified and labeled about 88.8% of the arguments and modifiers of semantic predicates. The experiments found that even when querying the corpus with semantic predicates identified by ASSERT, the semantic structure present in the query still helped the system achieve higher precision results than a keyword only baseline. These experiments, which smoothed the element/annotation with the document and corpus, provide some of the first experiments on investigating the role of *annotation-robustness* in a retrieval system.



## 3.6 Other Approaches

There is also related work that do not use any of the above retrieval models.

**CODER** Fox and France's work with knowledge-based modeling for retrieval of structured documents [20] modeled the structure of documents using frames. A frame is an object with typed attributes. For example, a `paragraph` frame may have a list of `sentence` frames, while `sentences` maybe lists of text items (words). This work was ambitious and required the construction of an extensive knowledge base. As such, there were few experiments evaluating this approach.

**Proximal Nodes** Navarro and Baeza Yates [59] considered the problem of querying structured text. They argue strongly for *structure-expressive* query languages and provide a detailed comparison of their proposed algebraic query operators to much of the related work in the area. While they do not explicitly address issues of ranking, it is an important piece of work as it establishes many desirable *structure-expressive* query operations.

**Logistic Regression** Larson [40] apply logistic regression of multiple factors to estimate the log-odds of each element's relevance. The logistic regression approach tries to estimate the probability of relevance for an element by a linear combination of features. Larson applies logistic regression to XML element retrieval by combining features of query length, element length, average *tf* for query terms in the element, average *idf* for query terms, and the number of the query terms present in the element. This approach is *result-universal*, but does not detail how it may be adapted to *structure-aware* or *structure-expressive* applications.

**MultiText** The MultiText system (Clarke et al [13]) supports passage retrieval and indexes documents and structural markup. This structural markup may be explicitly queried using the GCL query language [12]. The query language provides structure expressivity, and results may be passages or elements matching the query text.

**XQuery** XQuery\* is a query language for querying XML document collections. It is quite powerful, allowing complex element selection and transformation. The query language is certainly structure expressive, but it is targeted toward supporting text retrieval. The power of the language makes efficient implementations difficult; furthermore, the language is very complex, making it difficult to use by non-expert users.

---

\*<http://www.w3.org/TR/xquery/>

## 3.7 Summary

We have attempted in this chapter to succinctly but accurately discuss the related research on structured document retrieval. Given the sheer volume of literature on the subject, it is perhaps unsurprising that the chapter is not terribly succinct. However, we have aimed to discuss each work in the context of their retrieval model and whether they have the desirable properties for structured retrieval models. Figure 3.1 provides a visual timeline of major work in the area, but we will further summarize the desirable properties here.

**Result-Universal** The desire to rank any element has been addressed by much related research. One of the most salient findings is that length normalization is very important for *result-universal* retrieval systems. There has also been experiments addressing whether a single background model for *idf* or smoothing is adequate, or element type specific models give better results. In general, the research has found that a single background model is sufficient.

**Structure-Aware** Many researchers have investigated whether treating the occurrences of terms in different element types differently can improve retrieval effectiveness. For Web page retrieval of known-items, this has been clearly shown to be beneficial. For more ad-hoc retrieval tasks, the benefit of a *structure-aware* retrieval model has been less clear. Some of this results in difficulties in parameter estimation.

A parallel debate has been how the evidence from related elements should be combined. A simple and efficient way to do this is to use *score combination* approaches. However, *in-model combination* is often more intuitively appealing as the combination is explicitly addressed in the retrieval model. In addition *score combination* can have the undesirable side effect of breaking model properties, such as causing linear term frequency effects.

**Structure-Expressive** It has been a challenge to create *structure-expressive* retrieval models on top of other retrieval models in a principled way. Work built on top of the vector space and language models has been largely ad-hoc. However, the inference network retrieval model has some *structure-expressive* adaptations which remain largely untested. One unanswered question in structure-expressive systems is how should the evidence from multiple nested elements matching a query term be combined when ranking an outer element.

**Annotation-Robust** Work in passage retrieval for question answering systems has done little beyond using traditional text retrieval models with simple extensions to treat named-entities elements as query terms in ranking. JuruXML is among the first works to be

---

applied to element retrieval for question answering, but the use of annotations in the retrieval system has not been thoroughly investigated.

There have been few approaches that are simultaneously *result-universal*, *structure-expressive*, and *structure-aware*. There has been very little research on *annotation-robust* retrieval systems. We next explore these properties in the context of task-oriented retrieval models developed for the tasks of known-item finding of web documents and element retrieval. We draw from observations on these experiments and the related work presented in this chapter to present extensions to Inference Network retrieval model in Chapter 5.

# Task-Oriented Retrieval Models

---

This chapter presents baseline models developed for each of the three example retrieval tasks of known-item finding of web pages, XML element retrieval, and retrieval supporting a question answering system. Each model reflects an effective retrieval model adapting a state-of-the-art statistical language modeling and inference network techniques to the retrieval task at hand.

The language model presented for known-item finding in Section 4.1 serves primarily as a validation that a simple language modeling approach performs very strongly for retrieval of known items. The XML element retrieval models presented in Section 4.2 focus on a simple language modeling approach to support keyword retrieval and a basic extension of that approach to inference network models to support structured queries. Finally, Section 4.3 considers the extended inference network model for the retrieval of sentences containing linguistic annotations.

## 4.1 Language Models for Known-Item Finding

When ranking web pages for the task of known-item finding (Section 1.1), we observe that structural features of the text (such as titles) or inter-document structure (such as link text) can serve as useful representations of the content of a web page. Furthermore, for the specific task of known-item finding, words occurring in these alternative representations can be especially predictive of which queries a page is relevant. This work closely reflects the mixture of language models investigated by Ogilvie and Callan [63]. The approach to combining the evidence from multiple representations used in this model has the effect of *in-model* combination, and was cited by Robertson et al [73] as in the same spirit as the fielded Okapi BM25F retrieval model.

A simple model for ranking web pages estimates a unigram language model for each representation of a document. The retrieval model estimates a representation's language model by smoothing the maximum likelihood estimate formed from the text of the representation with that of the collection representation model formed from the concatenation of all representations of that same type:

$$\theta_{d,r} = \lambda_{d,r}MLE(d_r) + \lambda_{C,r}MLE(C_r), \quad (4.1)$$

where  $r$  is a representation type,  $d_r$  is document  $d$ 's representation of type  $r$ , and  $C_r$  is the text of all representations of type  $r$  in the collection. The retrieval model combines these language models using linear interpolation to form a single model for a document:

$$\theta_d = \sum_{r \in R} \lambda_r \theta_{d,r}. \quad (4.2)$$

The retrieval model then ranks documents for a query by the probability of observing the query string generated from the unigram language model for the document:

$$P(q|\theta_d) = \prod_{w \in q} P(w|\theta_d)^{\text{tf}(w,q)}. \quad (4.3)$$

As smoothing using Dirichlet priors is generally quite effective, we choose to use it in this approach, setting  $\lambda_{d,r} = |d_r|/(|d_r| + \mu_r)$  and  $\lambda_{C,r} = \mu_r/(|d_r| + \mu_r)$ . We choose as  $\mu_r$  the average length of  $d_r$  in the collection.

While Dirichlet priors are used for smoothing, recall that we can also apply a prior probability of relevance to documents, such as those described in Section 3.4.1. These prior probabilities,  $P(R = 1|D = d)$ , may be used in ranking by multiplying the result of Equation 4.3 by this prior probability:

$$P(R = 1|D = d) \cdot \prod_{w \in q} P(w|\theta_d)^{\text{tf}(w,q)}. \quad (4.4)$$

We choose to optionally apply URL types priors, following Westerveld et al [90]:

**root** if the URL of  $r$  matches the pattern *http://xxx.com/* possibly followed by *index.html*,

**subroot** if the URL of  $r$  matches *http://xxx.com/yyyy/* possibly followed by *index.html*,

**path** if the URL matched *http://xxx.com/yyyy/.../zzz/* possibly followed by *index.html*, and

**file** if the URL of  $r$  did not match any of the previous cases.

While we use the same classes of pages, we do re-estimate parameters specific to the tasks and corpora upon which we center our evaluations in this section. The prior probability may be estimated for documents belonging to a particular URL type class by dividing the number of relevant documents in the relevance judgments on training topics with that URL type by the number of documents in the collection with that same URL type.

This simple *structure-aware* retrieval model has the benefit that it is easy to construct and has only the  $\lambda_r$  parameters left unspecified. In addition, its in-model combination of the representations exhibits the non-linear term frequency aspects desirable in ranking. For known-item finding of Web page documents, we choose as representations title text, in-link text, and the body of the document.

	<b>WT10G</b>		<b>.GOV</b>	
Number documents	1,692,096		1,247,753	
Size (GB)	10		18	
Document types	html		html, doc, pdf, ps	
Task types	homepage finding		homepage and named-page	
	<b>t10ep samp.</b>	<b>t10ep off.</b>	<b>t12ki</b>	<b>t13mi</b>
Number topics	100	145	300	150

Table 4.1: Known-item finding testbeds

### 4.1.1 Evaluation Methodology

The TREC Web Tracks included a known-item search task several of the years during the existence of the track. Table 4.1 outlines the testbeds and corpora we use to evaluate the retrieval model. TREC 10 included a homepage finding task on the WT10G corpus, providing 100 sample and 145 official evaluation topics. We use Equation 4.2 to combine three representations: the *title* text, the anchor text from documents that *link* to the page being ranked, and the *body* text of the entire web page.

We use the sample topics (t10ep sample) and the official topics (t10ep official) for two-fold cross-validation to evaluate the retrieval model. Using these two folds will allow us to compare the results here directly with other results published for the task. The training phase of cross-validation sets the  $\lambda_r$  parameters and the URL type priors, while the test phase uses these parameters on the held-out testbed to evaluate results.

TREC 12 included a known-item finding task on the .GOV corpus, which consisted of 300 homepage and named-page finding topics. TREC 13 provided a mixed homepage finding, named-page finding, and topic distillation task using the .GOV corpus, from which we omit the topic-distillation task to make the topic set more comparable to that of the TREC 12 topics. The resulting subset contains 150 topics. As with the TREC 10 experiments, we use two-fold cross-validation with the TREC 12 known-item topics (t12ki) and the subset of the TREC 13 mixed topics (t13mi).

To evaluate the Homepage Finding task and the Named-Page Finding task, we use mean-reciprocal rank (MRR). MRR is the average over all topics of one over the rank where the first correct result was found. MRR was an official measure at TREC [29, 16] and emphasizes early success.

	<b>Representation</b>	<b>t10ep samp.</b>	<b>t10ep off.</b>	<b>t12ki</b>	<b>t13mi</b>
No priors	Body	0.274	0.309	0.351	0.333
	Link	0.508	0.531	0.478	0.446
	Title	0.397	0.333	0.359	0.355
URL priors	Body	0.747	0.695	0.441	0.374
	Link	0.635	0.615	0.527	0.486
	Title	0.771	0.636	0.472	0.504

Table 4.2: Performance of individual representations (MRR).

	<b>Combination Method</b>	<b>t10ep samp.</b>	<b>t10ep off.</b>	<b>t12ki</b>	<b>t13mi</b>
No priors	Equal $\lambda_r$	0.670	0.608	0.647	0.607
	MRR train $\lambda_r$	0.673	0.616	0.643	0.619
	MRR test $\lambda_r$	0.679	0.605	0.645	0.622
URL priors	Equal $\lambda_r$ + train priors	0.888	0.799	0.707	0.673
	Equal $\lambda_r$ + test priors	0.890	0.813	0.706	0.666
	MRR train $\lambda_r$ + priors	0.889	0.803	0.707	0.664
	MRR test $\lambda_r$ + priors	0.890	0.813	0.701	0.665

Table 4.3: Performance of combined models (MRR).

### 4.1.2 Results

Table 4.2 shows the training performance of the individual representations using Dirichlet prior smoothing with and without URL priors. These individual representations show varying degrees of effectiveness, and the relative effectiveness may change with the use of the prior probabilities. The values in this table were used to compute the  $\lambda_r$  parameters in the cross-validation.

We consider two approaches to setting  $\lambda_r$  parameters; equal parameters and parameters set proportional to the the MRR performance on the training topics. Table 4.3 shows the results of these approaches. The “train” approaches are results set using weights and, if applicable, prior probabilities estimated from the same set of topics, while the “test” approaches use weights and prior probabilities estimated from the corresponding fold of training topics in an effort to show how well the “train” parameters generalize. For example, the MRR test parameters for the t10ep sample approach are estimated from the t10ep official topics.

The t10ep official results using prior probabilities are better than the best official results for the task (0.774) [29]. The t12ki results are also similar to our prior results of 0.727 at TREC (best official results, [17]). Our official submission [64] also used the text of the URL, image alternate text, and meta tag keywords and descriptions, which resulted in a slight boost in performance over the results presented here. The t13mi are strong, but not as strong relative to the best systems at TREC [18]. The best performing system, submitted by Microsoft Research Cambridge [94], received a MRR of 0.738 on the homepage and named-page finding topics. Their system introduced Okapi BM25F and made use of PageRank instead of a URL type prior. Microsoft Research Asia submitted the second best system [83], which achieved a MRR of 0.703 through the use of HostRank (similar to PageRank, but for domains) and a wide array of keyword methods including some using term proximity. The third best system was submitted by the University of Amsterdam [35]. Their system received a MRR of 0.649 by using a very similar mixture of language models but a link in-degree prior in place of a URL type prior.

These results demonstrate that a simple unigram language model and a URL type prior can provide results competitive with other strong systems. Yet using the estimated  $\lambda_r$  values does not improve the performance over equally distributing weight across the representations. One potential cause for this unsatisfying result is that heuristically choosing  $\lambda_r$  proportional to the performance of a representation may not result in the best performance, even on the training set. Another possibility is that the estimated parameters do not transfer well from the training topics to the test topics. We explore these issues in the context of element retrieval below (Section 4.2).

## 4.2 Language Models and Inference Networks for Element Retrieval

In this section we propose a model for retrieving parts of documents, also known as element retrieval (Section 1.2). The model we propose for the retrieval of XML elements is also based on mixtures of language models. However, this retrieval model explicitly uses the structure of the document to smooth each element’s language model. The hierarchical language model we propose estimates the language model for an element  $e$  using:

$$\theta_e = \lambda_e MLE(e) + \lambda_d MLE(d(e)) + \lambda_C MLE(C), \quad (4.5)$$

where  $d(e)$  is the document containing  $e$ . This model is similar in spirit to the hierarchical smoothing of Ogilvie and Callan [61] while being more efficient to evaluate in an element retrieval system. This estimation approach also allows more weight to be placed on the document’s language model during smoothing.



In order to bias retrieved elements by their length, we include a length-based element prior (Section 3.4.1):

$$P(R = 1|E = e, Q = q) \propto P(R = 1|E = e)P(Q = q|R = 1, E = e), \quad (4.6)$$

where

$$P(R = 1|E = e) = c \cdot |e|^\beta \quad (4.7)$$

and

$$P(Q = q|R = 1, E = e) \approx P(q|\theta_e) = \prod_{w \in q} P(w|\theta_e)^{\text{tf}(w,q)}. \quad (4.8)$$

Recall that the constant  $c$  may be omitted during ranking. This simple model utilizes in-model combination of evidence through smoothing to incorporate the context of its containing document to improve estimates for which queries the element is relevant. The approach is both *result-universal* and *structure-aware*.

We also extend this model to support a subset of NEXI query operations (Appendix A) by converting the queries into the format supported by the inference network model (Section 3.5). However, we replace the beliefs estimated at the representation nodes with  $P(w|\theta_e)$ . The counts for composite nodes such as ordered window nodes input to the language models are computed as described in Section 3.5. The model interprets structural constraints strictly, meaning that only elements of the requested types will be ranked; for the query

```
4.1 //paragraph[about(., suicide bombings)]
```

only paragraph elements are ranked. The corresponding translation of the NEXI query is

```
4.2 #AND[paragraph]( suicide bombings )
```

For queries where the user requests that descendent elements of the requested element match query terms, we surround the nested extent restriction with a #MAX operator. For example, the query

```
4.3 //article[about(./paragraph, suicide bombings)]
```

requests that article elements be ranked by how well a paragraph element contained within the article matches the query terms suicide bombings. The corresponding inference network query is

```
4.4 #AND[article]( #MAX(
    #AND[paragraph]( suicide bombings )
) )
```

The inclusion of the #MAX operator is important because the default Inference Network implementation would connect all matching paragraphs to the inference network node

for the article, resulting in a multiplication of these beliefs. Zhao and Callan [97] discuss this issue in depth and we also revisit this question in Section 5.2 and Chapter 6.

Some NEXI queries place constraints on broader contexts than that of the result element. For example, the query

```
4.5    //article[about(., terrorism)]
        //paragraph[about(., suicide bombings)]
```

specifies that paragraph elements about suicide bombings be returned in articles about terrorism. There is no previously introduced query operator that expresses this, so we introduce the #CONTEXT operator, which treats elements or extents matching the first argument as the desired result, multiply its belief with those of the other arguments. We translate Query 4.5 into

```
4.6    #CONTEXT[article](
        #AND[paragraph]( suicide bombings )
        #AND( terrorism ) )
```

This model applies the length prior probabilities only to the element being ranked, allowing the user to specify a preference for longer or shorter elements. This *structure-expressive* extension of the inference network model allows for some limited support of structured queries.

### 4.2.1 Evaluation Methodology

For these experiments, we use INEX's IEEE v1.4 and v1.8 journal article test collections [23, 24]. We focus on evaluating element retrieval two query types: those with keywords only, and those with keywords and structural constraints. Table 4.4 outlines the testbeds and corpora we use to evaluate the retrieval model. The i2004k and i2005k topics correspond to INEX 2004's CO (content-only) and INEX 2005's CO topics. The i2003s and i2004s topics correspond to INEX 2003's CAS (content-and-structure) and INEX 2004's CAS topics. The queries for the keyword and structure topics are in the NEXI format (Appendix A). The official evaluation of the INEX 2004 CAS topics used a loose interpretation of the structured query, allowing any element to be considered relevant. However, we pruned the relevance judgments to contain only elements satisfying the structural constraints of the NEXI queries in the topic definition. This approach for constructing relevance judgments for strict interpretation of the structural constraints was common practice among those wishing to evaluate a systems ability to rank only the desired target elements. In addition to the conversion of NEXI queries described above, we ignore phrasal constraints and the '+' in front of phrases or keywords, and drop terms and phrases prefixed with a '-' from the query. These interpretations of the phrasal constraints, '+', and '-' suggestions are common practice among many INEX participants.

	<b>IEEE v1.4</b>		<b>IEEE v1.8</b>
Number documents	11,980		17,000
Size (MB)	531		764
	<b>i2004k</b>		<b>i2005k</b>
Keyword topics	34		29
	<b>i2003s</b>	<b>i2004s</b>	
Keyword and structure topics	30	26	

Table 4.4: Element retrieval testbeds

As with the known-item finding experiments, using the two folds per task allows us to compare the results here directly with other results submitted to INEX. To evaluate system performance, we use MAP, which corresponds to a variant of the official MAep used in INEX 2005 (mean average effort-precision) where all elements marked relevant to any degree are considered equally relevant. All statistical significance tests are performed using a one-sided permutation test with a sample size of 50,000 [82]. Tests are corrected within a collection and topic set by controlling the false discovery rate using Benjamini and Hochberg’s method with a desired p-value of 0.05 (Section 10.7 of [89]). Statistical significance tests are performed between the experimental results presented here and all of the official submissions.

We restrict ourselves to using the title fields of topics, even though description and keyword fields may also be present. In any case requesting a ranking where any element type may be returned (keyword queries, or structured queries with a wildcard filter), we restrict ourselves to ranking elements covered by the (somewhat long) list of element types listed in Appendix B. For keyword queries, we dropped quotations, ignoring phrasal requests and dropped terms with a minus in front.

### 4.2.2 Grid Search for Parameter Estimation

While the choices of parameters in Section 4.1 provided strong results for known-item retrieval, we do not feel it is sufficient for estimating the parameters of this element retrieval model. It was unclear whether the estimated parameters were any more appropriate than simply distributing the weight equally across the representations. Furthermore, the method of estimation required ranking elements by each of the representations. The model in Section 4.1 used Dirichlet priors to smooth each representation with an element specific collection model. The Dirichlet prior weights were chosen heuristically without any exploration. Finally, the model presented in this section uses Jelinek-Mercer smooth-

ing of the representations, where we use a single collection model as a representation and do not do Dirichlet prior smoothing of the other representations. This reduces the number of parameters we must estimate, but also means that we cannot use the method of setting representation weights proportional to the performance of each representation. Indeed, it does not make sense to rank elements solely by the collection language model. The method also provides no guidance on the choice of the element length prior  $\beta$ .

A very straightforward and simple approach to parameter estimation is to use a grid search to optimize the desired retrieval metric on some training topics. This direct maximization approach has some distinct advantages over methods that perform a guided search [50]. Grid search directly optimizes the desired retrieval measure and is also readily applicable to any form of parameters for any retrieval system. Other methods of optimization compute gradients [9, 10, 51], requiring adaptations of the retrieval source code or custom source code. Some methods only work on linear combinations of features [14, 15, 32, 22].

Computational complexity might be a concern, but it is surprisingly manageable. It is easy to optimize cache usage in the retrieval system by scanning many parameter values one query at a time. This method can be easily parallelized across multiple machines. Other approaches must make a decision about the next set of parameter choices to evaluate after analyzing results across all queries. While still possible to dedicate machines to the evaluation of a query in the other approaches, this code is more specialized and queries of differing complexity may result in underutilized machines.

There are challenges when using a grid search to set model parameters. The granularity of the step size in the search can limit the optimality of the resulting parameters. If the retrieval performance is not very peaked in the search space relative to the step size, then we can be confident in our parameter selection. Otherwise, grid search will be insufficient, or a detailed and computationally expensive search would be required.

The number of parameter values that must be tested is exponential in the number of steps taken per parameter and the number of parameters. If we test  $s$  steps per parameter, with  $r$  representations and  $p$  additional parameters (such as a length prior parameter), then the number of total combinations of parameter values is given by

$$C(s + r - 1, r - 1) \cdot (s + 1)^p = \frac{(s + r - 1)!}{(r - 1)! \cdot s!} \cdot (s + 1)^p \quad (4.9)$$

For example, a search with 20 steps per parameter ( $s = 20$ ), the use of three representations (e.g. collection, document, and element language models) and the estimation of an element length prior ( $p = 1$ ) results in  $C(20 + 3 - 1, 3 - 1) \cdot 21^1 = C(22, 2) \cdot 21 = 4851$  parameter combinations that must be tested. With 20 steps per parameter, each  $\lambda_r$  will be tested with a step size of  $1/20 = 0.05$  over the values  $(0, 0.05, 0.10, \dots, 1)$ . Figure 4.1 shows the relationship between the number of representations and the number of valid

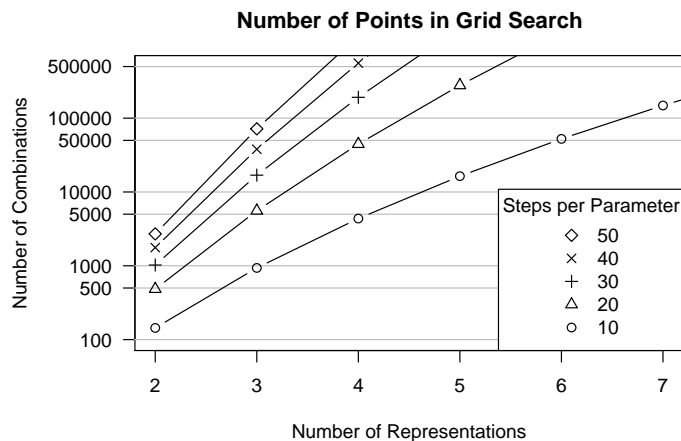


Figure 4.1: The number of parameter combinations for a range of representations, including an additional free parameter (such as a length prior parameter).

parameter combinations that must be tested for a range of steps per parameter and a single additional parameter ( $p = 1$ ).

If we assume that each query/parameter combination can be evaluated in one second on average, 100,000 parameter combinations would take just under 28 hours to perform all combinations for a single query on a single-core single-processor machine. Multiple cores and/or processors further reduce the time required. Given current hardware limitations and testbed sizes, testing up to 100,000 parameter combinations is reasonable, thus one can easily experiment with 10 steps per parameter for up to six representations and one additional parameter.

If we require a smaller step size when estimating parameters, we must rely on a multiple pass grid search, where the first step searches at course granularity and the subsequent steps do more refined grid searches. In this case, we are left with the same drawback of other parameter estimation methods: if the search space is not convex, we do not know whether we have reached a local or a global maximum. However, if the granularity of the earlier passes are fine enough relative to the steepness of the retrieval performance, we can be fairly confident of the quality of the resulting parameters.

A simple grid search does not control overfitting. Yet there are practical ways to reduce the risk of overfitting. Working with retrieval models that have only a few parameters that must be estimated can greatly control the risk of overfitting. Expert knowledge of the retrieval task can be valuable in developing a model with a good feature set that is not too large. The expense of grid search will help control the risk of overfitting, because the researcher must choose a small set of features to make the search tractable. Another approach to reducing the risk of overfitting is to ensure that the training collection con-

Parameter	i2004k	i2005k	i2003s	i2004s
Element	0.12 (0.04, 0.32)	0.24 (0.12, 0.40)	0.48 (0.04, 0.88)	0.48 (0.16, 0.84)
Collection	0.64 (0.40, 0.80)	0.36 (0.16, 0.60)	0.12 (0.04, 0.48)	0.20 (0.04, 0.40)
Document	0.24 (0.16, 0.32)	0.40 (0.20, 0.64)	0.40 (0.08, 0.88)	0.32 (0.00, 0.56)
Length	0.96 (0.72, 1.20)	1.20 (0.84, 1.68)	0.96 (0.60, 1.20)	0.84 (0.60, 1.08)

Table 4.5: Parameters estimated using a grid search with 25 steps per parameter. A 95% bootstrap confidence interval for the best parameter values is shown in parentheses for each parameter.

tains many representative queries. Finally, analysis on the results of the grid search can help verify that the parameter choices are reasonable.

### 4.2.3 Results

To estimate the parameters, we used a grid search with increments of 0.04 for the mixture model parameters and increments of 0.12 for the length prior parameter over the range [0,3]. These choices resulted in 25 steps per parameter and a total of 9,126 parameter combinations. This level of granularity is sufficient to find nearly globally optimal parameters. The solid blue lines (without boxes) in Figures 4.2 show the results of the search. The smoothness of these lines and relative flatness in most regions suggest that the granularity of the search is reasonable.

We also performed a search with 10 steps per parameter, or a step size of 0.1 per mixture model parameter and 0.3 for the length prior parameter. The corresponding searches are shown in black with boxes in Figure 4.2. The search with 10 steps per parameter approximates the 25 step search, providing similar maxima with only 726 parameter combinations. The close approximation of the curve shape suggests that a search where subsequent passes have a small number of possible parameter values limited to a narrowing range of values would be a reasonable method for mixtures of language models for element retrieval tasks. The insignificant loss in performance suggests that a search with a step size of 0.1 for mixture model parameters may be sufficiently granular for this task.

Table 4.5 shows the resulting parameters from the grid search when using 25 steps per parameter. The table also shows a 95% confidence estimate for each parameter. We estimated the confidence interval using the bootstrap method, which repeatedly resamples the query set with replacement to estimate variance. In each sample, the summary statistic is computed (in this case, the parameter that optimizes MAP) and recorded. After the summary statistics for all samples have been computed, we constructed a 95% bootstrap percentile confidence interval from the summary statistics using the percentile method,

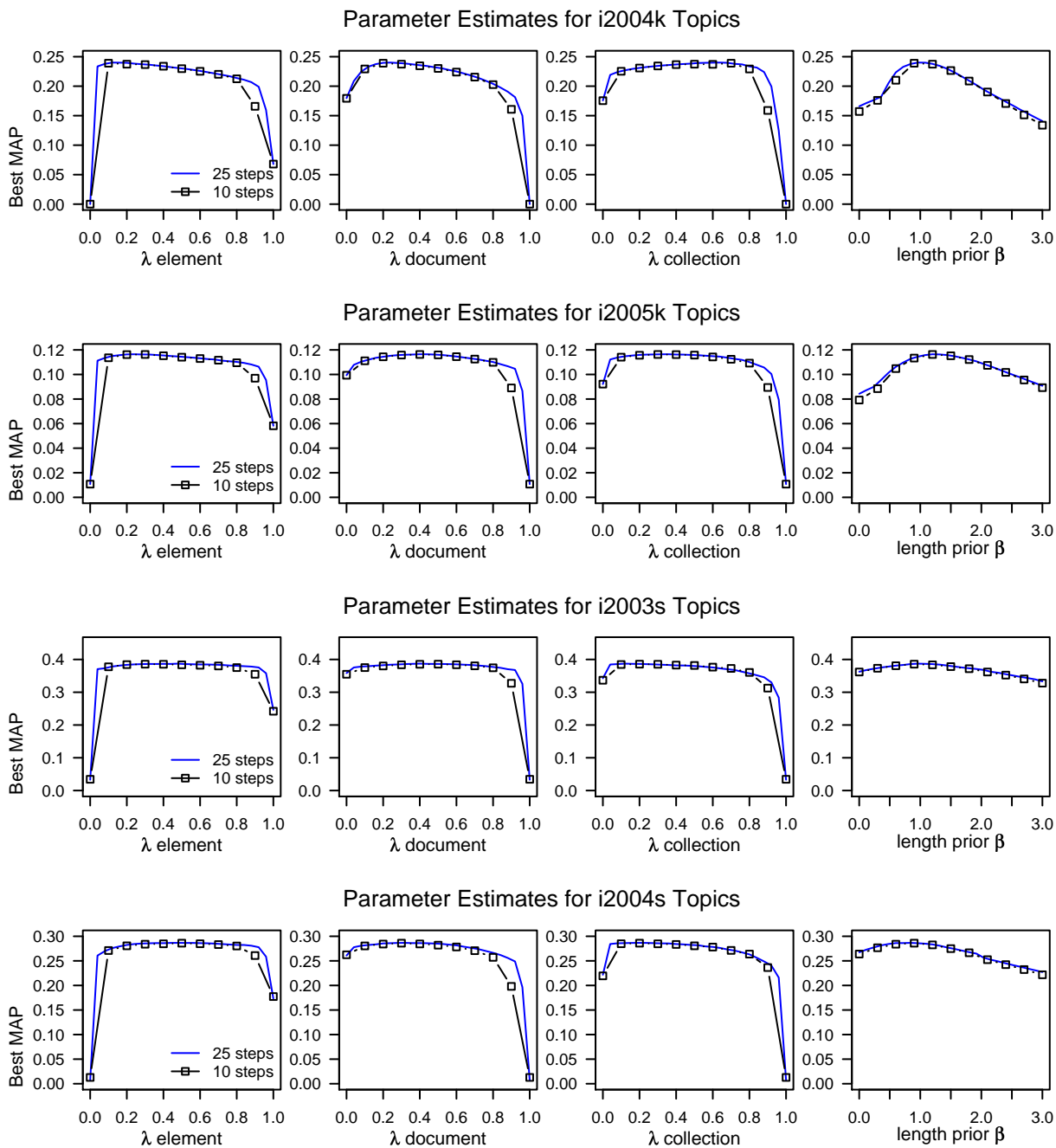


Figure 4.2: The grid search with 25 steps per parameter shows a fairly smooth search space. The height of the line represents the best mean average precision observed with the parameter value set to the value specified in the horizontal axis. A grid search with 10 steps per parameter approximates the shape of the more refined search reasonably well.

		Combination Method	i2004k	i2005k	i2003s	i2004s
25 steps	{	Train $\lambda_r, \beta$	0.240	0.116	0.387	0.287
		Test $\lambda_r, \beta$	0.235	0.113	0.384	0.284
10 steps	{	Train $\lambda_r, \text{priors}$	0.239	0.116	0.386	0.286
		Test $\lambda_r, \text{priors}$	0.234	0.112	0.384	0.280

Table 4.6: Performance of the element retrieval model (MAP). Using 10 steps per parameter instead of 25 during the grid search results in nearly identical performance.

which takes the middle 95% of the sorted statistics to form the confidence interval. We used bootstrap 10,000 samples during estimation. Chapter 8 of [89] contains a more detailed description of the bootstrap and its use to estimate confidence intervals. We can see in Table 4.5 that for most parameters the confidence interval is quite wide. Figure 4.2 demonstrates that for most topic sets and parameters there is a wide range of values that can achieve near optimal performance. It is perhaps unsurprising that the confidence intervals around the best parameter choices may be similarly wide. However, it is still true that as the training set size increases, it is expected that the width of the confidence intervals will decrease.

The variance of parameter optimality across test collections and query types should not be worrying. Figure 4.2 reminds us that despite these differences, the parameter estimate curves are remarkably similar, even with the small query sets used in this evaluation. As the training sets increase, we'd expect the curves for a query type to converge across the different evaluation folds. It is not a concern that the parameters for different query types (structured vs. unstructured) may be different; naturally one will get the best performance with representative training queries, but Figure 4.2 suggests that using any of the parameters in Table 4.5 on any of the test collections will yield reasonably good results.

### Comparison to official submissions

The comparison to official results in this subsection focus on the use of parameters estimated on the corresponding training fold. That is the i2004k results use parameters estimated from the i2005k testbed (and vice versa). The i2003s results use parameters estimated on the i2004s testbed (and vice versa).

Table 4.6 shows the results in MAP of the element retrieval model. Comparing the i2004k test results to official INEX submissions, we find that our results would have placed better than all of the official submissions. Our approach performed statistically significantly better than all but three of the submissions.



The three runs our system did not perform statistically significantly better than included one submission from the University of Amsterdam (0.235) [79], which used pseudo-relevance feedback. IBM Haifa submitted strong runs with MAP of 0.216 and 0.212 [47]. Their top scoring submission used pseudo-relevance feedback. Their other submission is interesting because it used a similar feature set to our approach. They linearly interpolated the cosine similarity of tf-idf weighted vectors for the element to the query with that of the document and the query.

Our i2005k results have a higher MAP than the official submissions, and all but eleven of these differences are statistically significant. Three runs from the University of Amsterdam with MAP of 0.104, 0.103, and 0.102 were the results of a language model system using features similar to the model presented here, but performed experiments on variations on which elements were indexed [77]. University of Waterloo's submission used term proximity features with OKAPI BM25 to achieve MAP of 0.102\*. IBM Haifa submitted a system with query expansion similar to their INEX 2004 submission that received a MAP of 0.101 [48].

For the i2003s testbed, our system receives a higher MAP than any of the official systems. Our system performed statistically significantly better than all but 6 of the 34 submissions. The top performing system, with a MAP of 0.379, was submitted by the University of Amsterdam [78]. Their submission combined the results of three rankings, each of which used queries constructed from words present in the title and description fields of the topic. The only use of the structural constraints they made was to filter the result list by desired result element. An article and element ranking was performed using pseudo-relevance feedback. The third ranking used the unexpanded queries to rank elements. The final ranking was computed by combining the scores of the element rankings with the score of the element's article.

The official evaluation of systems for the i2004s testbed interpreted the structural constraints in the NEXI queries only as suggestions. Many of the systems took this view of the topics, but some systems did interpret these constraints literally. Our system adopts the literal interpretation of the NEXI queries, so this comparison to official submissions from i2004s should only serve as a guide that our results are reasonable. For the i2004s testbed, no systems performed statistically significantly better than our model, and the model performed statistically significantly better than all but 7 of the submissions. The top performing system measured by MAP was the University of Twente's submission (0.352) [56]. Their approach takes a fairly strict interpretation of the NEXI query and combines language model beliefs estimated for each query clause and combines multiple matching subordinate elements by using a length weighted average of each element's belief. This submission also duplicates each `about` clause that has a structural

---

\*There was no paper published describing this submission, but the method was described in the comments of the official submission

constraint with an `about` clause that does not have that same restriction. For example, `//article[about(../kwd, genetic algorithm)]` would become

```
4.7 //article[about(../kwd, genetic algorithm)
      and about(., genetic algorithm)]
```

Their second best submission, with a MAP of 0.295, did not perform this query rewrite. The University of Amsterdam's top submission (0.303) also used language model beliefs, but propagated all query terms to every `about` clause [79].

### Query Analysis

Figure 4.3 shows a by-query comparison for the i2004k and i2005k testbeds of our model to the top performing system (per query) and the 75th percentile system (per query). Apart from Topics 190 and 229, our model performs quite well when compared to the 75th percentile results. The submissions to INEX's 2004 CO task used here for comparison on i2004k were optimized for a variety of tasks, while for i2005k we were able to restrict comparison to the *CO.Thorough* submissions of INEX 2005. Systems submitted to the *CO.Thorough* task were explicitly optimized for an evaluation measure very similar to the MAP measure used here, which could explain why the 75th percentile system per query seems to perform closer to the top system on the i2005k testbed than the i2004k testbed.

Figure 4.4 shows the by-query comparison of our system to official submissions on the i2003s and i2004s testbeds. Our model performed quite poorly relative to other submissions on Topics 61, 63, 77, 81, and 84 in the i2003s. The comparison to official submissions on the i2004s testbed includes systems which did not take a strict interpretation of the structural constraints, returning elements of types not considered relevant in our evaluation. This could be one cause of why the average precision of 75th percentile system is low for many topics. However, we note that many submissions did take the literal interpretation of structural constraints present in the queries. This is reflected in the high average precision for the best performance observed for many queries.

Table 4.7 shows a categorization of system failures on the keyword and structured testbeds. We also present an example topic from each of the failure modes. The author of Topic 229 requested information about latent semantic indexing and latent semantic analysis. Our system processed the topic title,

```
4.8 "latent semantic anlysis" "latent semantic indexing"
```

by removing the quotation marks from the query to create a keyword query. The misspelling "anlysis" was not corrected. Official submissions should also have been run without a spelling correction, so we do not believe that is the reason for our poor results. Upon further inspection, we observed that our results had many bibliography elements returned for this query, which the topic author explicitly stated were not relevant in the

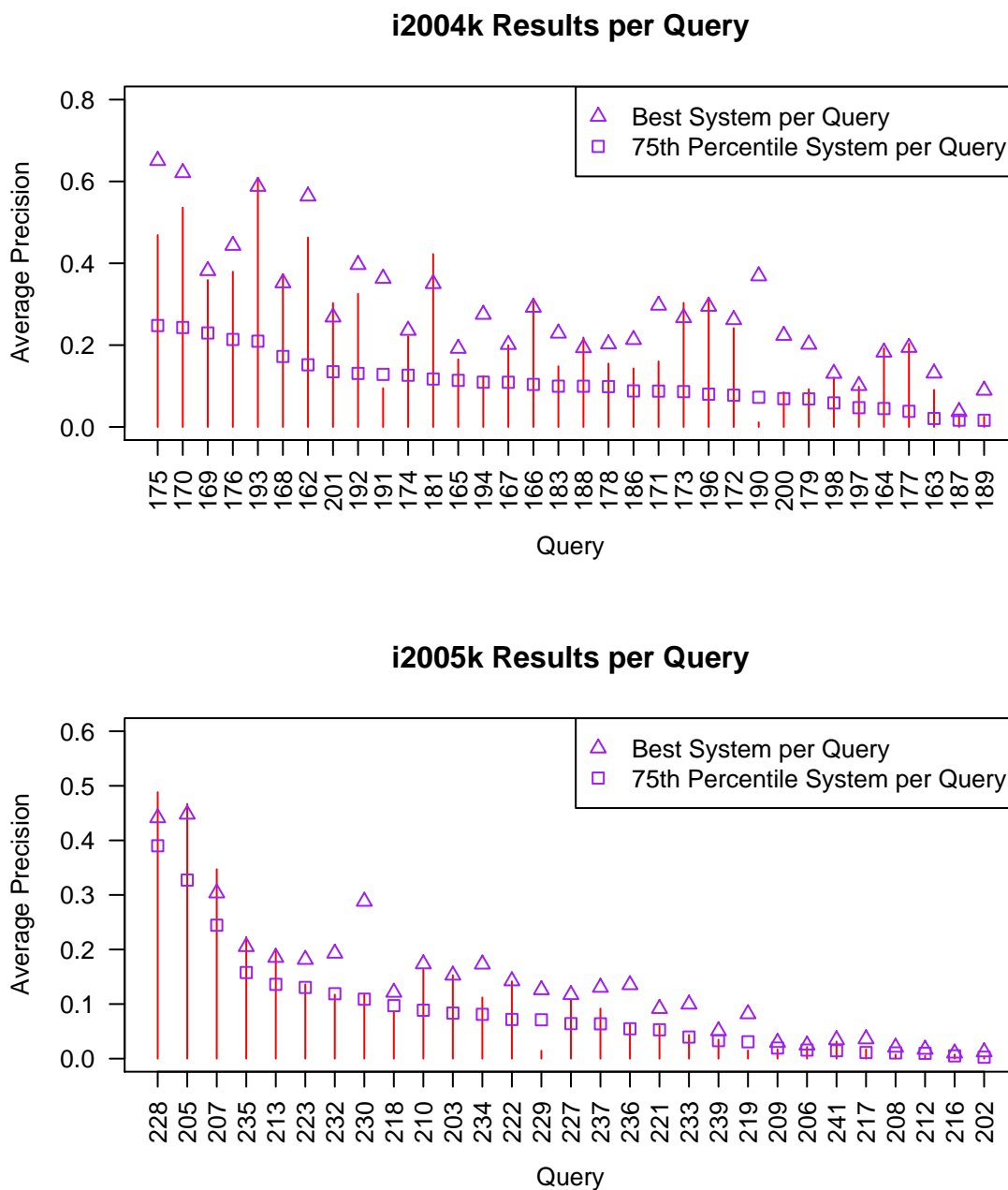


Figure 4.3: Results on keyword queries compared to systems submitted to INEX. Topics are sorted in descending order by the observed performance of the 75th percentile system for each query. With the exception of Topics 190 and 229, our results compare quite favorably with the INEX submissions. We expect there is more room for improvement of our system with the i2004k testbed than with the i2005k testbed.

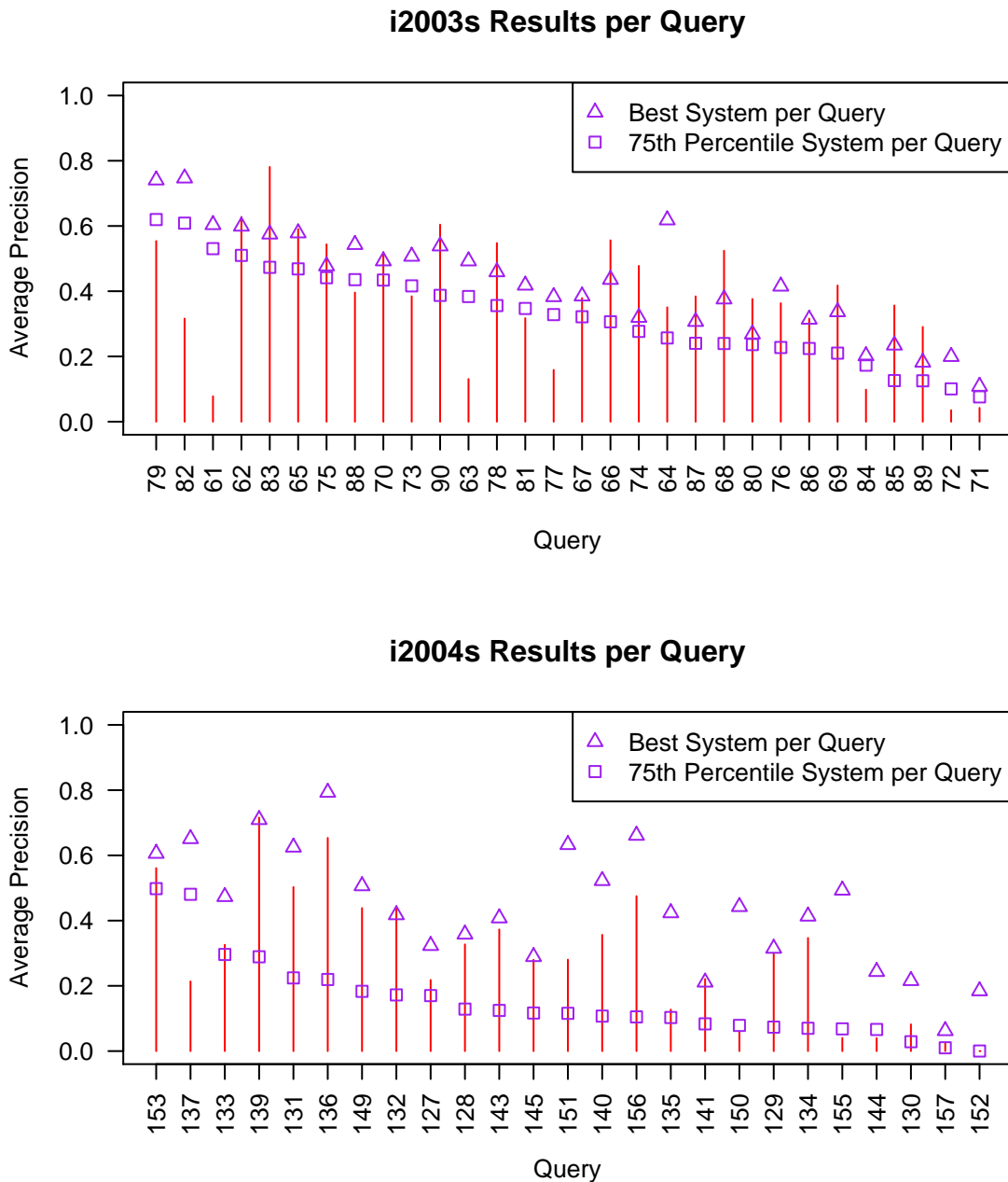


Figure 4.4: Results on structured queries compared to systems submitted to INEX. Queries are sorted in descending order by the observed performance of the 75th percentile system for each query. Our model performed worse than the 75th percentile system for several of the queries on the i2003s testbed. It is interesting that the 75 percentile system tended to perform quite closely to the best system for each topic in the i2003s testbed. This is not true for the i2004s testbed, although many systems did not take the strict interpretation of the topics that we used for evaluation. Our system performed noticeably worse than other submissions for only Topic 137 in the i2004s testbed.

	Topic Ids			
	i2004k	i2005k	i2003s	i2004s
Results biased toward irrelevant element type		229		
Too much smoothing			82	
Handling of poorly structured query			61, 63, 77	137
No use of phrase clues	190		72, 84	

Table 4.7: A categorization of system failures for keyword and structure queries, for topics where performance was noticeably worse than that of the 75th percentile system.

narrative of the topic. We suspect that higher performing submissions for this topic may have done a better job of returning fewer bibliography elements in their results for this query.

For Topic 82 better parameters would have greatly improved performance. The assessments for Topic 82's NEXI query

```
4.9 //article[about(.,handwriting recognition) AND
      about(./fm//au,kim)
```

took a very strict view on the requirement that the author contain `kim`. Our best results would have been achieved by using no length prior and performing no smoothing of an element with the collection or document language models. Doing so would have resulted in an average precision of 0.6 for that topic, which is closer to but still somewhat lower than the highest scoring systems.

Topic 63 is a clear example of a mismatch between the query structure and the information need. The query

```
4.10 //article[about(., "digital library") AND
      about(./p, +authorization
            +"access control" +security)]
```

places a restriction that there be a paragraph discussing “security, authorization, or access control matters” in digital library systems. The narrative similarly requests that there be “one or more paragraphs” discussing the security aspects. However, we believe that the query would better represent the information need by placing all keywords and phrases in the first about clause and dropping the paragraph clause. Any article discussing these issues at all will have at least one paragraph on the matter. Indeed, our performance for this query was strongest when we placed almost no weight on the element language model, instead placing most emphasis on the collection and document language models.

Finally, in some cases we believe ignoring phrasal suggestions may have harmed performance. The query for Topic 84 is one example:

```
4.11 //p[about(.,overview "distributed query processing" join)]
```

We believe that a better handling of phrases would have resulted in improved performance for several queries.

### Summary

Our retrieval model performed quite strongly for the keyword testbeds, with no official submissions demonstrating statistically significantly stronger performance. Systems with higher MAP often performed query expansion or used additional fields of the topic.

For the structured query evaluations, our system performed strongly despite our very literal interpretation of the structural constraints in the queries. None of the official submissions performed statistically significantly better than our model, even though a few had much larger MAP. This is in part due to the small query sets, but it is also due to differences in query handling from one system to another. Strict interpretations for structural constraints on keywords in queries hurts performance sometimes, but may be advantageous in other situations. Finally, our interpretation of phrasal constraints may have been detrimental to performance in some cases.

However, we do not wish to focus further investigation on query rewrites to relax our strict interpretations of structural constraints. Nor do we wish to expend effort on studying the interpretation of phrasal constraints in NEXI queries. This dissertation focuses on how best to use document structure in the ranking, making the most of the queries with which we are presented. We feel that these results demonstrate strong systems for use as baselines in further experiments regarding the use of document structure.

## 4.3 Inference Networks for Annotation Retrieval

This section proposes a model for the task of retrieving annotations to support other natural language processing applications. Specifically, we focus on the information needs of the hypothetical question answering system motivated in Section 1.3. We propose here that the adaptations of the Inference Network model used in Section 4.2 can be directly applied with some simple extensions to the query language. Taking the view that an annotation can be represented similarly to fields in the Inference Network model, the application of the model is fairly straightforward.

The experiments presented in this section are in a similar spirit to our prior work in Bilotti et al [5], there is one very important difference. In the prior work, we formulated queries directly from answer-bearing sentences, resulting in experiments which were overly optimistic about a question-answering system's ability to hypothesize the structure of semantic predicates contained in answer-bearing sentences. In this section,

we form queries from a manual analysis of the predicates contained directly in the questions themselves. While still optimistic about the system's ability to analyze the structure of a question, this form of analysis is analogous to the type of analysis performed when identifying the semantic predicates in the sentences present in the corpus. It is not beyond the realm of possibility for question answering system to perform this sort of analysis on questions. This results in experiments that share no overlap with those in our prior work, and we feel this change in query construction greatly improves the realism of the retrieval scenario.

In order to support the queries in Section 1.3, the Indri query language needs a way to express a parent-child structural constraint between annotations. This need is accommodated by the addition of an XPath-like structural constraint. Specifically, in an extent restriction nested in a query, the syntax `./type` allows the restriction of the query clause to rank elements of the requested type that are indexed as children of the outer element. For example, the query

```
4.12 RETURN target MATCH trained AND (suicide bombers) IN
      ./arg1
```

can be expressed as

```
4.13 #AND[target]( trained
      #MAX( #AND[./arg1]( suicide bombers ) ) ) .
```

Prior to supporting retrieval of annotations, the Indri retrieval system did not have sufficient index structures to support structural constraints. We briefly discuss extensions to the Indri index structures developed to support this model. The most simple way to support arbitrary structural constraints is to index the document structure as a graph and allow loading it for a document during query operator evaluation. However, loading the document structure for each document in a large corpus can be inefficient.

We can improve the efficiency of evaluating structural constraints by indexing additional information in the Indri retrieval system's field lists. An Indri field list is specific to an element type and stores the list of documents and elements of that type within each document. It is very similar to an inverted list with term locations, except that instead of storing a single location, the begin location and end location of the element must be stored. This allows Indri to efficiently identify which terms or elements are contained within the element in question during inference network evaluation. We extend the field list structure to additionally store a unique element identifier within a document and an identifier for the parent of the element (Figure 4.5). Augmenting the field list in this way enables parent and child structural constraints to be efficiently computed using a simple join operation on the field lists for the two element types.

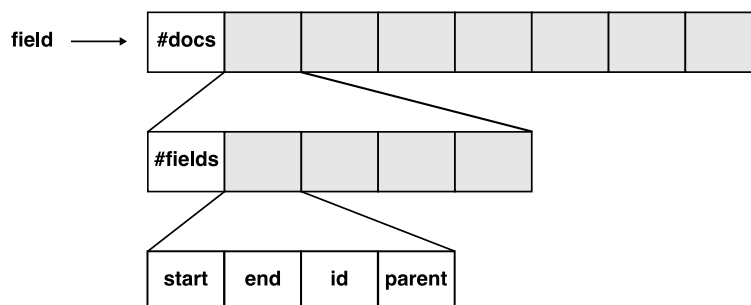


Figure 4.5: Field lists can be augmented with structural information for more efficient parent/child restrictions.

### 4.3.1 Experimental Methodology

The experiments in this section focus on the retrieval of sentences that contain the answers to factoid questions. The “MIT 109” questions [3, 42] we use have near-exhaustive document-level judgements for the AQUAINT corpus [27]. The AQUAINT corpus has 1,033,162 documents totaling 3.0 GB of text and contains news articles from the Associated Press (1998-2000), the New York Times (1998-2000), and stories written in English from the Xinhua News Agency (1996-2000).

Sentence level judgments are also available for the relevant documents in this collection [4, 5]. These sentence level judgments indicate whether the correct answer to the question can be inferred from the sentence and do not require additional context from the rest of the document for the inference.

Our goal is to retrieve these *answer-bearing* sentences from the AQUAINT collection. We have identified sentence boundaries with MXTerminator [70], named entities using BBN Identifier [2], and semantic predicates identified in sentences using ASSERT v0.11c [68]. The semantic roles identified by ASSERT use PropBank [37] style annotations. We indexed these annotations along with the AQUAINT corpus using Indri, the Krovetz stemmer, and the Inquiry stopword list.

The MIT 109 questions have been manually processed [4] to identify named entities, semantic predicates, and the type of semantic argument or named-entity that would contain an answer to the question. The manual tagging of the questions is unfortunately required because ASSERT is not able to process questions.

For training and testing, we use 5-fold cross-validation using the same folds as Bilotti [4]. The evaluation measure we use is MAP over retrieved sentences, making these experiments directly comparable to those in Chapter 6 of [4]. There are 18 questions with no relevant sentences in the AQUAINT corpus, so MAP is computed over the 91



questions with relevant sentences.

We consider two retrieval approaches. The first uses keywords and named-entity placeholders that correspond to the answer type. This approach is representative of most approaches to passage retrieval in question answering systems. The second approach makes explicit use of the semantic structure of the question and candidate sentences during ranking.

### Keyword + named-entity

This simple query creation method takes all keywords identified in the question and uses a #ANY operator for the answer type if a named-entity is the expected answer type. For example, question 1403

4.14    When was the internal combustion engine invented?

is converted into the query

```
4.15    #AND[sentence] (  
         internal combustion engine invent  
         #ANY:date  
         ) .
```

This simple conversion method is representative of much of the work around retrieval for question answering systems.

### Structured queries

We convert the semantic predicates in the questions into structured queries by creating a clause for each semantic predicate, where the target verb is connected with the terms in the arguments. This method is similar in spirit to the conversion of answer-bearing sentences into queries presented in [5], but uses the structure present in the query rather than a known relevant sentence.

Question terms not present in any semantic predicate are included in the #AND[sentence] operator that surrounds these terms and the subqueries for the semantic predicates. We also use an #ANY operator if the target answer type is a named-entity. For example, question 1403 is converted into the structured query

Parameter	Held-Out Fold				
	1	2	3	4	5
Element	0.3 (0.1, 0.5)	0.2 (0.1, 0.6)	0.3 (0.1, 0.5)	0.3 (0.1, 0.5)	0.3 (0.1, 0.5)
Collection	0.5 (0.3, 0.8)	0.7 (0.2, 0.8)	0.5 (0.2, 0.8)	0.5 (0.3, 0.8)	0.5 (0.3, 0.8)
Document	0.2 (0.1, 0.2)	0.1 (0.1, 0.3)	0.2 (0.1, 0.3)	0.2 (0.1, 0.2)	0.2 (0.0, 0.3)
Length	2.1 (1.2, 2.7)	1.5 (1.8, 3.0)	2.1 (1.2, 3.0)	2.1 (1.5, 2.1)	2.1 (1.5, 2.7)

Table 4.8: Estimated parameters for keyword + named-entity queries on the training sets corresponding to the held-out folds. A 95% confidence interval for each parameter is shown in gray.

```

4.16  #AND[sentence] (
        #MAX( #AND[target] (
            invent
            #MAX( #AND[./arg1] (
                internal combustion engine
            ) )
        ) )
        #ANY:date
    ) .

```

Answer-bearing sentences that match the semantic structure and keywords identified in the query should be ranked near the top of the result list.

### 4.3.2 Results

We employ a grid search with increments of 0.1 per  $\lambda_r$  parameter over the range [0,1] and 0.3 for the length prior parameter over the range [0,3], resulting in 11 possible values per parameter. Tables 4.8 and 4.9 show the parameter estimates on each of the training sets for the keyword + named-entity and structured queries. Figures 4.6 and 4.7 show the parameter estimate curves for the two query types. From these figures we see there is a fairly wide range of good parameter estimates for each parameter. While the magnitude of MAP can vary widely from one test fold to another, the shapes of the curves for the test folds are usually quite similar to those for their training data.

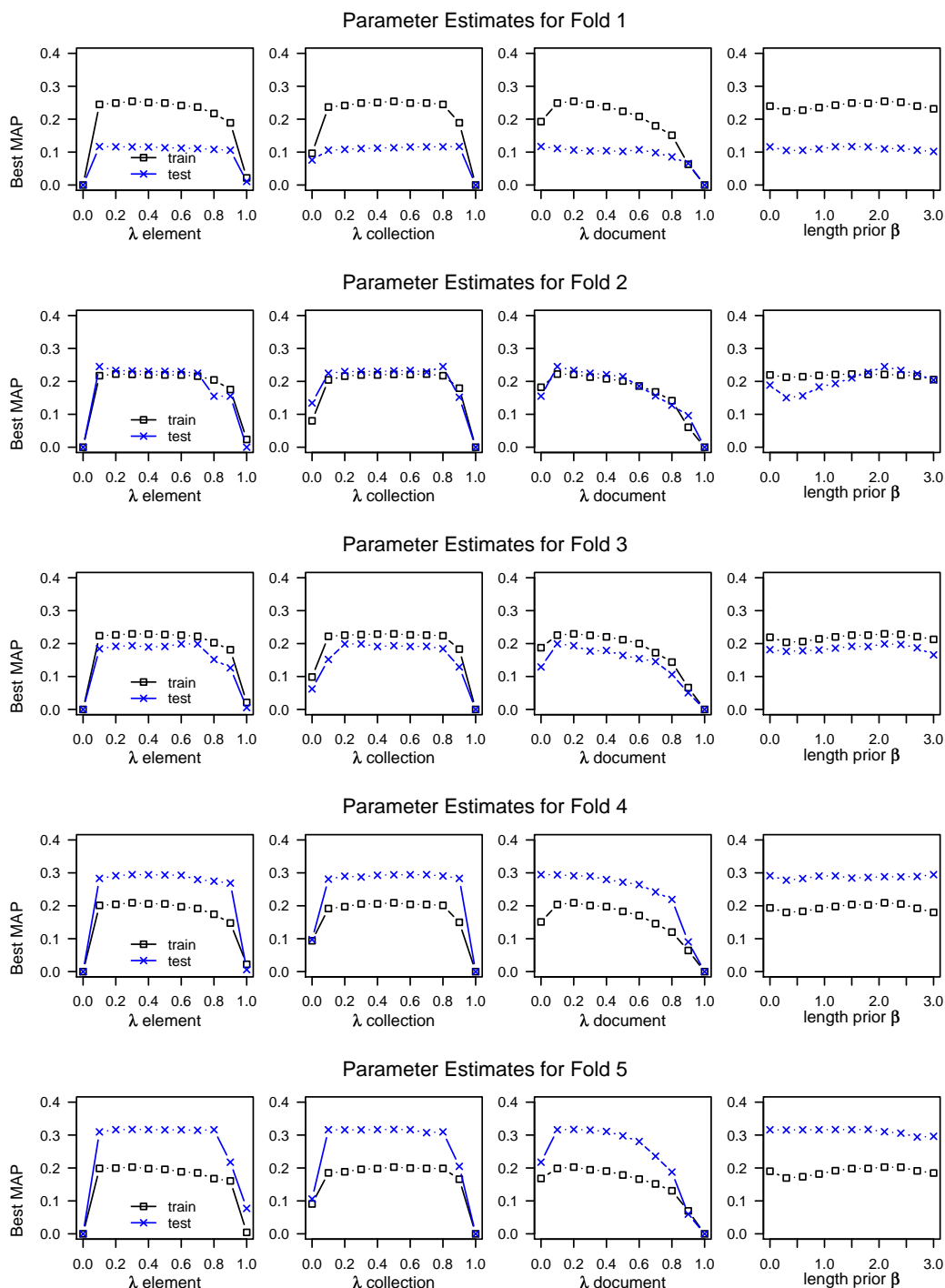


Figure 4.6: Parameter estimates for keyword + named-entity queries on training folds and test folds. While the test performance can vary significantly from fold to fold, the shapes of performance of the training folds are quite similar to with the test folds.

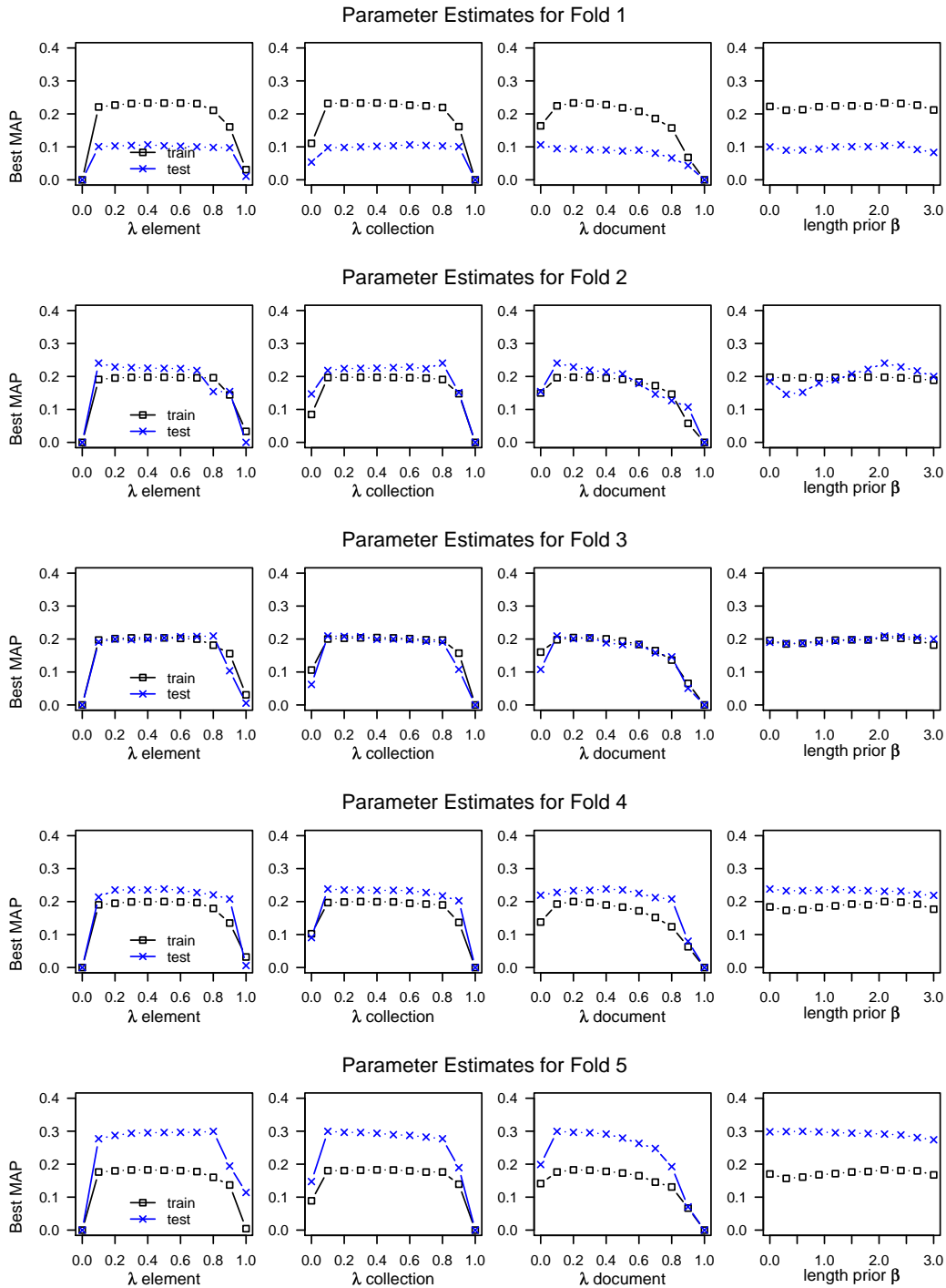


Figure 4.7: Parameter estimates for structured queries on training folds and test folds. While the test performance can vary significantly from fold to fold, the shapes of performance of the training folds are quite similar to with the test folds.

Parameter	Held-Out Fold				
	1	2	3	4	5
Element	0.4 (0.2, 0.6)	0.4 (0.2, 0.8)	0.4 (0.1, 0.5)	0.5 (0.1, 0.7)	0.4 (0.1, 0.6)
Collection	0.4 (0.1, 0.7)	0.3 (0.1, 0.7)	0.4 (0.1, 0.8)	0.3 (0.1, 0.7)	0.4 (0.1, 0.8)
Document	0.2 (0.1, 0.4)	0.3 (0.1, 0.5)	0.2 (0.1, 0.4)	0.2 (0.1, 0.3)	0.2 (0.1, 0.5)
Length	2.1 (0.0, 2.4)	2.1 (0.0, 2.4)	2.1 (0.0, 2.4)	2.1 (1.5, 2.4)	2.1 (1.2, 2.7)

Table 4.9: Estimated parameters for structured queries on the training sets corresponding to the held-out folds. A 95% confidence interval for each parameter is shown in gray.

The first row of Table 4.10 shows the performance (MAP). We see from these results that the keyword + named-entity queries outperform the structured queries. Bilotti [3] (Chapter 6) uses a similar keyword + named-entity baseline,  $T_{bbm}$ , reporting a MAP of 0.190. Bilotti wraps keywords occurring in a named-entity within  $\#MAX(\#AND[type](terms))$ . The stronger results reported here could be attributed to the differences in queries, the more thorough tuning provided by the grid search, or the inclusion of the length prior. Our results are worse than the learning-to-rank method proposed by Bilotti (with a MAP of 0.233), which includes numerous features that make use of the semantic structure and named-entities present in the query and candidate sentences.

Bilotti [3] (Chapter 6) observes that 54 of the 91 questions with answer-bearing sentences in the MIT 109 collection do not have semantic predicates. These *shallow* questions are a direct result of the fact that PropBank does not cover certain verbs, such as *is*, *be*, and *become*. The queries for these questions are identical for the two query conversion methods. *Deep* questions are those with semantic predicates. The second and third rows of Table 4.10 show the results if we estimate parameters separately on the shallow and deep questions (using the same five-fold cross-validation as before). The final row shows the combined results; training weights separately on shallow and deep questions does not improve performance for either query creation method.

### Query Analysis

Figure 4.8 directly compares the performance of the structured queries with the keyword + named-entity queries. We see 9 questions for which the structured query had at least 0.05 greater AP than the corresponding keyword + named-entity query and 14 where the structured queries performed at least 0.05 lower than the keyword + named-entity queries.

Inspection of these questions where the performance between the two query methods is large does not yield any obvious trends, so we present the two questions for each

Partition	Keyword + named-entity	Structured
All	0.218	0.201
Shallow	0.197	0.197
Deep	0.232	0.206
Combined	0.211	0.201

Table 4.10: MAP for retrieving answer-bearing sentences. The first row shows performance on test folds when using parameters optimized across all query types. The rows below show performance when using parameters optimized separately for shallow and deep queries, with the last row showing the combined shallow and deep results. Training separately for shallow and deep questions provides no improvement.

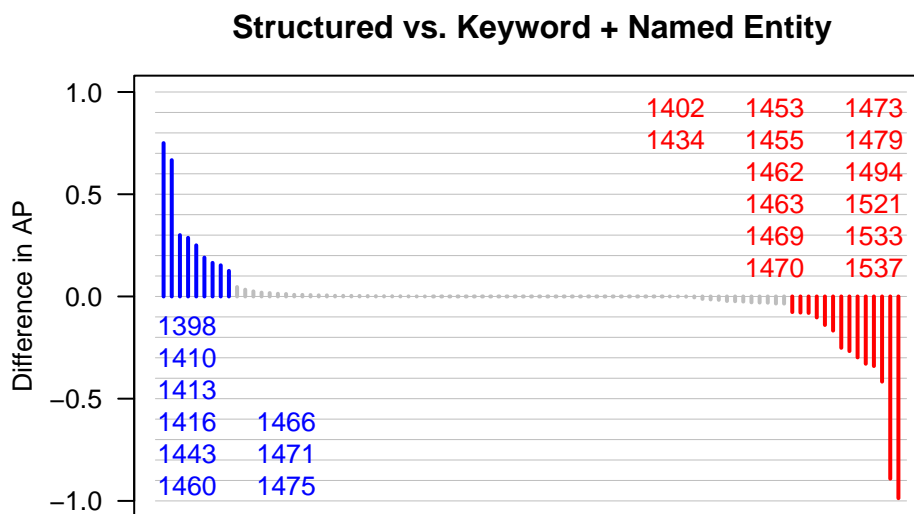


Figure 4.8: While the structured queries perform on average worse than the keyword + named entity queries, there are some questions where the structured query noticeably outperform the keyword + named-entity query. Questions where the structured query has AP at least 0.05 greater than the keyword + named-entity query are on the left of the graph in blue with numbers listed below. Questions where the structured query has AP at least 0.05 less than the keyword + named-entity query are shown on the left in red with numbers listed above.

method where the improvement over the other method was greatest. The biggest improvement for the use of a structured query is for Question 1410, What lays blue eggs?. The keyword + named-entity query conversion is

```
4.17 #AND[sentence] ( lay blue egg )
```

and the structured query for it is

```
4.18 #AND[sentence] (
      #MAX( #AND[target] (
            lay
            #MAX( #AND[./arg1] (
                  blue egg
                ) )
          ) )
    )
```

This structured query makes light use of structure, where the target verb *lay* is connected to an *arg1* argument matching *blue egg*. The *arg1* argument in this case is the recipient of the “laying” action. There is only one relevant sentence for this question:

```
4.19 A special breed of Araucana hen comes in many hues, from
     dusty pink to slate grey, and lay pale green or blue eggs.
```

which was annotated with two predicates, one of which did not match any query terms, the other marking *lay* as the target verb and *pale green or blue eggs* as its *ARG1*. Even though the identified semantic predicate was not perfect, failing to find *A special breed of Araucana hen* as its *ARG0*, the structure in the query and the semantic predicate match very well.

The other question for which the structured query outperformed the keyword + named-entity approach was Question 1460,

```
What was the name of the dog in the Thin Man movies?
```

This question contains no identifiable semantic predicates, so the difference in performance is only a result of the different smoothing parameters chosen during training. Question 1460 was in fold 2, which had placed a weight of 0.7 on the collection model for the keyword + named-entity queries; the collection weight for structured queries on fold 2 was only 0.3, with a weight of 0.4 on the element model and 0.3 on the document model. These more even weights for fold 2 on the structured queries are the cause for better the structured query approach’s strong performance for Question 1460.

Question 1494 was most harmed by the structured query. The question

```
Who wrote "East is east, west is west and never the twain
shall meet"?
```

was converted into the keyword + named-entity query

```
4.20 #AND[sentence]( wrote east east west west never twain
      shall meet #ANY:person )
```

and the structured query

```
4.21 #AND[sentence](
      #MAX( #AND[target](
            wrote
            #MAX( #AND[./arg1](
                  east east west west never twain shall meet
                ) )
          ) )
      #ANY:person
    )
```

There is only one answer-bearing sentence in the collection,

```
4.22 One hundred years ago, Kipling wrote, ``Oh, East is East,
      and West is West, and never the twain shall meet.''
```

ASSERT correctly identified that there was a semantic predicate with the target verb *wrote* and BBN did identify Kipling as a person. However, ASSERT failed to pick up the quotation as the ARG1 clause of the semantic predicate. Here we see that the retrieval model is not sufficiently *annotation-robust* to use the structured query for this question; annotation errors cause the more specific structured query to fail in this case.

Question 1453, Where was the first J.C. Penney store opened?, was the second most harmed question by the structured query approach. It was converted into the keyword + named-entity query

```
4.23 #AND[sentence]( first jc penney store open #ANY:location )
```

and the structured query

```
4.24 #AND[sentence](
      #MAX( #AND[target](
            open
            #MAX( #AND[./arg1]( first jc penney store ) )
          ) )
      #ANY:location
    )
```

There is a single answer-bearing sentence, which is duplicated twice in the collection:

```
4.25 In 1902, J.C. Penney opened his first store, in Kemmerer,
      Wyo.
```



This sentence was tagged with a single semantic predicate, with `opened` as the target verb, `J.C. Penney` as the predicate's ARG0, `his first store` as the ARG1, and `Kemmerer and Wyo.` tagged as locations. While this is a correct interpretation of the sentence, the ARG1 `his first store` is a poor match to the query's parse, which additionally placed `jc penney` in the ARG1.

### Summary

The question-answering experiments presented in this section demonstrate that while the use of semantic structure in queries is intuitively appealing, the retrieval model used here is not sufficiently *annotation-robust* to provide strong results. These results on the surface may seem contradictory to our prior work in [5], which found that even in the presence of errors in the annotation process a structured query could give better results than a keyword query. However, in the prior work, the queries were formed directly from the answer-bearing sentences. This resulted in retrieval scenarios we feel were overly optimistic about the question-answering systems ability to identify, in advance of retrieval, the structures that contain the answer to the questions. The results in this section represent a more realistic scenario, which relied only on the semantic predicates present in the questions themselves.

When the semantic structure in the question and the answer-bearing sentences match perfectly, the structured queries can do a better job of retrieving these sentences than an approach which does not use the structure. However, Questions 1494 and 1453 illustrate that this method is not robust to errors in the annotations or a mismatch between the structure in the query and the answer-bearing sentence. Yet for both of these questions there is a partial match between these semantic predicates. We believe that improving the annotation-robustness of the retrieval model may result in boosts to retrieval performance. We propose specific ways to address annotation robustness in Section 5.4, and revisit these experiments in Section 6.3.

## 4.4 Summary

This chapter presented retrieval models developed separately for each task of known-item retrieval of web pages, element retrieval, and retrieval of annotations supporting natural language processing applications. These models serve as baselines in the experiments in Chapter 6. Each model has its own strengths; the mixture of language models for known-item finding effectively combines evidence from alternative document representations, the element retrieval models present related approaches to ranking parts of documents and provides some support for structured queries, and the Inference Network model for annotation retrieval allows complex information needs to be expressed.

Section 4.2.2 provided a thorough investigation of the use of grid search for parameter estimation. The costs of grid search are surprisingly manageable on modern architectures, readily applicable to parameters of any form for any retrieval system, and can be used to directly optimize the evaluation measure of interest. The grid search results are helpful during failure analysis, enabling the examination the parameters yielding the best performance for that query. The data from the grid search also provide the ability to estimate confidence intervals around parameters.

The grid search employed for element and sentence retrieval demonstrate that a fairly coarse grid search is sufficient for estimating mixture-model parameters, with relatively flat performance estimates over a fairly wide range of parameter values. The wide values for the bootstrap confidence intervals for parameter estimates provides additional evidence that a coarse grid search is appropriate for parameter estimation in these retrieval tasks.

Yet the development of these models leaves us wanting. That they are different models makes it difficult to analyze properties of models that serve these three tasks well. The known-item retrieval experiments used a generative language-modeling approach. The approach to element retrieval for keyword queries used a different generative model, opting to use Jelinek-Mercer smoothing with a single collection model instead of using representation-specific collection models for smoothing using Dirichlet priors. These belief estimates were heuristically used within the Inference Network model for used structured queries in the element and answer-bearing sentence retrieval experiments.

The models separately address some of the desirable properties of models supporting the retrieval of structured documents and information needs, but none of them satisfy all properties. Each of these models uses statistical language modeling techniques for belief estimation, leading us to conclude that a unified retrieval model inspired by these separate models will form a better framework for understanding the properties necessary for effective retrieval.

# *An Extended Inference Network Model*

---

This chapter proposes a retrieval model based on the Inference Network model, extended to provide more support for structured document retrieval. Section 3.5 presented the Inference Network model and detailed how it has some desirable *structure-expressive* and *result-universal* properties. The model in this chapter connects the task-oriented retrieval models presented in Chapter 4 to the Inference Network model to provide a single framework for the investigation of the retrieval of structured documents, elements, and annotations. We also propose specific extensions to make it the Inference Network model more *structure-aware* and *annotation-robust*.

The changes proposed in this chapter are also motivated by a desire to clarify the use of the Inference Network model and the Indri query language. While the Indri query language may not have been designed for the casual user, it was designed with the researcher in mind. Our own anecdotal experience has found that researchers not intimately familiar with the Inference Network model and the internals of Indri have difficulty understanding how to construct good queries. In our adaptations of the model, we attempt to simplify the query language wherever possible, moving smoothing decisions outside the query language. We also propose changes to the query language to make experimenter decisions about extent retrieval in structured queries more explicit.

It is important that the distinctions between prior work and proposed extensions be explicitly stated. Metzler et al [53] formally connected the Inference Network model and statistical language modeling techniques through the introduction of multiple-Bernoulli estimation for belief operators. Strohmman et al [84] provided new extensions to network creation with the introduction of the Indri query language. These extensions allow ranking document elements using the extent restriction operator and the dynamic creation of mixture models through the use of the weighted sum operator and field restriction. They also recognized the need for document priors when ranking.

However, many issues surrounding ranking of structured documents, elements, and annotations remain unaddressed. The use of weighted sum and field evaluation in Indri allows limited ability to smooth across related representations for a document (Section 5.1), but its use and intuitions are not formally connected to the Inference Network and the multiple-Bernoulli model in prior work. Section 5.1 discusses these operations and formally connects the mixture models and the multiple-Bernoulli model in the Infer-

ence Network model. Section 5.1 also extends the representations available for smoothing beyond those contained strictly within the element being ranked.

The semantics of nested extent restrictions in the Indri query language are not well described in prior work. Section 5.2 considers issues surrounding how the evidence from multiple matching extents should be combined when ranking. In order to clarify the semantics of extent restriction, the unified model in this chapter uses a new `#SCOPE` operator to perform extent restriction rather than allowing extent restrictions on arbitrary belief nodes.

Finally, prior work with inference networks did little to address errors in annotations. Section 5.4 describes how one could model annotation errors within the Inference Network model.

## 5.1 Structure-Aware

In order to elegantly and flexibly specify and combine multiple representations within the Inference Network model, we introduce a representation layer (Section 5.1.2). The representation layer allows the grouping of related elements within a collection to form a representation (Section 5.1.3). Multiple representation nodes can be combined to form a smoothed model for an element (Section 5.1.4). This approach gracefully combines multiple representations and allows for the grouping of model choices within parameter files, simplifying the query construction process. First, we motivate why the existing approach within the Inference Network model is not sufficiently clear.

### 5.1.1 Motivation

While there is support within Indri to create *structure-aware* retrieval, we do not feel that this support is adequate. Consider a scenario where one wishes to place extra emphasis in ranking on terms occurring in the `title` elements in a document. There are several mechanisms within the Inference Network model and the Indri query language to isolate term occurrences in fields. We will use the example where we are searching for articles about suicide bombings, and consider transformations of the base query `suicide bombings`.

The first approach is through *field restriction*, which restricts counts in estimation to occurrences within that field. For example, in the query

```
5.1 #AND( suicide bombings suicide.title bombings.title )
```

the belief operator for `suicide.title` counts only occurrences of the term `suicide` in the document that occur in a `title` field. The belief estimate for this field restriction does

not create a separate representation of the title fields, but uses the document language model. Specifically, the belief estimate for `suicide.title` is

$$P(\text{suicide.title}|d, \alpha_d, \beta_d) = \frac{\#(\text{suicide in title of } d) + \alpha_d}{|d| + \alpha_d + \beta_d}. \quad (5.1)$$

The use of field restriction does not effectively create an alternative representation of the document, it just restricts counts to the requested field. As a result, field restriction is more commonly used to restrict term occurrences to annotations of certain types, such as a part-of-speech or a named entity type.

A different approach to using the title field is through extent retrieval, such as

```
5.2    #AND( suicide bombings
        #MAX( #AND[title]( suicide bombings ) ) )
```

In this approach, all title fields in the document are ranked and the belief of the best matching title field is selected. This method results in a *score-combination* approach, because the score of the best matching `title` element is multiplied by the scores of the concept nodes for `suicide` and `bombings`. We would ideally prefer an *in-model combination* method.

The first step for in-model combination is to use *field evaluation* to form a representation from the title field. Field evaluation is done through the syntax *term. (field)*, for example `suicide. (title)`. The belief estimate is formed by using the language model formed from the `title` fields present in the document:

$$P(\text{suicide.(title)}|d, \alpha_{\text{title}(d)}, \beta_{\text{title}(d)}) = \frac{\#(\text{suicide in title of } d) + \alpha_{\text{title}(d)}}{|\text{title of } d| + \alpha_{\text{title}(d)} + \beta_{\text{title}(d)}}. \quad (5.2)$$

To use field evaluation to combine beliefs in a query, one could write

```
5.3    #AND(#WAND(0.6 suicide 0.4 suicide.(title))
        #WAND(0.6 bombings 0.4 bombings.(title)))
```

The document's language model is used for the `suicide` and `bombings` query terms, while the title language model is used for `suicide. (title)` and `bombings. (title)`. Since the term occurrences in the `title` elements are also contained in the text of the document, this approach will place additional emphasis on the titles.

While this approach is in a sense an *in-model combination* as the ranking can be expressed by a single query, the effect is in practice identical to that of *score combination*. For Query 5.3, applying the belief estimation in the inference network shown in Table 3.4 results in a belief estimate of

$$\begin{aligned} P(I = 1|D = d, \alpha, \beta) &= [P(r_s|\theta_d)^{0.6} \cdot P(r_s|\theta_{t(d)})^{0.4}] \cdot [P(r_b|\theta_d)^{0.6} \cdot P(r_b|\theta_{t(d)})^{0.4}] \\ &= [P(r_s|\theta_d) \cdot P(r_b|\theta_d)]^{0.6} \cdot [P(r_s|\theta_{t(d)}) \cdot P(r_b|\theta_{t(d)})]^{0.4} \end{aligned} \quad (5.3)$$

where  $r_s$  denotes a representation vector where  $r_s = 1$  and the  $s$  dimension is represents the `suicide` concept and  $r_b$  corresponds to an observation of the `bombings` concept. Equation 5.3 shows that this application of the Inference Network model is no different from multiplying the belief of documents satisfying the information need expressed by the query `suicide bombings` by the belief for that same query estimated using a representation for those documents estimated from the `title` elements:

```
5.4    #WAND(0.6 #AND(suicide bombings)
        0.4 #AND(suicide.(title) bombings.(title)))
```

This is not what we believe the users would intend; they do not wish to rank documents by the belief that the document *and* its `title` representation are both about `suicide bombings`.

The developers of Indri were aware of this shortcoming, and introduced the `#WSUM` operator to allow for *in-model combination*. This weighted sum operator has the syntax `#WSUM( $w_1$   $b_1$   $w_2$   $b_2$ ... $w_n$   $b_n$ )` and combines beliefs using the formula  $\sum_{i=1}^n w_i \cdot bel(b_i)$ . Thus, our example query may be better written as

```
5.5    #AND(#WSUM(0.6 suicide 0.4 suicide.(title))
        #WSUM(0.6 bombings 0.4 bombings.(title)))
```

which results in documents being ranked by

$$P(I = 1|D = d, \alpha, \beta) = [0.6P(r_s|\theta_d) + 0.4P(r_s|\theta_{t(d)})] \cdot [0.6P(r_b|\theta_d) + 0.4P(r_b|\theta_{t(d)})] \quad (5.4)$$

where  $\theta_d$  and  $\theta_{t(d)}$  are estimated by smoothing the counts with Dirichlet prior parameters given by  $\alpha$ ,  $\beta$ ,  $\alpha_t$ , and  $\beta_t$ .

Here we see a major point of confusion. The  $\alpha$  and  $\beta$  parameters for smoothing are specified as parameters to the retrieval model, while the mixture models and their corresponding weights must be specified in the query language. Both are done with smoothing and the combination of representations in mind, but some parameters are in the queries and others are in parameter files. Placing some parameters in the query and others in parameter files is confusing. Furthermore, this approach requires the construction of long queries, where query terms are duplicated for each representation.

These smoothing parameters and representation choices are commonly set only at the batch experiment level. For most research, it makes sense to group all of these model choices within the parameter files. For example, in when one invokes `IndriRunQuery`, which searches for a set of queries and returns formatted results, one can easily set the smoothing parameters on the command line or in a parameter file. To use Dirichlet prior smoothing with prior parameter of 1500, one could include the parameter file:

```
<parameters>
  <rule>method:dirichlet,mu:1500</rule>
</parameters>
```

These smoothing parameters would be used for all queries. It would be natural to devise a similar mechanism for the specification of representations and the weights these representations should have. Grouping these model and smoothing choices within the parameter files would clarify and simplify the design and implementation of these experiments, removing the need to create complicated queries to perform conceptually simple smoothing tasks. Specifically, one would no longer need to use the weighted sum operator; Query 5.5 could be simplified to `#AND( suicide bombings )`, with the choice of representations (title, document) and representation weights (0.6, 0.4) specified in the parameter file and not in the query. In order to facilitate this, we need a mechanism to formalize the model choices one makes during representation selection, combination, and smoothing.

The Inference Network model presented in Section 3.5 uses a maximum-a-posteriori method to estimate the language models. It uses estimates of  $P(r_i|C)$  formed from observations across the entire collection to set the parameters  $\alpha$  and  $\beta$  of the multiple-Beta prior. It is conceivable that a *structure-aware* adaptation of the model could simply do this through choices of  $\alpha$  and  $\beta$  of the multiple-Beta prior that also make use of the related elements, such as titles, or related representations, such as in-link text. However, using the prior in this way obfuscates the process. The model itself would provide little guidance on how the observations in these multiple related representations, elements, and the collection model estimate should be combined to form a meaningful prior.

Yet another approach would use the method of Robertson et al. [73] to weight observation of terms from related elements. While this approach would effectively bias the estimation process to incorporate the evidence from related text, it does not explicitly state how the weighted term combination relates to statistical estimation techniques.

Instead, we believe that the introduction of a representation layer to the Inference Network model effectively captures the process used to create representations and combine them for ranking.

### 5.1.2 The Representation Layer

Rather than assuming that the text of an element or representation are the only valid observations of the concepts for which that element or representation are relevant, we instead assume that any observation in the collection may influence this estimate. The model achieves this by introducing a model representation layer, which is an intermediate layer between the observed text in the element nodes and the model nodes for the elements (Figure 5.1).

The model representation layer is similar in spirit to the text representation layer espoused by Turtle [86]. The observed collection elements are similar to the document nodes in Turtle's inference network. However, Turtle chose a one-to-one correspondence between the document and text representation nodes for simplicity. As a result, many presentations of Turtle's application of the Inference Network model simply omit the text

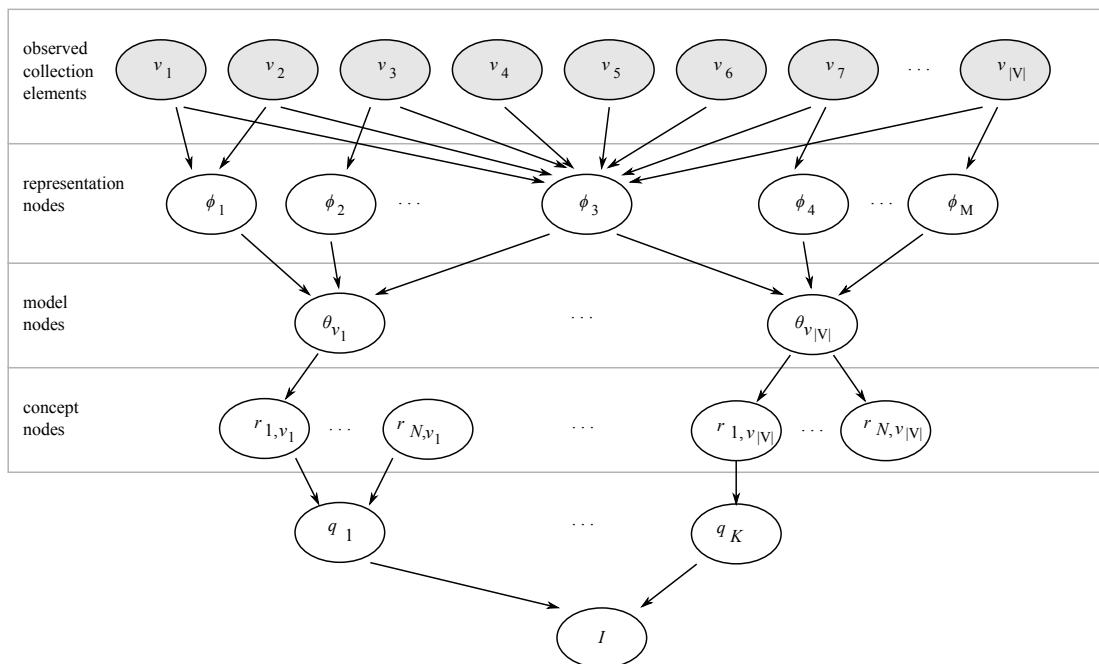


Figure 5.1: The proposed inference network modifies prior inference networks by introducing the representation models  $\phi$  between the observation nodes and the model nodes.

representation nodes. Our work reintroduces this concept through (model) representation nodes but does not restrict them with a one-to-one correspondence; a representation for an element may be estimated from one or many elements in a collection.

The (element) model nodes we use correspond directly to the language model nodes of modern applications of the Inference Network model proposed by Metzler et al [52], except that we estimate them through an interpolation of the representation nodes. Metzler et al's approach makes the same simplifying choice Turtle made: a one-to-one correspondence between model nodes and representation nodes.

The model nodes estimate for which concepts the element is likely to be relevant. It is typical that each model node is estimated by combining the evidence from a small number of representations. These representation nodes group the observations from related nodes into a single inference network node for use with the model nodes. Most representation nodes are connected to only a few observation nodes. However, the model does explicitly allow for representation nodes connected to very many observation nodes, such as one grouping all document elements to form a collection language model.

For example, the model node for news article in Figure 1.1 can be estimated from a representation node for the article itself, one for the title of the article, and a collec-



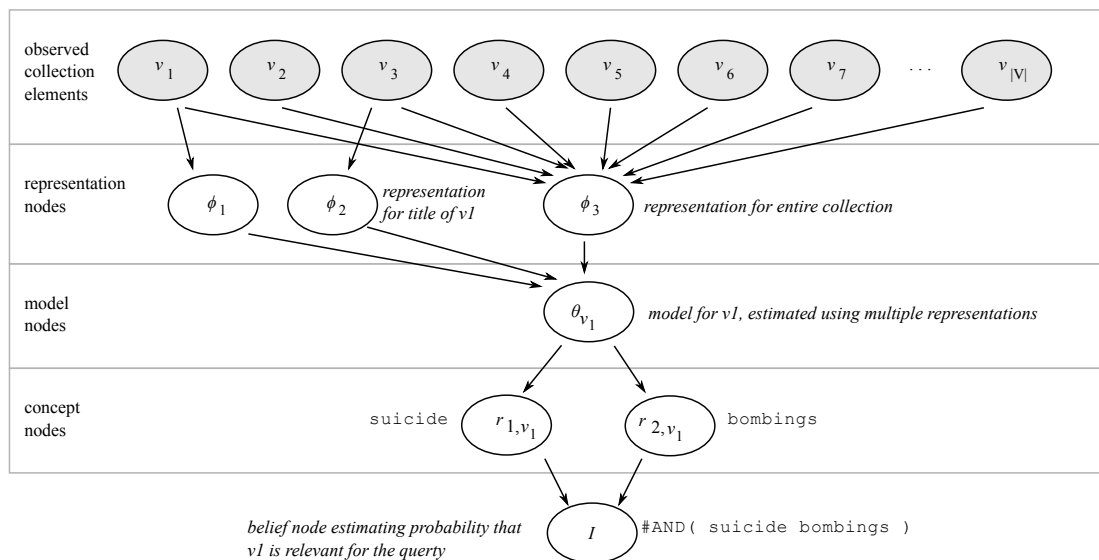


Figure 5.2: An example of an inference network for the query `#AND( suicide bombings )` using *element*, *title*, and *collection* representations.

tion language model. Figure 5.2 shows an instantiation of the inference network for estimating the belief that the article element  $v_1$  is relevant to the query `#AND( suicide bombings )`. The model node  $\theta_{v_1}$  is estimated using a mixture of three representations: the article element itself ( $\phi_1$ ), the title of the article ( $\phi_2$ ), and one formed from all elements in the collection ( $\phi_3$ ). The element representation  $\phi_1$  is only connected to the element's vertex  $v_1$ . Similarly, the title representation  $\phi_2$  for element  $v_1$  is only connected to the vertex  $v_3$ , the title of  $v_1$ . Finally, the collection representation  $\phi_3$  is connected to all vertices in the collection.

### 5.1.3 Creating Representation Nodes

The unified Inference Network model uses functions of the collection graph structure to identify the element observation nodes forming a representation for a model node. Each function  $f$  of the graph structure maps a vertex  $v \in V$  to a set of vertexes. Recall the one-to-one mapping between vertexes in the collection graph structure and elements. We use the terms “vertex” and “element” interchangeably. For example, the “self” function simply returns a set containing only the vertex to which the function is applied:

$$s(v) = \{v\}.$$

Similarly, the “document” function returns a single element set containing only the document that contains the element:

$$d(v) = \{v_i \in V : v_i \text{ is the document containing } v\}.$$

On the other hand, the “collection” function returns a set containing all documents in the collection:

$$C_d(v) = \{v_j \in V : v_i \in V \wedge v_j \in d(v_i)\}$$

Here, the  $C_d$  function returns all “document” elements. One could define a title collection function  $C_{\text{title}}$ , which constructs a set containing all `title` elements. The self and document functions are one-to-one, but the collection function returns a set of vertexes corresponding to the notion of “document” vertexes. In a corpus of news articles like the one presented in Chapter 2, the collection function may return all vertexes of the `article` type.

The functions return sets of elements that represent the model in a common manner, so it is reasonable to treat each observation of a term or complex feature within an element in that set as equally representative of the representation model. We view the observations of concept vectors contained within an element as sets of observations anchored in text. Therefore, when estimating the representation nodes, we form a set of observations from the set of elements returned by the function  $f(v)$ . A maximum likelihood estimator is a reasonable estimator for  $\phi_{f(v)}$ :

$$P(r_i | \phi_{f(v)}) = P(r_i | MLE(f(v))) = \frac{\text{obs}(r_i, f(v))}{\text{obs}(f(v))} \quad (5.5)$$

where  $\text{obs}(r_i, f(v))$  is the number of observations of the feature  $r_i$  in the union of the sets of observations for elements in  $f(v)$  and  $\text{obs}(f(v))$  is the total number of observations in the union of the sets of observations for elements in  $f(v)$ .

### 5.1.4 Estimating Model Nodes

Introducing the representation nodes and functions of the collection graph structure for their creation enables a very simple and direct way to specify how to create and estimate the model nodes. By specifying a set of representation functions, the designer of the inference network explicitly expresses which representation nodes should be instantiated and connected to the model node for an element. Generally, the choice of representation functions depends on both the corpus and the task. In principle, one could condition the set of representation functions on the element or its type. For example, a medical record search system may have element types where smoothing may best be performed by conditioning the representation function on the element type; it may be important to smooth

free-text patient notes differently than more structured fields such as treatments. However, for simplicity, this dissertation focuses on choosing a single set of representation functions for all elements in the collection.

Given a set of representation functions  $F$ , the retrieval model estimates  $\theta_v$  using a mixture of multiple-Bernoulli models:

$$P(r_i|\theta_v) = \frac{1}{Z_{F,v}} \sum_{f \in F} \left( P(f(v) \xrightarrow{r} \theta_v) \cdot P(r_i|\phi_{f(v)}) \right) \quad (5.6)$$

where

$$Z_{F,v} = \sum_{f' \in F} P(f'(v) \xrightarrow{r} \theta_v) \quad (5.7)$$

is a normalizing constant.  $P(f(v) \xrightarrow{r} \theta_v)$  represents the belief that an observation from the elements in  $f(v)$  is representative of the model of concepts for which  $v$  is relevant.

In derivations where the vertex being passed to a function of the collection graph structure is unambiguous, we may choose to simplify notation by omitting  $(v)$ . For example, Equation 5.6 can be rewritten as

$$P(r_i|\theta_v) = \frac{1}{Z_{F,v}} \sum_{f \in F} \left( P(f \xrightarrow{r} \theta_v) \cdot P(r_i|\phi_f) \right)$$

Writing  $P(f \xrightarrow{r} \theta_v)$  has the nice property that if one believes that set of observations in two different representation functions  $f_i$  and  $f_j$  are equally representative of  $\theta_v$ , then one can simply assert that  $P(f_i \xrightarrow{r} \theta_v) = P(f_j \xrightarrow{r} \theta_v)$ .

In traditional presentations of generative models, it is more common to write

$$P(r_i|\theta_v) = \sum_{f \in F} \left( P(f(v)) \cdot P(r_i|\phi_{f(v)}) \right) \quad (5.8)$$

where  $P(f(v))$  is the probability of choosing the representation  $f(v)$  in the generation process. The traditional presentation of generative models is misleading. It does not fully stress the goal that  $\theta_v$  be a model of the queries to which the element  $v$  is relevant. Using  $P(f \xrightarrow{r} \theta_v)$  reminds us of this process, and that we make assumptions about how representative a representation function is of the queries to which the target element is relevant when we choose a smoothing method.

### Equivalences with Smoothing Methods

By choosing  $F = \{s, C_d\}$ ,  $\phi(f(v)) = MLE(f(v))$ , and making additional assumptions about how  $P(f \xrightarrow{r} \theta_v)$  should be estimated, it is possible to show connections between this model and the smoothing approaches presented in Section 3.4.2.

**Jelinek-Mercer Smoothing** One can achieve the same effects as Jelinek-Mercer smoothing [96] by setting  $P(f \xrightarrow{r} \theta_v)/Z_{F,v} = \lambda_f$ :

$$\begin{aligned} P(r_i|\theta_v) &= \sum_{f \in F} \frac{P(f \xrightarrow{r} \theta_v)}{Z_{F,v}} P(r_i|\phi(f(v))) \\ &= \lambda_s \cdot P(r_i|\phi(s)) + \lambda_{C_d} \cdot P(r_i|\phi(C_d)) \\ &= \lambda_s \cdot P(r_i|MLE(v)) + \lambda_{C_d} \cdot P(r_i|MLE(C)) \end{aligned} \quad (5.9)$$

Jelinek-Mercer smoothing is equivalent to setting  $P(f \xrightarrow{r} \theta_v) = \lambda_f \cdot Z_{F,v}$ . Although the constant  $Z_{F,v}$  is unknown, Jelinek-Mercer smoothing asserts a belief that the relative representativeness of a representation function is a constant relative to the other representation functions. That is,  $P(s \xrightarrow{r} \theta_v)/P(C_d \xrightarrow{r} \theta_v) = \lambda_s/\lambda_{C_d}$  is a constant.

**Smoothing with Dirichlet Priors** We can derive an estimate equivalent to that of smoothing with Dirichlet priors [95, 96, 62, 90] by assuming that the document collection function  $C_d$  has a constant probability of being representative of the queries for which  $v$  is relevant, while the probability of representativeness of the observations from the element itself increases linearly with the element's length. We can model these assumptions by estimating

$$\begin{aligned} P(s \xrightarrow{r} \theta_v) &= \alpha \cdot |v|, \text{ and} \\ P(C_d \xrightarrow{r} \theta_v) &= \alpha \cdot \mu. \end{aligned} \quad (5.10)$$

Applying these assumptions to Equation 5.6 results in estimating

$$\begin{aligned} P(r_i|\theta_v) &= \frac{\alpha \cdot |v|}{\alpha \cdot |v| + \alpha \cdot \mu} \cdot P(r_i|\phi(s)) + \frac{\alpha \cdot \mu}{\alpha \cdot |v| + \alpha \cdot \mu} \cdot P(r_i|\phi(C_d)) \\ &= \frac{|v|}{|v| + \mu} \cdot P(r_i|MLE(v)) + \frac{\mu}{|v| + \mu} \cdot P(r_i|MLE(C)). \end{aligned} \quad (5.11)$$

Here we see that smoothing using Dirichlet priors asserts that the representativeness of an element relative to the collection increases linearly as an element's length increases. That is,  $P(s \xrightarrow{r} \theta_v)/P(C_d \xrightarrow{r} \theta_v) = |v|/\mu$ .

**Two-Stage Smoothing** To show the relationship between the mixture of multiple Bernoullis and two-stage smoothing, we must make a rather different assumption. We must assume that the representativeness of the element approaches a constant times our estimated representativeness of collection model as the length of the element increases. We can achieve this behavior by setting

$$P(s \xrightarrow{r} \theta_v) = \frac{|v|}{|v| + \alpha} \cdot \beta \cdot P(C_d \xrightarrow{r} \theta_v) \quad (5.12)$$

where  $\alpha, \beta > 0$ . By making this assumption, we find that

$$\begin{aligned}
\frac{P(C_d \xrightarrow{r} \theta_v)}{Z_{F,v}} &= \frac{P(C_d \xrightarrow{r} \theta_v)}{\frac{|v|}{|v|+\alpha} \cdot \beta \cdot P(C_d \xrightarrow{r} \theta_v) + P(C_d \xrightarrow{r} \theta_v)} \\
&= \frac{1}{\frac{|v|}{|v|+\alpha} \cdot \beta + 1} = \frac{|v|+\alpha}{|v| \cdot \beta + |v| + \alpha} \\
&= \frac{|v|+\alpha}{(\beta+1) \cdot |v| + \alpha} = \frac{\frac{|v|}{\beta+1} + \frac{\alpha}{\beta+1}}{|v| + \frac{\alpha}{\beta+1}}.
\end{aligned} \tag{5.13}$$

If we set  $\beta = \frac{1-\lambda}{\lambda}$  where  $0 < \lambda < 1$  then

$$\begin{aligned}
\beta + 1 &= \frac{1-\lambda}{\lambda} + 1 \\
&= \frac{1}{\lambda}.
\end{aligned} \tag{5.14}$$

If we also set  $\alpha = \frac{\mu}{\lambda}$ , then

$$\frac{P(C_d \xrightarrow{r} \theta_v)}{Z_{F,v}} = \frac{\lambda \cdot |v| + \mu}{|v| + \mu}. \tag{5.15}$$

From this derivation we can see that the two-stage smoothing approach of Zhai and Lafferty [96] can be expressed in the unified Inference Network model by assuming Equation 5.12 and using  $\phi(f(v)) = MLE(f(v))$ . If we examine the relative representativeness of the element's representation function to that of the collection, we see that

$$P(s \xrightarrow{r} \theta_v) / P(C_d \xrightarrow{r} \theta_v) = \frac{|v|}{|v| + \alpha} \cdot \beta$$

$\beta$  controls the asymptote of representativeness of the element relative to the collection (similar to the constant  $\lambda_s / \lambda_{C_d}$  in Jelinek-Mercer smoothing), while  $\alpha$  controls the rate at which the element's estimated representativeness increases as a function of observations.

### 5.1.5 Observations

We have created a process for estimating model nodes in inference networks from multiple representations. This process consists of three steps:

1. Choose a small set of representation functions  $F$  appropriate for the collection and retrieval task.
2. Specify how the observations from elements returned by a representation function should be combined to estimate  $\phi(f(v))$ .

3. Estimate or choose the combination parameters  $P(f(v) \xrightarrow{r} \theta_v)$ .

By the using the functions  $F = \{s, C_d\}$ ,  $\phi(f(v)) = MLE(f(v))$ , and making specific choices about  $P(f(v) \xrightarrow{r} \theta_v)$  we have shown that the model estimation techniques used in this chapter can be equivalent to the statistical language model estimation techniques presented in Section 3.4.2. These equivalences lead us to believe that the using the maximum likelihood estimator for  $\phi(f(v))$  is reasonable. The estimation process outlined here makes explicit our assumptions about the relative merit of the element text and collection text in the estimation of the element's model. This process also casts new light on the traditional estimation methods.

We reassert here that the goal of the model estimation process is one where we wish to estimate the model of queries for which the element is relevant. The first approaches used straightforward combinations of models estimated from the text of the element and the text of the entire collection. This approach makes sense when little is known about the retrieval task or characteristics of the text collection. However, in some cases, biasing the estimation process using alternative representations has been shown to be successful for various retrieval tasks. Many of these cases are described in Section 3.4.4. The model presented here formalizes this process in the inference network framework.

Knowledge of the task and corpus should guide the researcher to choose an appropriate set of functions  $F$  that represent assumptions about which related vertexes in the graph  $G$  should be used in the estimation of a language model for vertexes. This section has already introduced three functions, the "self" function  $s$ , the "document" function  $d$ , and the collection function,  $C_d$ . Table 5.1 lists these and other example functions of document and collection structure. These functions can easily be applied to many retrieval tasks.

To keep the number of parameters we must estimate to a minimum, we choose to set the representativeness of each function of  $\theta_v$  as a constant, implying that

$$\frac{P(f(v) \xrightarrow{r} \theta_v)}{Z_{F,v}} = \lambda_f. \quad (5.16)$$

This approach is equivalent to using Jelinek-Mercer smoothing to combine the representation models. While it may be possible to have a more accurate estimation method by making different model choices, they introduce additional parameters that we must estimate. The choices we have made for the model results in  $|F| - 1$  unspecified parameters (the knowledge that the  $\lambda$  values must sum to one reduces the number of parameters by one). Making different choices is likely to increase the number of unspecified parameters. By doing so, it either increases the amount of training data needed to estimate the parameters for the  $|F| - 1$  parameters or reduces the number of functions for which we can reliably estimate an effective parameters with any given training collection.

Function	Definition	Description
$s(v) =$	$\{v\}$	the vertex itself
$d(v) =$	$\{v_j \in V : v_j \text{ is the document containing } v\}$	the “document” containing the vertex
$o_{\sigma_V}(v) =$	$\{v_j \in V : v_j \text{ contains } v, \ell_V(v_j) = \sigma_V\}$	all vertexes of type $\sigma_V$ containing the vertex
$C_d(v) =$	$\{v_j \in V : v_i \in V \wedge v_j \in d(v_i)\}$	the set of “document” vertexes
$C_{\sigma_V}(v) =$	$\{v_j \in V : \ell_V(v_j) = \sigma_V\}$	all vertexes of type $\sigma_V$
$p(v) =$	$\{v_j \in V : (v_j, p, v) \in E\}$	the parent of a vertex
$l(v) =$	$\{v_j \in V : (v_j, r, v) \in E\}$	the elements linking to the vertex
$k_{\sigma_V}(v) =$	$\{v_j \in V : (v, p, v_j) \in E, \ell_V(v_j) = \sigma_V\}$	all children (kids) of type $\sigma_V$
$de_{\sigma_V}(v) =$	$\{v_j \in V : de(v_j, v), \ell_V(v_j) = \sigma_V\}$	all descendants of type $\sigma_V$
$r(v) =$	$\{v_j \in V : v \in de(v_j), p(v_j) = \emptyset\}$	root of tree containing $v$

Table 5.1: Example functions of the graph structure encoding assumptions about related structural elements. A small subset of these functions is chosen for  $F$  when constructing a retrieval model.

Through choosing to estimate  $\phi(f(v))$  using the maximum likelihood estimator, using Jelinek-Mercer to combine the representation models, one can recreate retrieval models known to be effective for a variety of tasks. For example, for known-item retrieval of web documents, a choice of

$$F = \{s, C_d, de_{\text{title}}, l\} \quad (5.17)$$

results in a  $\theta_v$  that is estimated by interpolating a maximum likelihood estimators formed from the text of the document, the collection text, the title text, and in-link text from other documents. These choices recreate the language models investigated by Kraaij et al. [38]. One can recreate the highly effective keyword rankings of Sigurbjörnsson et al. [77] for XML element retrieval by choosing

$$F = \{s, C_d, d\}, \quad (5.18)$$

using the maximum likelihood estimator for  $\phi(f(v))$ , Jelinek-Mercer smoothing, and an element length prior.

While the estimation methods presented here are on some level just a new formalism describing the smoothing approaches used in Chapter 4, the presentation here is important. It formally justifies the use of mixtures of multiple-Bernoulli models for *in-model combination* of the evidence from multiple representations. The use of the mixtures of multinomials for belief estimation in the Inference Network model used in Chapter 4, while mathematically equivalent, was not well justified for complex concepts.

The introduction of the representation layers closely parallels the choices most experimenters will make in setting up retrieval tasks, resulting in simplifications to queries

that were unnecessarily verbose for the most common use cases. This allows grouping the representation functions  $F$ , estimation choice  $\phi$ , and combination choices  $P(f \xrightarrow{r} \theta_v)$  within parameter files outside of queries, greatly clarifying the estimation process and reducing the need for excessively verbose queries to create *in-model combination*.

It's important to note that the representation layer will fill most researchers' needs, greatly simplifying query construction for them. However, we do not wish to sacrifice the power introduced by field evaluation and the #WSUM operator. There are cases where a researcher may wish to explore models where the weights on representations vary from one concept to another. For example, Petkova and Croft [66] use collection statistics to weight terms per representation. One may also wish to use a weighted list of synonyms in a field when querying semantic annotations.

## 5.2 Structure-Expressive

The *structure-aware* adaptations proposed in Section 5.1 allow for the creation and interpolation of multiple representations. However, there are situations where combining representations and keyword retrieval is not a sufficient representation of a user's information need. The user may wish to retrieve elements of a certain type, touching on *result-universality*, or have the need to specify relationships between elements and keywords.

We first motivate the need for more expressivity than keywords in the retrieval language, highlighting challenges in clearly conveying the impact of element retrieval operators to the query creators. Section 5.2.2 then proposes the #SCOPE query operator that helps makes the impact of nesting element restrictions in a query more explicit to the user. Finally, Section 5.2.3 shows an important connection between one form of the #SCOPE operator and keyword retrieval.

### 5.2.1 Motivation

We described how one can produce rankings on elements of a specific type in Section 3.5.1. It may also be desirable to nest queries of this form. For example, a user may wish to retrieve articles about suicide bombings containing images of investigators in Kabul.

Recall the XML of our sample news article (Figure 2.1), which grouped image links with their captions in an `image` element. A query for the user's information need may combine matches of the keywords `Kabul` and `investigators` in the `image` elements with the score of the `article` element matched against the keywords `suicide` and `bombings`. One way to express a query such as this in the Indri query language is



```

5.6    #AND[article] (
        suicide bombings
        #AND[image] (Kabul investigators)
    )

```

The outer `#AND[article]` is needed in order to retrieve `article` elements; without it, `image` elements would be ranked. Indri's current implementation does not specify how the beliefs of multiple matching `image` extents should be combined. The default behavior is to include the belief of each extent in the containing operator. In this example, the containing operator, `#AND[article]`, would be used, resulting in a multiplication of the beliefs and exhibiting a bias toward retrieving articles with a sole image that matches the query terms. One could wrap the `#AND[image]` query clause with other operators, such as `#OR` or `#MAX`. However, it is easy to overlook the need to specify a combination method when using extent restrictions or other structural constraints, and this is a common error when creating structured queries.

### 5.2.2 Scope Operator

Here we assert that for clarity, one should separate the context restriction to `image` elements from the `#AND` operator and replace it with a new `#SCOPE` query operator that corresponds to the inserted inference network node. In addition to specifying an extent restriction for selecting desired elements, the `#SCOPE` operator has a mandatory argument specifying how the evidence from multiple matching elements should be combined.

Table 5.2 shows available combination techniques. The  $\psi(v)$  function operates on elements. It evaluates the structural constraints on  $v$ , identifying the elements in the collection that satisfy the structural constraints specified in the scope operator for the input element  $v$ . It then returns tuples of the concept nodes constructed for the nested query with their paired vertexes in the collection graph structure.

The `#SCOPE` belief operator requires the combination method and extent restriction as qualifiers, as well as a single belief operator as its nested query clause. Query 5.6 can be written using the `#SCOPE` operator as

```

5.7    #SCOPE[result:article] ( #AND (
        suicide bombers
        #SCOPE[or:image] ( #AND ( Kabul investigators ) )
    ) ) .

```

If any other operator is the outermost query operator, document retrieval is performed by the retrieval system. When a scope operator is nested within a query, the elements matching that inner context and also contained within the element being evaluated as its outer context are the ones used and combined by the scope operator.

Method	Formula	Remarks
result	return $b(t)$ for each tuple $t \in \psi(v)$	used when #SCOPE is outermost query operator to specify retrieval unit
or	$1 - \prod_{t \in \psi(v)} (1 - b(t))$	rewards multiple matching elements
and	$\prod_{t \in \psi(v)} b(t)$	biased toward few elements matching with all having high belief
avg	$\frac{1}{ \psi(v) } \sum_{t \in \psi(v)} b(t)$	balanced average of beliefs
min	$\min_{t \in \psi(v)} b(t)$	minimum belief of matching elements
max	$\max_{t \in \psi(v)} b(t)$	maximum belief of matching elements

Table 5.2: This table describes the combination method parameter of the #SCOPE operator. The  $\psi$  function returns tuples of inference network nodes and elements satisfying the structural constraint of the scope operator for element  $v$ . The  $b(t)$  is a belief estimated using the inference network for the tuple  $t$ . For the standard element retrieval without prior probabilities,  $t = (q_i, v_j) \in \psi(v)$  and  $b(q_i, v_j) = P(q_i | \theta_{v_j})$ .

In addition to providing support for matching subqueries on elements contained within the result elements, the unified Inference Network model introduces operations for specifying constraints on the graph structure. Table 5.3 shows the structural constraints we add to the Indri query language. For example, in the query

```
5.8 #SCOPE[result:title] ( #AND(
    afghanistan
    #SCOPE[or:.\article] ( #AND( suicide bombings ) )
  ) ) ,
```

the user requests that `title` elements be ranked by how relevant the `title` element is to `afghanistan` and the `title` element's ancestor `article` elements are relevant to the query `#AND( suicide bombings )`. In this query, "outer context" of the `.\article` scope context restriction is the `title` context restriction in this query.

In many collections, there are often elements of different types that a user may wish to view as equivalent during ranking. One way to support this would be to conflate the types during indexing. However, this has the disadvantage that different users of the system may have different views about which element types are equivalent. The example article in Chapter 2 has an `intro_paragraph` element as well as several `paragraph` elements. For some information needs, it may be appropriate to treat elements of the type

Constraint	Syntax
child	#SCOPE[method:./type] (...)
descendant	#SCOPE[method:../type] (...)
parent	#SCOPE[method:.\type] (...)
ancestor	#SCOPE[method:.\type] (...)

Table 5.3: Structural constraints for nested information needs. A \* may be substituted in place of a type to express that elements of any type matching the structural constraint are permissible. The slash notation used to express structural relationships is similar to that used in XPath.

`intro_paragraph` the same as `paragraph` elements, while for other needs, it may be appropriate to retrieve only elements of the `intro_paragraph` type. To facilitate this at retrieval time the extended query language allows a list of element types in parentheses wherever a single element type is permitted. For example, the query

```
5.9    #SCOPE[result:(paragraph,intro_paragraph)] (
        #AND( suicide bombers )
    )
```

allows for retrieval of elements with the types `paragraph` or `intro_paragraph`.

### 5.2.3 Observations

There is an important relationship between the `avg` combination method and statistical language modeling. Suppose we index special `token` elements, where each token in the document is tagged within its own `token` element. Consider ranking the query

```
5.10   #SCOPE[result:document] (
        #SCOPE[avg:token] ( afghanistan )
    )
```

with  $F = \{s, C_d\}$ ,  $\phi(f(v)) = MLE(f(v))$ , and Jelinek-Mercer smoothing. The  $\psi(d)$  function returns a tuple  $(c_i, v_j)$  for every `token` element in the document  $d$ . The  $c_i$  components of the tuples are the concept nodes connected to the `afghanistan` concept and the element  $v_j$ . Then, the belief node  $I_d$  for a document  $d$  is estimated using

$$\begin{aligned}
 P(I_d | (c_i, v_j) \in \psi(d)) &= \frac{1}{|\psi(d)|} \sum_{(c_i, v_j) \in \psi(d)} P(c_i = 1 | \theta(v_j)) \\
 &= \frac{1}{|d|} \sum_{(c_i, v_j) \in \psi(d)} P(q_1 | \theta(v_j)),
 \end{aligned}
 \tag{5.19}$$

where  $q_1$  corresponds to the term `afghanistan` and  $v_j$  corresponds to a `token` element in the document. Recall that  $\psi(v)$  returns tuples of concept nodes  $c_i$  paired with their corresponding vertices  $v_j$  in the collection graph that satisfy the structural constraint specified in the query. Expanding the estimation of  $\theta_v$  using  $F$ ,

$$\begin{aligned} P(I_d | (c_i, v_j) \in \psi(d)) &= \frac{1}{|d|} \sum_{(c_i, v_j) \in \psi(d)} [\lambda_s P(q_1 | \phi(\{v_j\})) + \lambda_{C_d} P(q_1 | \phi(\{docs \in V\}))] \\ &= \frac{1}{|d|} \sum_{(c_i, v_j) \in \psi(d)} [\lambda_s P(q_1 | MLE(v_j)) + \lambda_{C_d} P(q_1 | MLE(C))]. \end{aligned} \quad (5.20)$$

By separating the two terms in the summation and simplifying, we find that

$$\begin{aligned} P(I_d | (c_i, v_j) \in \psi(d)) &= \frac{1}{|d|} \sum_{(c_i, v_j) \in \psi(d)} \lambda_s I(q_1 \in v_j) + \frac{1}{|d|} \sum_{(c_i, v_j) \in \psi(d)} \lambda_{C_d} P(q_1 | MLE(C)) \\ &= \lambda_s \frac{1}{|d|} \sum_{(c_i, v_j) \in \psi(d)} I(q_1 \in v_j) + \lambda_{C_d} P(q_1 | MLE(C)) \\ &= \lambda_s P(q_1 | MLE(d)) + \lambda_{C_d} P(q_1 | MLE(C)), \end{aligned} \quad (5.21)$$

which is equivalent to belief estimation for the query

```
5.11 #SCOPE[result:document] ( afghanistan ) .
```

This example demonstrates that the use of the `avg` belief combination process behaves similarly to how statistical language modeling techniques combine the evidence of multiple matching terms when constructing a language model. Nevertheless, it is not clear that the combination methods that work well when estimating beliefs for terms will work well when combining the evidence of multiple elements matching an extent restriction; we investigate this problem experimentally in Chapter 6.

### 5.3 Result-Universal

The previous section described how the model can be extended to specify ranking of arbitrary elements. Adding the ability to specify a retrieval unit is perhaps the most important component to having a *result-universal* retrieval model. However, the addition of this functionality highlights the importance of the common need to bias the rankings of retrieved elements based on properties of the elements. This need arises because the biases present in the unified Inference Network model may be different from those of relevant elements. As such, this section focuses on the use of priors to improve result-universality.

### 5.3.1 Motivation

In previous work, priors have been applied in two use cases. In the first use case, the researcher has knowledge that certain document or element types are more likely to be relevant and there is no inherent bias by the ranking algorithm with respect to these different types. An example of this use case is the use of URL type priors in homepage or known-item finding of Web pages. The other use case is one where in addition to knowledge that some document or element types are more likely to be relevant, there is a known bias of the retrieval algorithm. An example of this use case is the use of element length priors to bias results toward retrieving longer elements.

Length biases in retrieval model are a well-known phenomena, dating back at least to the work of Singhal et al in the vector-space retrieval model [81]. This problem is even more pronounced in element retrieval because the retrieval system may need to estimate the relevance of very short elements as well as very long elements for a single query. The length bias results from properties of the model estimation. When using Jelinek-Mercer smoothing of maximum likelihood estimates, language modeling systems have a bias toward ranking short elements highly [34]. Smoothing using Dirichlet-priors and two-stage smoothing are not as strongly biased toward ranking short elements highly, but their biases may still not match our expected distribution of which elements are relevant.

Previous work with the Indri retrieval system does support a prior probability operator in its query language (Table 3.3, [55]), but places few restrictions on its use. For example, one could write

```
5.12  #OR( #PRIOR(length) afghanistan )
```

which would request that either the document be estimated relevant using the “length” prior or that it be relevant to the query “afghanistan.” We do not feel that this is a typical or appropriate use of prior probabilities in the Inference Network model; we have difficulty imagining an information need where this behavior is desirable. The unrestricted use of priors in the Inference Network model is confusing and does not guide the user about when priors are appropriate and how they should be used in queries.

### 5.3.2 Prior Probabilities Belong on Scope Operations

The application of priors in the statistical language models (Section 3.4.1) combines the prior probabilities with the probability of the query given the document observation. These #PRIOR probabilities are applied only to the scores of elements being ranked according to some query or subquery with an element restriction. This is exactly where the #SCOPE operators apply; it is appropriate to attach our choice of prior probabilities to the #SCOPE operator, such as in the query

```
5.13  #SCOPE[result:*:length]( #AND( suicide bombings ) ) .
```

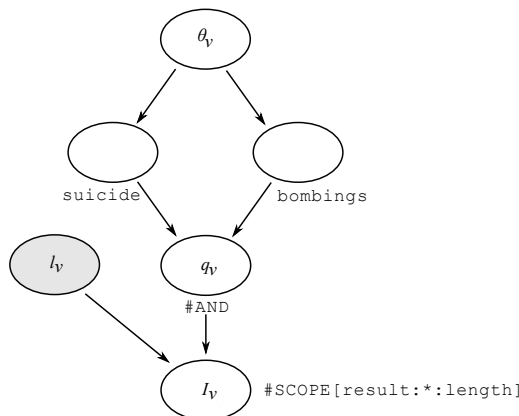


Figure 5.3: A partial inference network for query 5.13 which uses an element length prior.

When the prior probability is used in a result `#SCOPE` operator, the belief of relevance for the element is directly multiplied by the prior probability estimate. Figure 5.3 shows a partial inference network for Query 5.13.

When we have a nested scope operator, it would be appropriate to apply priors that adjust biases inherent in the retrieval algorithm to the inner operators, such as in the query

```
5.14 #SCOPE[result:article] (
      #SCOPE[or:paragraph:length] ( #AND(suicide bombings) )
    ) .
```

Figure 5.4 shows a partial inference network for this query. The combination methods in Table 5.2 are extended to support the priors by choosing  $b(p_i, q_i, v_j) = P(p_i = 1) \cdot P(q_i | \theta_{v_j})$ , where the  $(p_i, q_i, v_j) \in \psi(v)$  tuples have  $v_j$  matching the structural or extent restriction for  $v$ ,  $p_i$  as the prior node for element  $v_j$ , and  $q_i$  as the nested query node for element  $v_j$ .

Returning to our two use cases, we see that it makes sense to apply priors used to adjust belief estimation biases, notably length-based priors, directly to the `#SCOPE` operators that contain belief estimation for query terms. This is because we are adjusting the belief estimates resulting from properties of these elements. For the first type of prior, when we have knowledge of the relevance of result elements, it makes sense to apply the priors to the `#SCOPE[result...]` operator. Restricting placement of prior probabilities of relevance on scope operations is also consistent with our theme of clarifying the use query language.

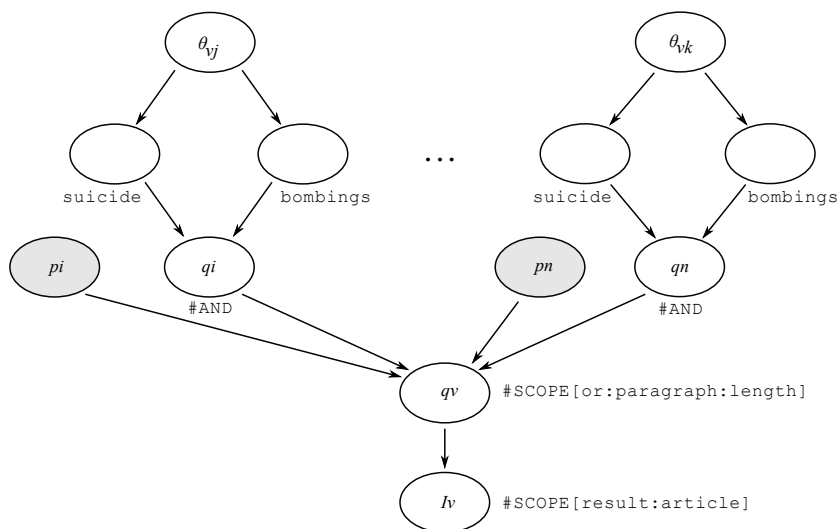


Figure 5.4: A partial inference network for Query 5.14 which uses an element length prior in a nested #SCOPE operator.

## 5.4 Annotation-Robust

An *annotation-robust* retrieval model should address issues surrounding both queries that may use annotation constraints and the types of errors an annotation tool may make. Section 1.3 described how a question answering system may make use of annotations to perform retrieval of linguistic structures annotated in a text corpus.

Recall query 1.10, which requested that the system return `target` predicates having a child `arg1` element about suicide bombers also having an unspecified child `argm-loc` element:

```
1.10 RETURN target MATCH trained AND (suicide bombers) IN
      ./arg1 AND HAS ./argm-loc .
```

On the surface, it may seem that the unified Inference Network model only needs an extension to the #ANY operator permitting structural constraints:

```
5.15 #SCOPE[result:target]( #AND(
      trained
      #SCOPE[and:./arg1]( #AND( suicide bombers ) )
      #ANY:./argm-loc
    ) ) .
```

However, we feel there are still unaddressed issues. Automated annotation tools may make errors through incorrect annotation boundaries, incorrect annotation labels, or in-

correct structure. Furthermore, automated annotation tools may produce confidence scores which may help the search engine provide better rankings.

### 5.4.1 Annotation Boundary Errors

Prior treatment of the model estimates each representation node from observations strictly contained within the element or annotation boundaries. Similarly, the elements considered in a nested `#SCOPE` operator with a containment restriction must be entirely contained within the outer element. This makes sense for XML elements, but may fail to perform well when retrieving annotations. Consider searching for predicates where President Ulysses Grant vetoed bills. A query for this information need might look like

```
5.16  #SCOPE[result:target (
        veto
        #SCOPE[and:./arg0] (
            #SCOPE[and:person] ( president ulysses grant )
        )
    ) .
```

In this query, the user requests that the `person` annotation be contained within the `arg0` annotation. Consider some example annotations of an answer-bearing sentence:

```
5.17  In 1874, President [PERSON Ulysses Grant] vetoed the
      Inflation Bill.
```

```
5.18  In 1874, [ARG0 President Ulysses] Grant [TARGET vetoed]
      [ARG1 the Inflation Bill].
```

Due to errors in the annotation process, the `person` annotation overlaps with but is not contained in the `arg0` annotation. Furthermore, not all of the query terms are contained within the `person` annotation, but they are located near the annotation boundary.

We handle the first problem by generalizing the  $\psi$  function to support partial containment for nested `#SCOPE` operators. We handle the second problem by allowing probabilistic observations of concepts when estimating the representation models.

### Probabilistic Containment

We have already shown how the combination of beliefs in the `#SCOPE` operator can be generalized for prior probabilities (Section 5.3). This section further generalizes the belief estimation and combination of the `#SCOPE` operator.

We do so by relaxing the containment restriction to allow overlapping elements and weighting  $b(t)$  by the probability that a randomly observed concepts from the inner element is one that is also present in the outer element. One way to estimate this probability



is

$$P(c \in v | c \in v_i) = \sum_{c \in v_i} \frac{I(c \in v)}{|v_i|} \quad (5.22)$$

where  $c$  is an observation vector anchored in the text of  $v_i$ . That is, we may estimate the probability that one element is contained within another, such as the probability that the person annotation of 5.17 ( $v$ ) is contained within the `arg0` annotation in 5.18 ( $v_i$ ). The model incorporates this estimate into the `#SCOPE` belief operator using

$$b(p_i, q_i, v_j) = P(c \in v | c \in v_i) \cdot P(p_i = 1) \cdot P(q_j | \theta_j) \quad (5.23)$$

and modifying  $\psi(v)$  to return all tuples  $(p_i, q_i, v_j)$  where  $P(c \in v | c \in v_i) > 0$ . This approach permits overlapping elements to contribute to the belief estimation process while penalizing elements  $v_i$  not completely contained within the element  $v$ . We justify the inclusion of  $P(c \in v | c \in v_i)$  here by remarking that this probability was implicitly present in the prior stated belief estimation and combination process; when evaluating  $\phi$ , we were implicitly asking the question “what is the probability that element  $v_i$  is contained in element  $v$ ?” In cases of explicitly coded structure, the probability of containment is known to be one or zero, so it was natural to overlook this in belief estimation.

When applying Equation 5.22 to estimating the beliefs in the inference network for Query 5.16, the probability that the `person` annotation in 5.17 is contained within the `arg0` annotation of 5.18 is

$$P(c \in v | c \in v_i) = 1/2 = 0.5. \quad (5.24)$$

This example illustrates how the inclusion of  $P(c \in v | c \in v_i)$  in the `#SCOPE` belief estimation process improves the robustness of the model to annotation boundary errors.

### Probabilistic Observations

We can provide additional robustness by allowing concept observation vectors outside but near the boundary of the annotation to influence our estimate for the representation nodes. As we do not believe these observations are as likely to be as good observations as those within the annotation, we introduce the notion of uncertain observations by allowing a probability to be attached to an observation.

$$P(r_i | \phi(f(v))) = \frac{\sum_{c:c_i=1} P(c \in f(v))}{\sum_{c'} P(c' \in f(v))}. \quad (5.25)$$

The numerator sums up the probabilities of observation where the term or feature corresponding to the  $i^{\text{th}}$  dimension in the observations. The denominator sums this probability over all observations to get a valid multiple-Bernoulli probability distribution. Setting  $P(c \in v) = 1$  if the observed concept vector  $c$  is contained within the annotation  $v$ 's

boundaries and 0 otherwise results the maximum likelihood estimate in Equation 5.5. However, through different choices of  $P(c \in f(v))$  we can achieve different behavior. For example, we may wish to set  $P(c \in f(v)) = 0.5$  for the observations just one token outside the annotation boundaries of elements in  $f(v)$ .

Consider estimating the representation model for the function  $f = s$  and the person annotation in 5.17. Let us choose  $P(c \in f(v)) = 0.5$  for concept vectors observed immediately adjacent to the boundaries of the annotations and  $P(c \in f(v)) = 1$  for concept vectors observed inside the boundaries of the annotations returned by  $f(v)$ . Then for  $r_i$  corresponding to the concept vector for the token `ulysses`, the model estimates

$$P(r_i | \phi(s(v))) = \frac{1}{0.5 + 1 + 1 + 0.5} \approx 0.333. \quad (5.26)$$

For  $r_j$  corresponding to the concept vector for the token `president`, the model estimates

$$P(r_j | \phi(s(v))) = \frac{0.5}{0.5 + 1 + 1 + 0.5} \approx 0.167. \quad (5.27)$$

The example illustrates how probabilistic observations of concept vectors can improve the robustness of the model to annotation boundary errors when estimating representation models.

## 5.4.2 Annotator Confidence Estimates

If the annotation tool provides confidence estimates on the correctness of an output annotation, then it is natural to believe that use of these estimates ought to be useful in ranking when using annotations as a part of the information need. For example, consider an annotator that outputs a probability distribution over the label types for each annotation. These distributions can be used in the  $b(t)$  belief in the `#SCOPE` operator by estimating

$$b(p_i, q_i, v_j, \sigma) = P(\ell(v_j) = \sigma) \cdot P(c \in v | c \in v_i) \cdot P(p_i = 1) \cdot P(q_i | \theta_{v_j}). \quad (5.28)$$

where  $P(\ell(v_j) = \sigma)$  is the annotator's confidence estimate of the label for the annotation. Our justification for the use of  $P(\ell(v_j) = \sigma)$  is similar to that of the probability of containment. We are already implicitly asking what the probability of label match is when using explicitly coded structure, but because the label is known, this probability is always zero or one. In the case of annotations, the label value may be an estimate, and it is natural to include the estimation in this way.

## 5.5 Summary

This chapter formally connected the mixtures of statistical language models to the Inference Network model using mixtures of multiple Bernoullis. This is achieved through the

introduction of the representation layer and the creation of representations via functions of the graph structure. The representation layer, akin to Turtle’s text representation layer, formalizes the process of selecting text representations. This makes the modeling choices researchers regularly make explicit.

The chapter also addresses the confusing nature of extent restrictions, which in prior work could be attached to any belief query operator. The retrieval model’s behavior in prior work defaulted to treating this as an expansion operation, simply duplicating the query nodes within the inference network. Without intimate knowledge of the inference network model and this behavior, writing effective queries with extent restrictions required luck. We addressed this confusing nature by introducing the #SCOPE operator, which explicitly calls out the extent restriction process and forces the query writer to think about and choose an appropriate method for combining the beliefs from multiple matching elements.

Finally, this chapter discusses how the Inference Network model may be extended to be more *annotation-robust*. We did this by proposing an alternative approach to the estimation representation language models, generalizing the notion of extent containment, and proposing an approach to incorporating annotator confidence estimates.

This extended Inference Network model can elegantly support a greater range of retrieval tasks than before. However, this greater flexibility results in a number of model choices that a researcher must make.

- What is the best estimator for the representation models  $\phi(f(v))$ ? We have argued that the maximum likelihood estimator is well-suited to this task.
- Which set of representation functions  $F$  is most appropriate for the retrieval task? This will depend on the collection’s structure and the nature of the retrieval task, but there may be noteworthy trends across tasks and collections. This task is typically not difficult; researchers make this choice regularly. For example, for a web search system it is common to choose the title text, web page text, the text of in-links as representations, and use a collection language model for smoothing. This choice is expressed by setting  $F = \{s, de_{\text{title}}, l, C_d\}$ .
- How should the  $\lambda_f$  parameters be chosen? We argued for the use of fixed parameters through Jelinek-Mercer smoothing and directly optimizing an evaluation measure using a training collection. The choice of fixed parameters results in only  $|F| - 1$  linear combination parameters that must be estimated. Furthermore, while smoothing using Dirichlet priors and two-stage smoothing is straightforward for smoothing when  $F = \{s, C_d\}$ , these methods do not generalize easily to more parameters. The use of fixed parameters is common practice for when  $|F| > 2$ ; we will use the same approach here.

- Which, if any, priors should be used during retrieval? As with other choices, this will depend on the collection and retrieval task. It may also depend upon the choice of the  $\lambda_f$  parameters and the information need.
- If working with structured queries, which combination methods should be used with #SCOPE operators? The choice of combination method can have a profound impact on ranking. We have shown how the `avg` combination method can behave similarly to term ranking methods, but have not investigated empirically which combination methods work well for which query types.
- Finally, we proposed methods for robustly handling the errors of imperfect annotators. We must investigate whether these extensions improve the robustness of the retrieval model.

Throughout the chapter we made a concerted effort to clarify use of the query language and its implications on the structure of the inference networks. The previous adaptations of language modeling to the Inference Network model required a mixture of placing smoothing parameters in the query language and parameter files in order to reconstruct mixtures of multiple representations. This chapter greatly simplifies this process by allowing the specification of representations and the smoothing choices to be expressed solely in parameter files outside of the query. These simplifications reduce the complexity of creating good queries.

Similarly, the impacts of nested extent restrictions in Indri queries is not obvious and can be confusing. We proposed the introduction of the #SCOPE operator to clarify the process of extent restriction, proposing a required argument specifying belief combination and restricting prior probabilities of relevance to placement on these #SCOPE operators.

The following chapters revisit the experiments on known-item finding and element retrieval to investigate the validity of the maximum likelihood estimator for  $\phi(f(v))$ , the choice of  $F$  for these tasks, and the choice of combination methods for #SCOPE operators.

# Experimental Results

This chapter presents direct evaluation of the unified retrieval model presented in Chapter 5, comparing it to the task-oriented retrieval models presented and evaluated in Chapter 4. In addition to verifying that our adaptations to the Inference Network model result in a system with strong performance for these tasks, we use these experiments to verify and investigate the model with respect to some of the properties we previously argued as important for retrieval of documents with structure and annotations. We investigate the *structure-aware* property in known-item and element retrieval experiments through our choice of representation functions (Sections 6.1 and 6.2). Our experiments with element retrieval in Section 6.2 investigate the *result-universal* property. Section 6.2 also contains a thorough investigation of the use of the inference network and #SCOPE combination methods, exploring the *structure-expressive* property. Finally, we explore how *annotation-robust* the model is through the retrieval of answer-bearing sentences (Section 6.3).

## 6.1 Known-Item Finding

The role of the known-item finding experiments in this chapter serve largely as a verification of the model’s performance relative to the task-oriented mixture model presented for the task in Section 4.1. These experiments touch on the model’s ability to be *structure-aware* and the stability of estimated parameters.

We investigate a small number of functions to select related language models in these experiments. We chose the functions in Table 6.1. This specific model is instantiated by choosing  $F = \{d, l, de_{\text{title}}, de_{\text{header}}, de_{\text{meta}}, C_d\}$ ,  $\phi(f(v)) = MLE(f(v))$ , and constant  $\lambda_f$  values across all documents. This is equivalent to using Jelinek-Mercer smoothing to combine the representation language models. The evaluation collections we use are the same as those in Section 4.1.1. We convert the topics’ keyword queries into queries of the form

```
6.1 #SCOPE[result:document:url] (  
    #AND( term1 term2 ... termN )  
)
```

using the same URL-type based prior as used in Section 4.1.

Label	Function	Description
document	$d$	Text of the web page
link	$l$	In-Link text
title	$d_{\text{title}}$	Titles in the document
header	$d_{\text{header}}$	Text included in $h_n$ tags ( $h_1, h_2, h_3, \dots$ )
meta	$d_{\text{meta}}$	Text of meta keywords and descriptions
collection	$C_d$	The collection model

Table 6.1: Functions of graph structure chosen in known-item finding experiments.

This model differs from the task-oriented model presented in Section 4.1 in two ways. First, we use add the `header` and `meta` representations. Second, the model in Section 4.1 used a linear interpolation of models estimated by smoothing the representation with a representation type specific collection model using Dirichlet priors. The model in this section drops the Dirichlet prior smoothing in favor of using a single collection model with a fixed weight in the linear interpolation. This simplification allows us to include estimating the collection’s weight during the grid search process. Doing so with the task-oriented model in Section 4.1 would require the estimation of an extra parameter for each representation type specific collection model. With 10 steps per parameter, 5 interpolation parameters and 5 Dirichlet prior parameters, a grid search for estimating the parameters of the task-oriented model would require the evaluation of over 161 million parameter combinations. Instead, the simplifications used in this section result in a tractable parameter estimation process.

To estimate the parameters for the Inference Network model, we performed a grid search with a step size of 0.1 per  $\lambda$  parameter, resulting in 11 possible values for each of the 6 parameters and a total of 3003 parameter combinations in the grid search. Figures 6.1 and 6.2 show the best attained MRR when setting the  $\lambda$  parameter for one of the representations constant. The figures are evocative of those in Section 4.2.3, and we use this as evidence that this parameter sweep identifies sufficiently good parameters. Table 6.2 shows the parameter estimates for these testbeds with a 95% bootstrap confidence interval. Given the relative flatness of the curves in Figures 6.1 and 6.2, we expect that the parameter estimates will transfer well from the training folds to the test folds. The results of the experiments are shown in Table 6.3. For the test results we use the parameters estimated from the corresponding training topics. For example, we use the `t10ep` samples parameters and priors for test evaluation on the `t10ep` official testbed and we used the `t13mi` maximized parameters and priors for test evaluation on the `t12ki` testbed.

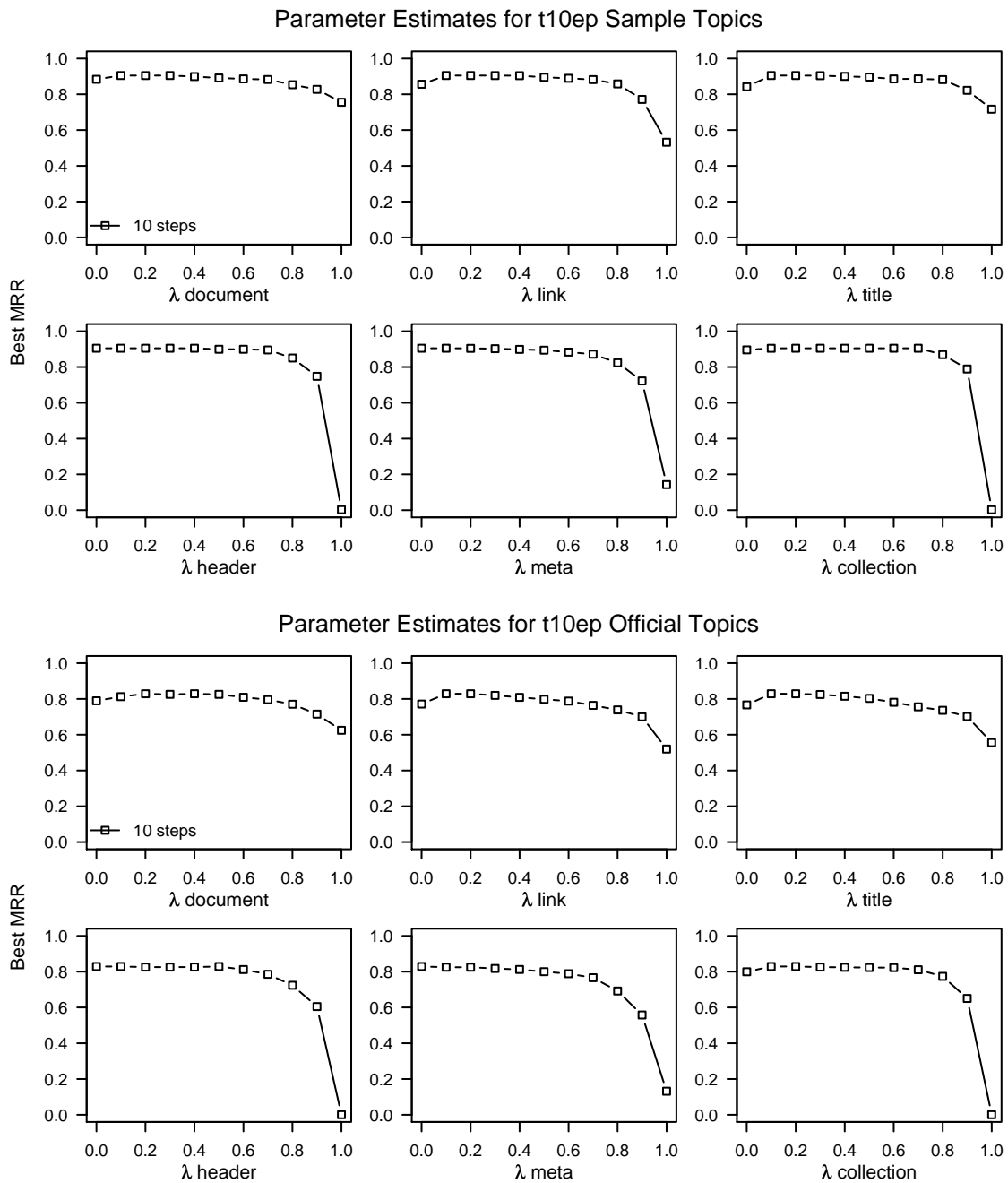


Figure 6.1: The results of the grid search on the t10ep sample and t10ep official topics.

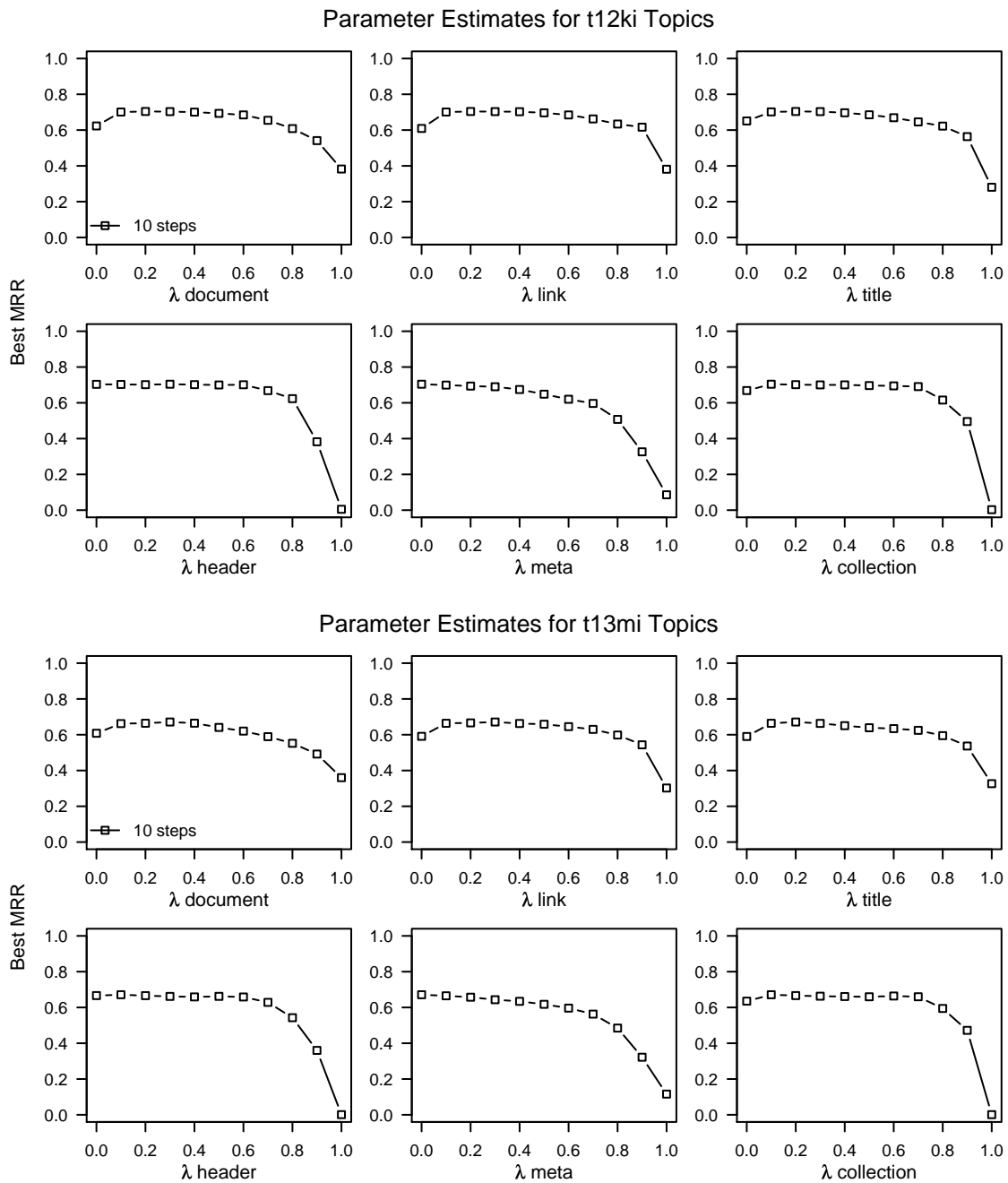


Figure 6.2: The results of the grid search on the t12ki and t13mi topics.



Parameter	t10ep samp.	t10ep off.	t12ki	t13mi
document	0.3 (0.1, 0.6)	0.4 (0.2, 0.5)	0.2 (0.1, 0.4)	0.3 (0.1, 0.3)
link	0.2 (0.1, 0.5)	0.2 (0.1, 0.2)	0.2 (0.2, 0.4)	0.3 (0.1, 0.5)
title	0.2 (0.1, 0.7)	0.2 (0.1, 0.3)	0.2 (0.1, 0.3)	0.2 (0.1, 0.3)
header	0.1 (0.0, 0.4)	0.0 (0.0, 0.5)	0.3 (0.0, 0.4)	0.1 (0.0, 0.4)
meta	0.0 (0.0, 0.0)	0.0 (0.0, 0.2)	0.0 (0.0, 0.1)	0.0 (0.0, 0.2)
collection	0.2 (0.0, 0.4)	0.2 (0.1, 0.5)	0.1 (0.1, 0.3)	0.1 (0.1, 0.6)

Table 6.2: Estimated parameters for the unified retrieval model on the task of known-item finding. A 95% bootstrap estimated confidence interval for each parameter is shown in parentheses.

	Combination Method	t10ep samp.	t10ep off.	t12ki	t13mi
Task-Oriented	Equal $\lambda_r$ + train priors	0.888	0.799	0.707	0.673
	Equal $\lambda_r$ + test priors	0.890	0.813	0.706	0.666
	MRR train $\lambda_r$ + priors	0.889	0.803	0.707	0.664
	MRR test $\lambda_r$ + priors	0.890	0.813	0.701	0.665
Inference Network	Train $\lambda_r$ + priors	0.905	0.829	0.704	0.671
	Test $\lambda_r$ + priors	0.891	0.821	0.702	0.650

Table 6.3: Performance on known-item finding (MRR). The top 4 rows of results are duplicated from Table 4.3 for convenience, while bottom two rows show results using the Inference Network model proposed in Chapter 5.

Label	Function	Description
self	s	Text of the element
collection	$C_d$	The collection model
document	d	The text of the document
titles	$de_{st}$	Text of section titles contained within the element
fig	$de_{fgc}$	The text of figure captions contained within the element

Table 6.4: Functions of graph structure chosen in element retrieval experiments.

When we compare these results to those provided by our task-oriented model presented in Section 4.1, we see that these two systems behave quite similarly. The performance of the model examined here is slightly stronger than our task-oriented model on the t10ep testbeds, but slightly worse than the task-oriented model on the t12ki and t13mi testbeds. However, none of the differences are statistically significantly different. While the experiments in this section are brief, they do verify that the model satisfies the *structure-aware* property. The estimated parameters seem relatively stable from one query set to another, and the performance of the model is as strong as the relatively strong task-oriented retrieval model presented Section 4.1.

## 6.2 Element Retrieval

In this section we apply the retrieval model to the task of element retrieval using the same reference collections as in Section 4.2. The experiments around the keyword topics exist mostly to verify the model’s performance on the task, while the experiments surrounding the structured topics explore the role of scope combination. We choose model parameters of  $F = \{s, C_d, d, de_{st}, de_{fgc}\}$ ,  $\phi(f(v)) = MLE(f(v))$ , and constant  $\lambda_f$  values across all documents. Table 6.4 describes the functions in  $F$ .

Our handling of phrasal constraints, ‘+’, and ‘-’ in keyword sections of queries remains the same as in the task-oriented model presented for element retrieval in Section 4.2; we discard phrasal constraints, ‘+’ characters in front of terms or phrases, and drop the terms or phrases prefixed by a ‘-’ character from the query. The primary differences in the model are the addition of the titles and fig representations and the use of the #SCOPE operator in place of Indri’s extent retrieval.

Our separation of extent retrieval and structural constraints from belief operators into the newly introduced #SCOPE operator proposed in Section 5.2 require a different conversion of NEXI queries. To keep our results as comparable to the experiments with our task-oriented baseline, we will apply a similarly literal conversion of the NEXI queries.

We use the same example queries for conversion as those in Section 4.2. The first example query, which requested paragraphs about suicide bombings,

```
4.1 //paragraph[about(., suicide bombings)]
```

was previously converted into

```
4.2 #AND[paragraph]( suicide bombings )
```

With our query language changes, this query is instead written

```
6.2 #SCOPE[result:paragraph:prior]( #AND( suicide bombings ) )
```

where *prior* is the name of a prior estimation method. The query requesting articles having paragraphs about suicide bombings,

```
4.3 //article[about(./paragraph, suicide bombings)]
```

was previously converted into

```
4.4 #AND[article]( #AND[paragraph]( suicide bombings ) )
```

but is now written

```
6.3 #SCOPE[result:article:prior](
      #SCOPE[method:paragraph:prior](
        #AND( suicide bombings )
      )
    )
```

where *method* is one of the scope combination methods described in Table 5.2. Our improved support for structural constraints in the query language eliminates the need for the #CONTEXT operator introduced for the conversion of the query

```
4.5 //article[about(., terrorism)]
      //paragraph[about(., suicide bombings)]
```

This NEXI query was translated into the query

```
4.6 #CONTEXT[article](
      #AND[paragraph]( suicide bombings )
      #AND( terrorism ) )
```

However, this query is now written more clearly as

```
6.4 #SCOPE[result:paragraph:prior](
      #AND(
        #AND( suicide bombings )
        #SCOPE[method:.\article:prior]( terrorism )
      )
    )
```

With this extended support for structural constraints in the query language, we could have more precisely expressed Query 6.3 as

```
6.5    #SCOPE[result:article:prior] (
        #SCOPE[method:./paragraph:prior] (
            #AND( suicide bombings )
        )
    )
```

However, given the strictly hierarchal structure corresponding directly containment of text within the INEX collection, these queries will result in identical results.

In summary, the adaptations to the Inference Network model proposed in Chapter 5 have greatly clarified the query construction process. The #SCOPE operator makes explicit the process of combining evidence from multiple elements that match a structural constraint, clarifying to the user the need to make this choice. The enhanced support for structural operations in the #SCOPE operator facilitate the removal of the awkward #CONTEXT operator that was introduced in Section 4.2, and the inclusion of priors on the #SCOPE operator allows the explicit invocation of an element length prior. In the approach used in Section 4.2, we had to modify Indri source code to make use of an element length prior.

For the purposes of comparison, it may be illustrative to characterize the task-oriented model presented in Section 4.2 in terms of our extended Inference Network model. That task-oriented model is mathematically equivalent to choosing  $F = \{s, C_d, d\}$ ,  $\phi(f(v)) = MLE(f(v))$ , and constant  $\lambda_f$  values across all documents. The query conversion method used for the task-oriented model is equivalent to the conversion described above with a length prior on the result scope operator and using the MAX method of scope combination with no length prior for nested scope operators. The differences explored in this chapter include the addition of more representation functions and the exploration of different combination methods for nested scope operators.

## 6.2.1 Results

We employ a grid search with increments of 0.1 per  $\lambda_f$  parameter (for the five representation functions in Table 6.4) over the range [0, 1] and 0.3 for the length prior parameter over the range [0, 3], resulting in 11 possible values per parameter. This results in a total of 11,011 different valid combinations of parameters.

Table 6.5 shows the parameter estimates for keyword queries on the i2004k and i2005k testbeds. The similarity of the parameter estimates on the two test collections suggest that the two testbeds are relatively representative of each other. Figure 6.3 shows the parameter estimate curves for the two testbeds. Given the estimated values of 0 for the

Parameter	i2004k	i2005k
self	0.1 (0.1, 0.3)	0.3 (0.1, 0.4)
collection	0.7 (0.4, 0.7)	0.3 (0.1, 0.6)
document	0.2 (0.2, 0.3)	0.4 (0.2, 0.6)
fig	0.0 (0.0, 0.1)	0.0 (0.0, 0.2)
titles	0.0 (0.0, 0.0)	0.0 (0.0, 0.1)
length	0.9 (0.9, 1.2)	1.2 (0.9, 1.5)

Table 6.5: Parameter estimates maximizing MAP for the keyword queries. The grid search used 10 steps per parameter, resulting in increments of 0.1 for the mixture model parameters and 0.3 for the length prior parameter. A 95% confidence interval estimated using bootstrap resampling of topics is shown in parentheses.

	Combination Method	i2004k	i2005k
Task-Oriented	Train $\lambda_r$ + priors	0.240	0.116
	Test $\lambda_r$ + priors	0.235	0.113
Inference Network	Train $\lambda_r$ + priors	0.239	0.116
	Test $\lambda_r$ + priors	0.234	0.112

Table 6.6: Performance of the element retrieval model on keyword queries (MAP).

fig and titles representation functions, we do not expect very different results from those presented in Section 4.2.3. Indeed, Table 6.6 shows that results are nearly identical.

Table 6.7 shows the estimated parameters resulting from the grid search on the i2003 and i2004s testbeds for each of the scope combination methods. We do see some difference across the methods. Figures 6.4 - 6.8 show the corresponding plots for the best possible MAP achievable for any given parameter. While the shape of these plots do not vary wildly from one method to another, some of the combination methods do result in higher MAP than others.

Table 6.8 shows the MAP of the results on the i2003s and i2004s testbeds when using training and testing parameters for each of the scope combination methods. Focusing first on the training data, we would choose the MIN combination method from the i2003s training data to use for test evaluation on the i2004s testbed. When examining the i2004s training data, we would choose the AVG combination method for test evaluation on the i2003s testbed.

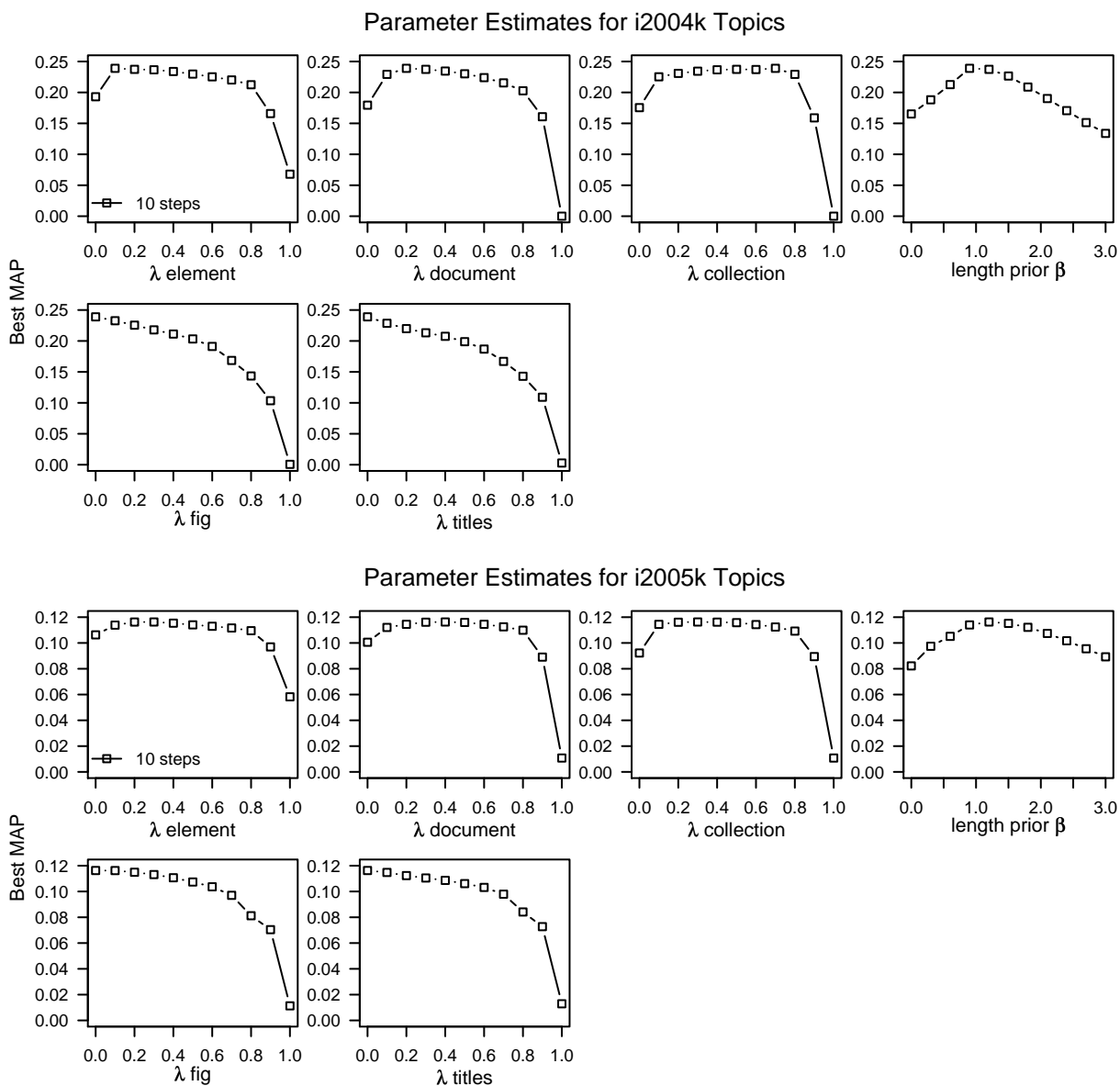


Figure 6.3: The results of the grid search testing 10 steps per parameter on the i2004k and i2005k topics. The lines show the best found MAP given the value for a specific parameter.

i2003s					
Parameter	AND	AVG	MAX	MIN	OR
self	0.4 (0.2, 0.7)	0.3 (0.1, 0.7)	0.4 (0.0, 0.7)	0.4 (0.1, 0.7)	0.3 (0.1, 0.7)
collection	0.0 (0.0, 0.5)	0.2 (0.1, 0.6)	0.2 (0.1, 0.5)	0.2 (0.1, 0.6)	0.2 (0.1, 0.5)
document	0.5 (0.1, 0.7)	0.5 (0.2, 0.6)	0.4 (0.2, 0.8)	0.4 (0.1, 0.5)	0.5 (0.2, 0.7)
fig	0.0 (0.0, 0.0)	0.0 (0.0, 0.0)	0.0 (0.0, 0.1)	0.0 (0.0, 0.0)	0.0 (0.0, 0.0)
titles	0.1 (0.0, 0.1)	0.0 (0.0, 0.1)	0.0 (0.0, 0.1)	0.0 (0.0, 0.1)	0.0 (0.0, 0.1)
length	0.9 (0.9, 1.2)	0.9 (0.6, 1.2)	1.2 (0.6, 1.2)	1.2 (0.9, 1.8)	0.9 (0.6, 1.2)

i2004s					
Parameter	AND	AVG	MAX	MIN	OR
self	0.5 (0.3, 0.8)	0.5 (0.3, 0.8)	0.5 (0.3, 0.8)	0.5 (0.2, 0.7)	0.5 (0.3, 0.8)
collection	0.2 (0.0, 0.4)	0.2 (0.1, 0.4)	0.2 (0.1, 0.4)	0.2 (0.1, 0.4)	0.2 (0.1, 0.4)
document	0.3 (0.0, 0.5)	0.3 (0.0, 0.5)	0.3 (0.0, 0.5)	0.3 (0.2, 0.5)	0.3 (0.0, 0.5)
fig	0.0 (0.0, 0.0)	0.0 (0.0, 0.0)	0.0 (0.0, 0.0)	0.0 (0.0, 0.0)	0.0 (0.0, 0.0)
titles	0.0 (0.0, 0.1)	0.0 (0.0, 0.1)	0.0 (0.0, 0.1)	0.0 (0.0, 0.1)	0.0 (0.0, 0.1)
length	0.9 (0.6, 1.2)	0.9 (0.6, 1.2)	0.9 (0.6, 1.2)	0.9 (0.6, 1.2)	0.9 (0.6, 1.2)

Table 6.7: Parameters for the structured queries estimated on the i2003s testbed (above) and the i2004s testbed (below).

		Combination Method	i2003s		i2004s	
			train	test	train	test
Task-Oriented	{	(MAX)	0.387	0.384	0.286	0.280
		AND	0.282	0.273	0.224	0.174
Inference Network	{	AVG	0.403	0.401	0.294	0.290
		MAX	0.386	0.384	0.286	0.280
		MIN	0.407	0.403	0.291	0.285
		OR	0.403	0.400	0.290	0.284

Table 6.8: Performance of the element retrieval model on structured queries (MAP).

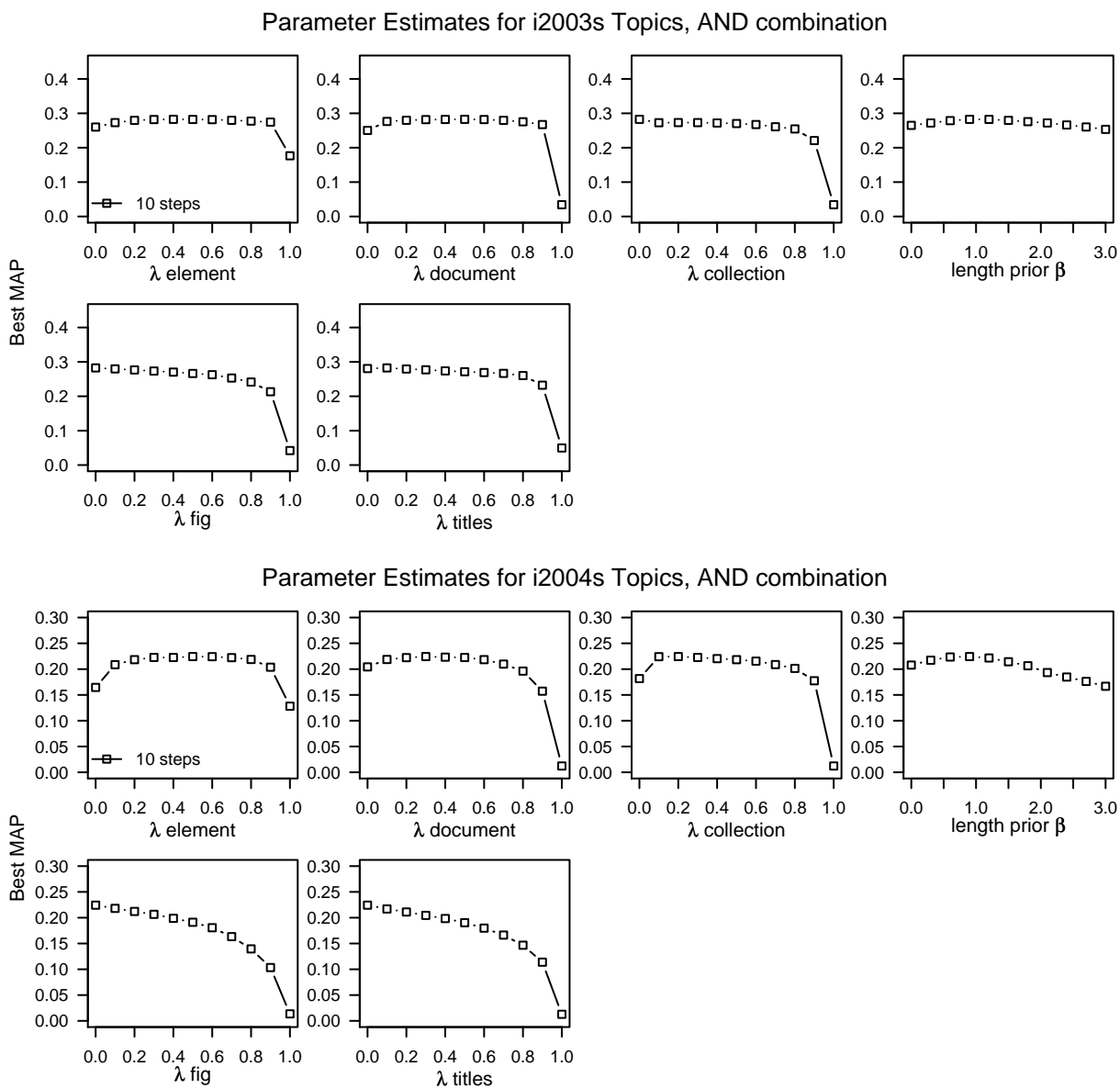


Figure 6.4: The results of the grid search testing 10 steps per parameter for the AND combination method.



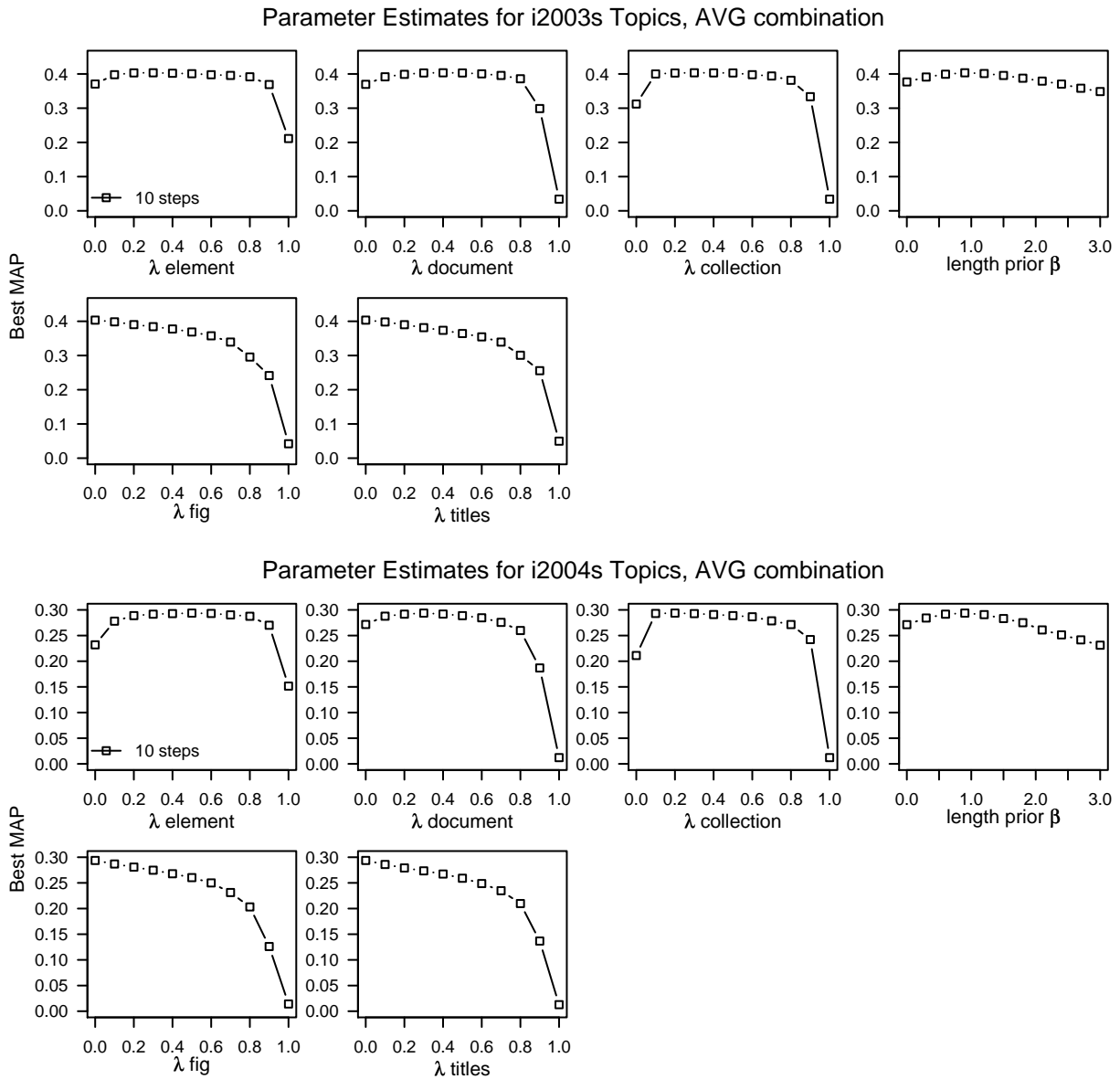


Figure 6.5: The results of the grid search testing 10 steps per parameter for the AVG combination method.

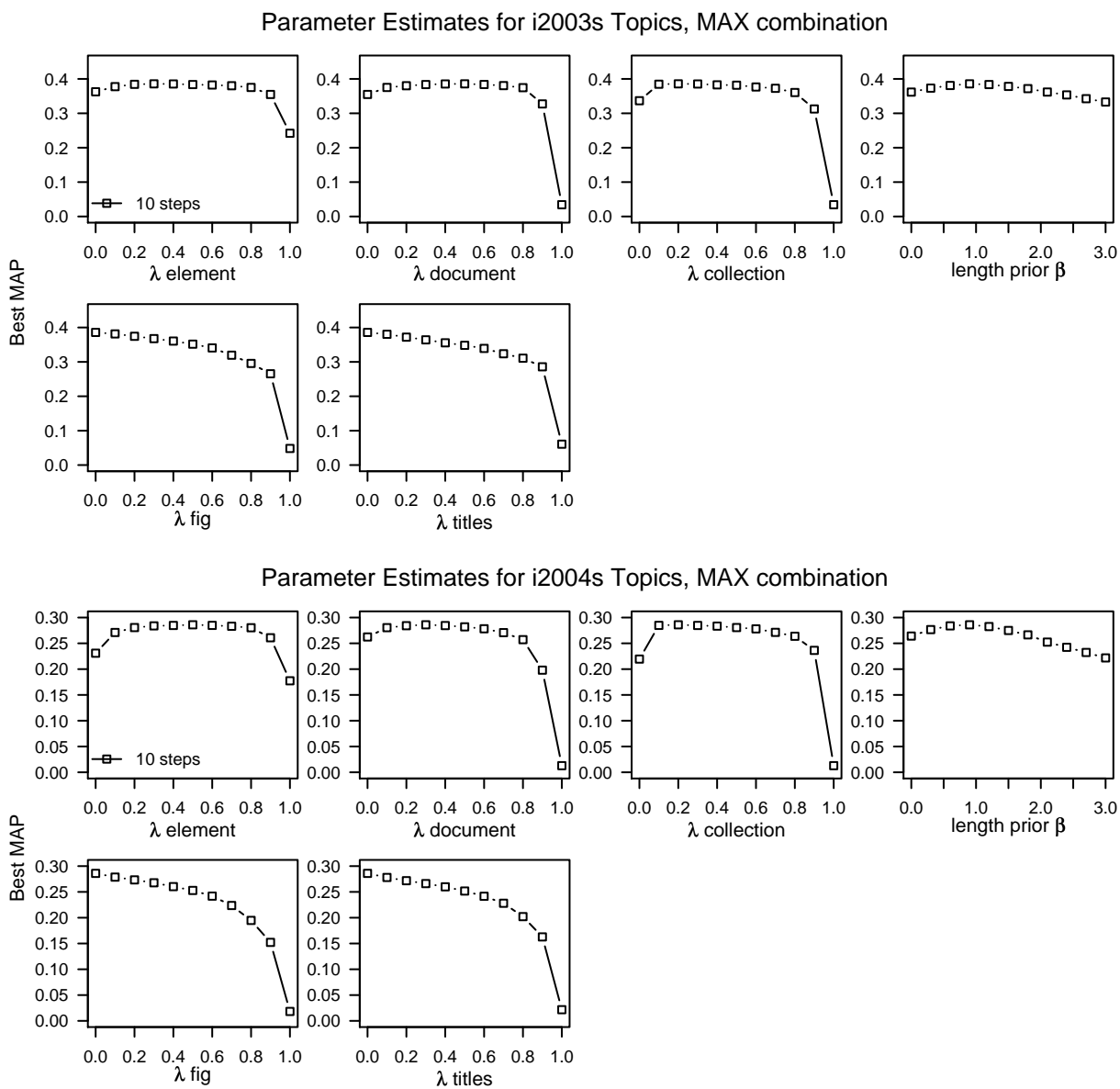


Figure 6.6: The results of the grid search testing 10 steps per parameter for the MAX combination method.

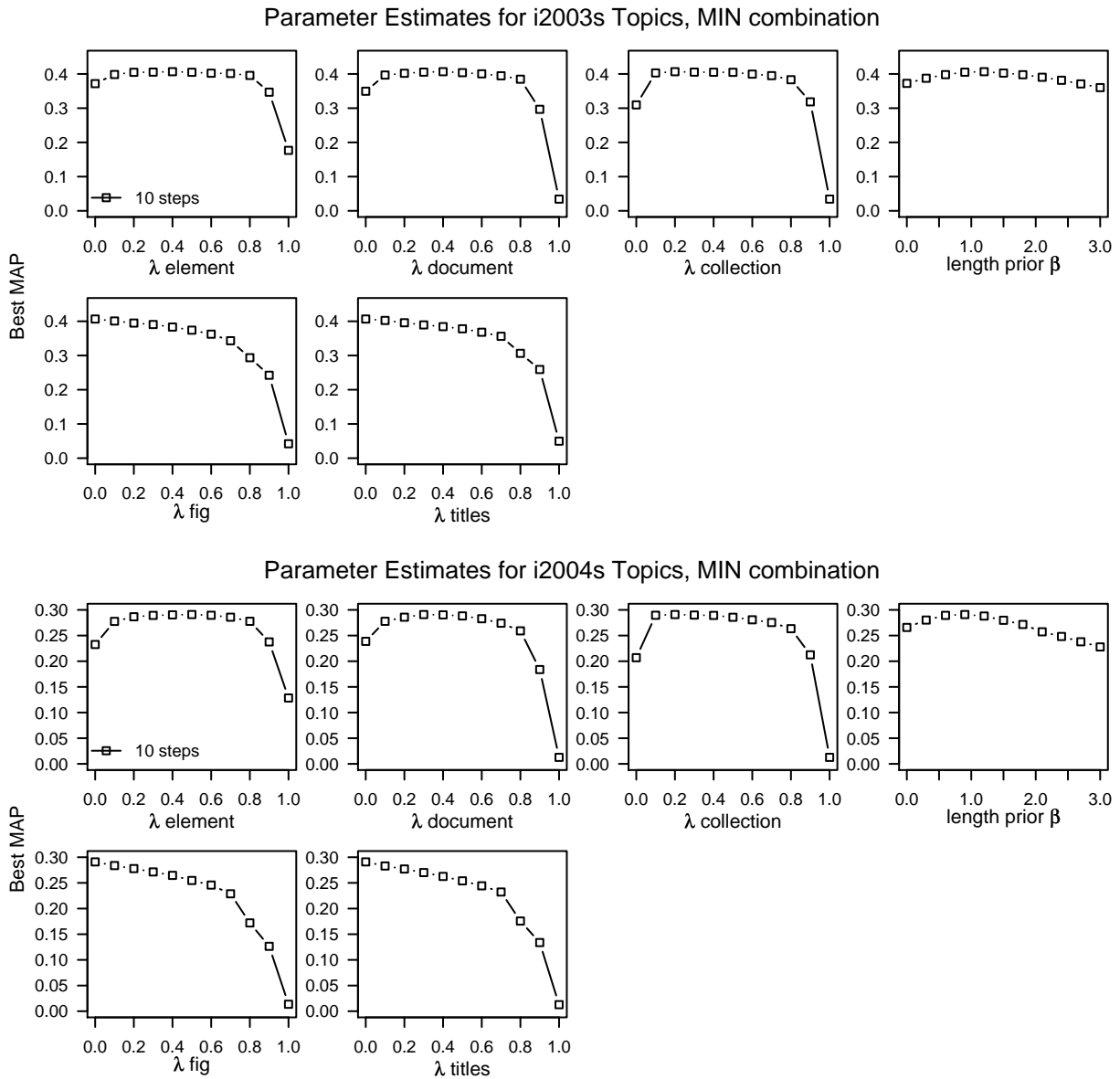


Figure 6.7: The results of the grid search testing 10 steps per parameter for the MIN combination method.

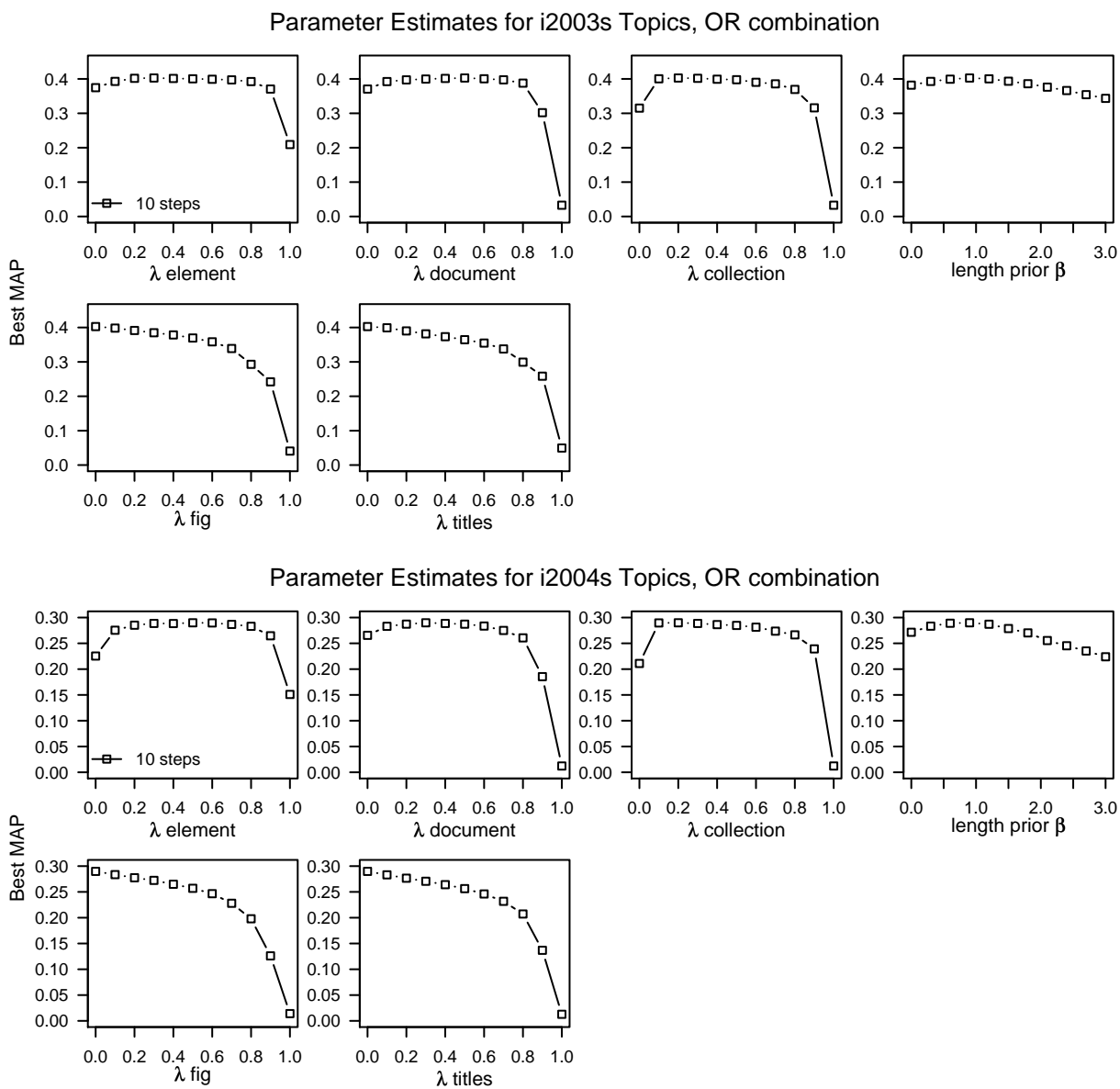


Figure 6.8: The results of the grid search testing 10 steps per parameter for the OR combination method.

On the test data, we see that the scope combination method choices result are reasonable; while the choice of MIN for application to the i2004s test data does not result in the best performance, the difference in MAP is minimal. Indeed, on the two keyword + structure testbeds only the AND combination method performs statistically significantly differently than the other combination methods. The results of our model show small improvements over the strong task-oriented approach presented in Section 4.2. With a relatively simple retrieval model and small set of representation functions, our results rival those that perform query rewrites, use pseudo-relevance feedback, or additional fields from the topic. While these subjects are not the focus of the dissertation, we would expect that using techniques such as those would additionally improve our results.

### 6.2.2 Discussion

The difference in results from one scope combination method to another for the structured queries warrants further investigation. Specifically, it is interesting to examine these methods relative to each other to see if the differences result from a few or many queries.

Figures 6.10 and 6.11 show a query-by-query comparison of the different scope combination methods using the best global parameters for each of the i2003s and i2004s testbeds. There are two causes for differing performance in these plots. The first is the scope combination method. The second results from the fact that the best global model parameters may be different from one method to another, which can impact even the queries which do not have nested scope operators. Queries with only a scope result operator will only be affected by parameter changes. If a scope operator is guaranteed to return only one element for all results due to the structure of the document collection, the rankings for these queries will also only be affected by parameter changes. There are 13 queries for the i2003s testbed and 16 queries for the i2004s testbed that are only affected by parameter changes. This leaves 17 queries in the i2003s testbed and 10 queries in the i2004s testbed that are directly affected by the choice of the scope combination method.

Examining Figure 6.10, we see that on the i2003s testbed the AVG scope combination method improves the performance over other methods most frequently, while the MIN scope combination method tends to hurt performance compared to other methods least frequently. Figure 6.11 shows a reversal of this behavior in the i2004s testbed, with AVG harming the performance on fewer queries and MIN improving the performance on more queries. For both testbeds, the AND method performs poorly for many queries. Given this analysis and the performance of the training parameters shown in Table 6.8, we would still probably choose the MIN method for test evaluation on the i2004s topics and the AVG method for test evaluation on the i2003s topics. This would be the wrong conclusion for the i2004s testbed, although this difference is very slight and certainly not statistically significant.

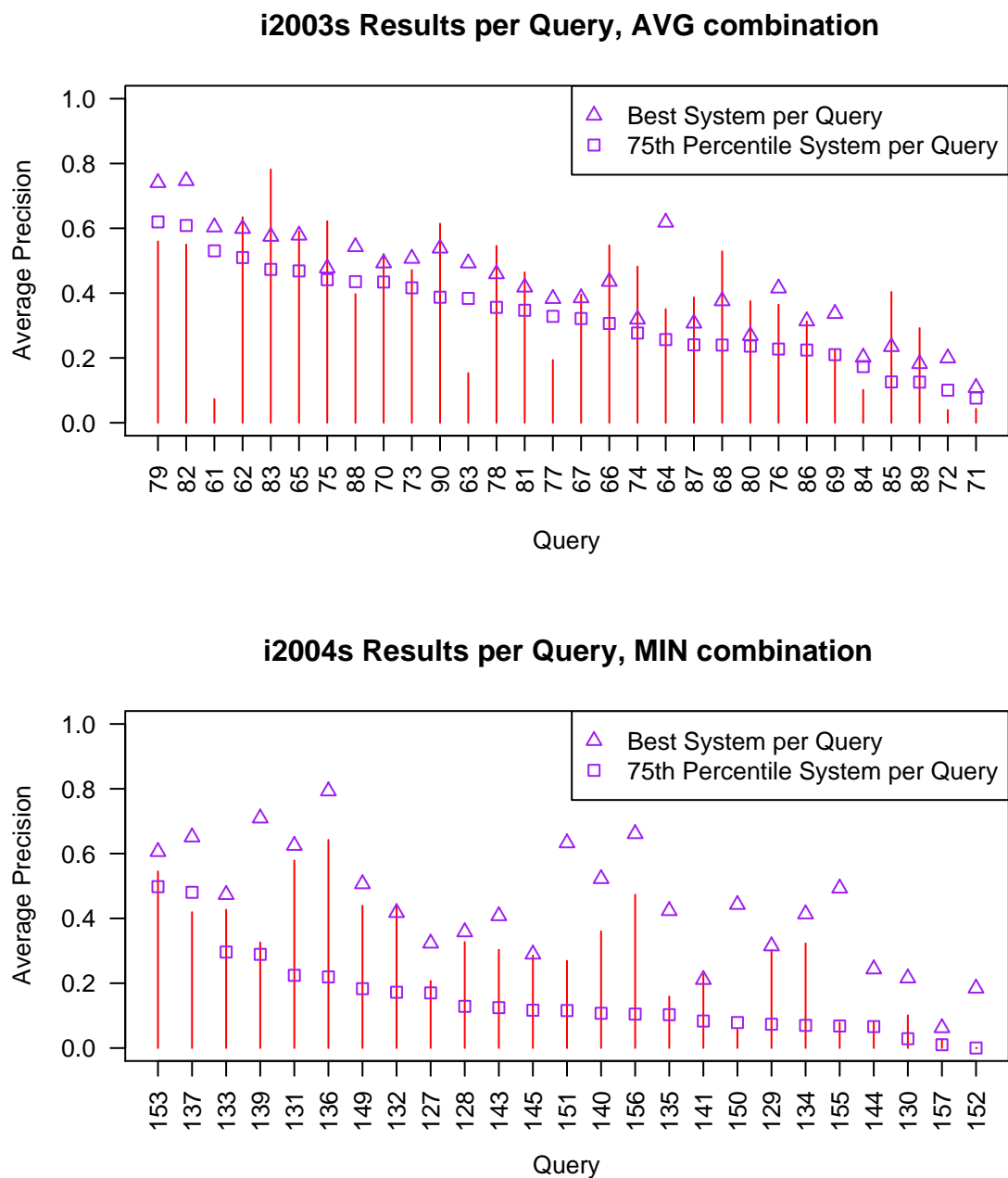


Figure 6.9: Results on structured queries compared to systems submitted to INEX. Queries are sorted in descending order by the observed performance of the 75 percentile system for each query.

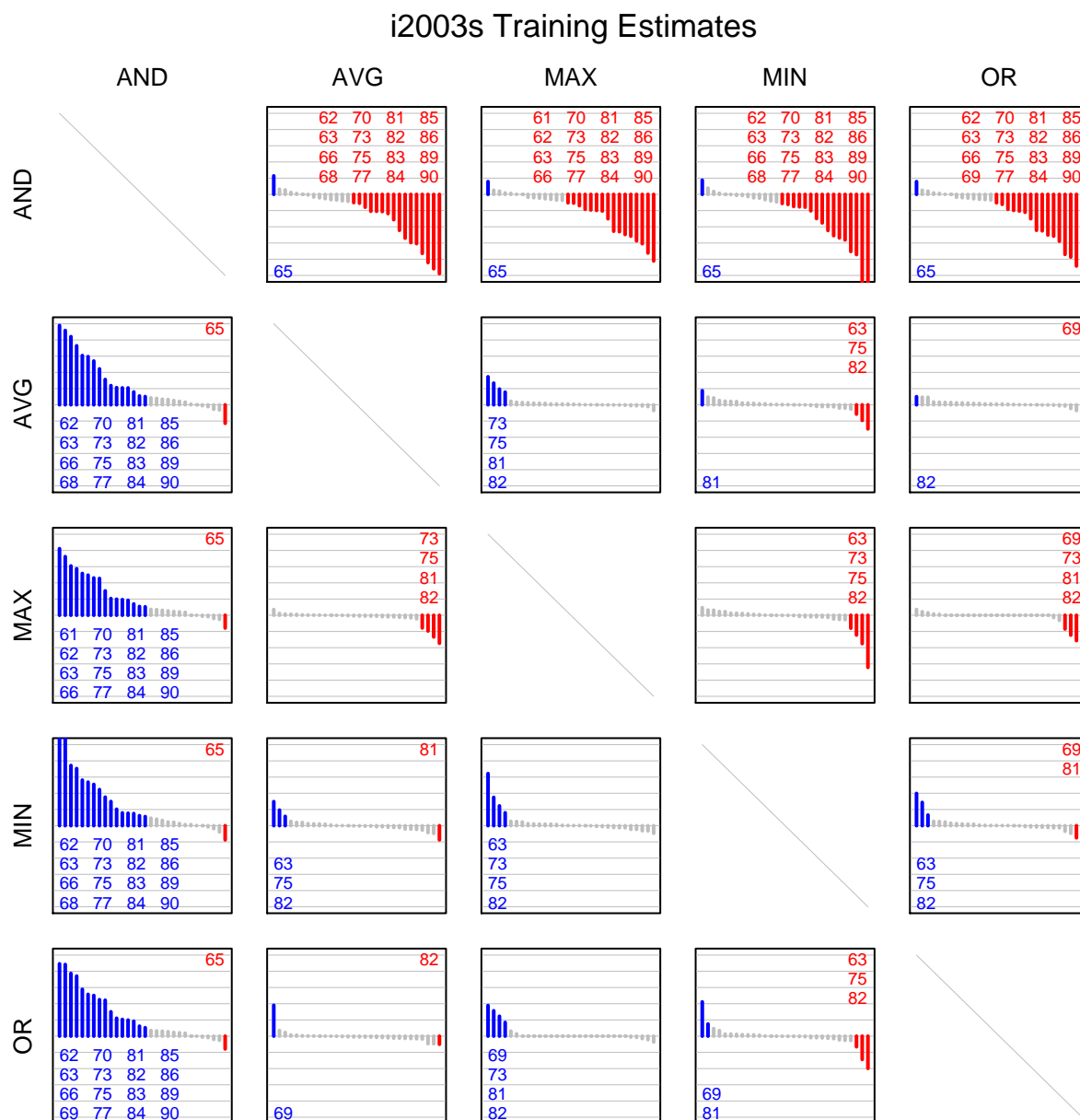


Figure 6.10: A comparison of the nested scope combination methods on i2003s with parameters chosen to maximize MAP across all queries. The plots show the unscaled difference in AP of queries of the row method with the column method. Queries for each plot are sorted in descending difference, with blue bars for systems where the row method received an AP at least 0.05 higher than the column method, and red bars for systems where the row system received an AP 0.05 or more less than the column method. Gray bars indicate a difference of less than 0.05 between the two methods. The gray horizontal lines indicate increments of 0.1. Query numbers in red in the top portion of the charts are queries where the row method performed worse than the column method, while query numbers in blue in the bottom indicate stronger performance by the row method.

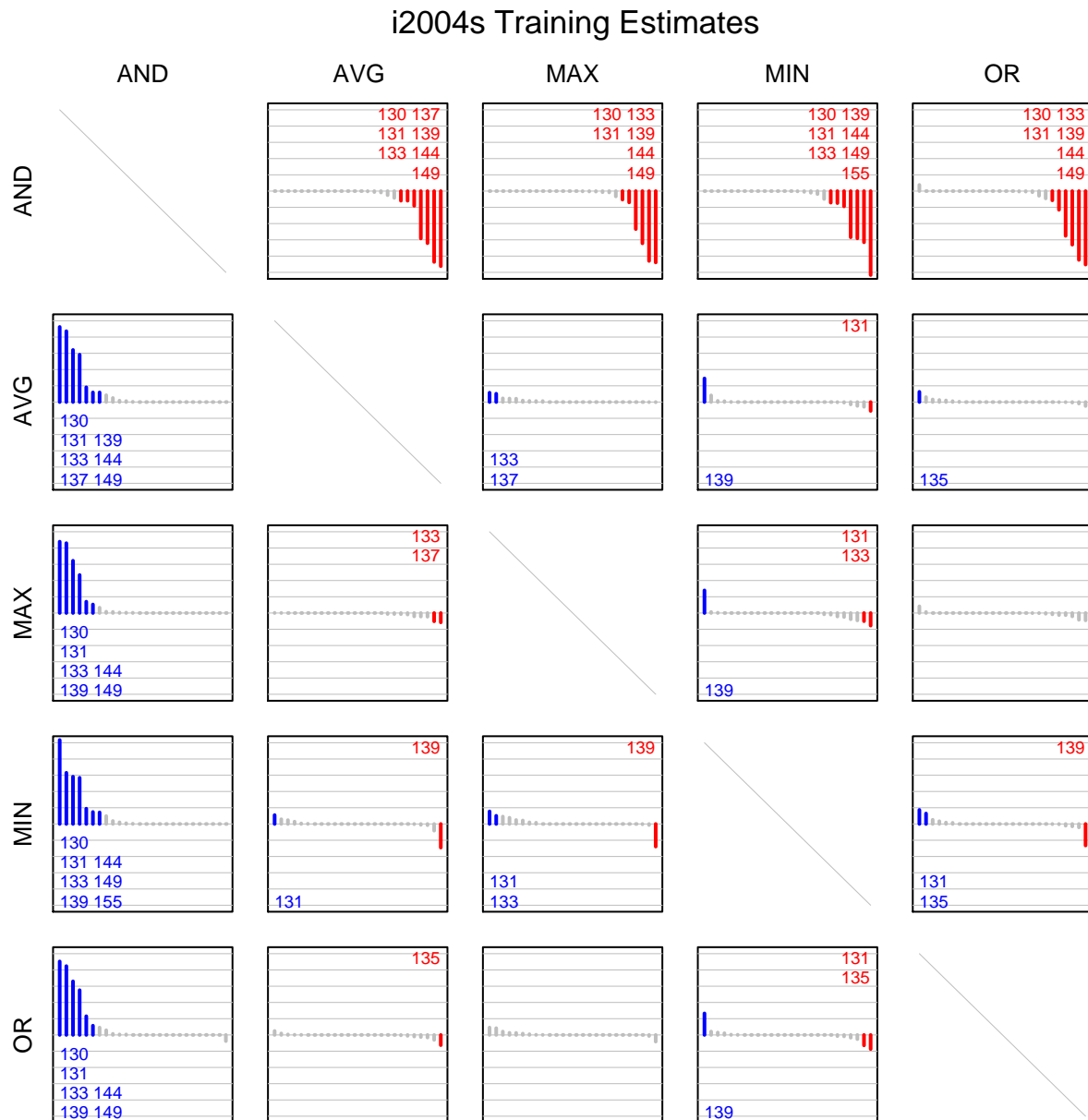


Figure 6.11: A comparison of the nested scope combination methods on i2003s with parameters chosen to maximize MAP across all queries. The gray horizontal lines indicate increments of 0.1. Query numbers in red in the top portion of the charts are queries where the row method performed worse than the column method, while query numbers in blue in the bottom indicate stronger performance by the row method.



The optimal parameters may change somewhat from one method to another, resulting in different performance even in the queries that are not directly affected by the choice of scope combination method. It is natural then to try and isolate the effect of parameter choice to see if there are inherent limitations for the scope combination method for these queries. To do so, we recompute the query-by-query comparison plots using the best parameters for each query instead of parameters chosen to maximize MAP across all queries. (Figures 6.12 and 6.13).

With the exception of the AND combination method, there seems to be very few queries for which one method greatly underperforms or outperforms other methods. There may be few queries for which there are a combination method has inherent limitations. Instead, we see artifacts of how these methods interact with the smoothing. That is, either the stronger performing scope combination methods may be less sensitive to smoothing parameters or they tend to have more consistency across queries.

Nevertheless, the few queries for which there are clear advantages or disadvantages of a specific scope combination method are worth investigation. Using the AVG combination method tends to outperform other methods for Topic 81, with out regard to whether we use the best parameters for the query or parameters optimized to for global performance. This topic is oriented toward retrieval of articles and is converted into the query

```
6.6    #SCOPE[result:article:length] ( #AND (
        #SCOPE[avg:p] ( multi concurrency control )
        #SCOPE[avg:p] ( algorithm )
        #SCOPE[avg://fm//at1] ( databases )
    ) )
```

Articles with a greater proportion of paragraphs are on the subject of algorithms for multi concurrency control in databases are more likely to be relevant. The AVG scope combination method simply averages the beliefs for each of the paragraphs being ranked (there is only one element that matches `./fm//at1` per article, the article's title).

The MIN scope combination method provides a very strong advantage for Topic 63, for which we use the query

```
6.7    #SCOPE[result:article:length] ( #AND (
        digital library
        #SCOPE[min:p] ( #AND (
            authorization access control security
        ) )
    ) )
```

The nested scope operator, to which the MIN combination method applies, selects the minimum belief from all paragraphs in the article; it need not contain one of the query terms to be considered by the model for ranking. It would be surprising if every paragraph in an article contain at least one of the query terms in the nested scope operator,

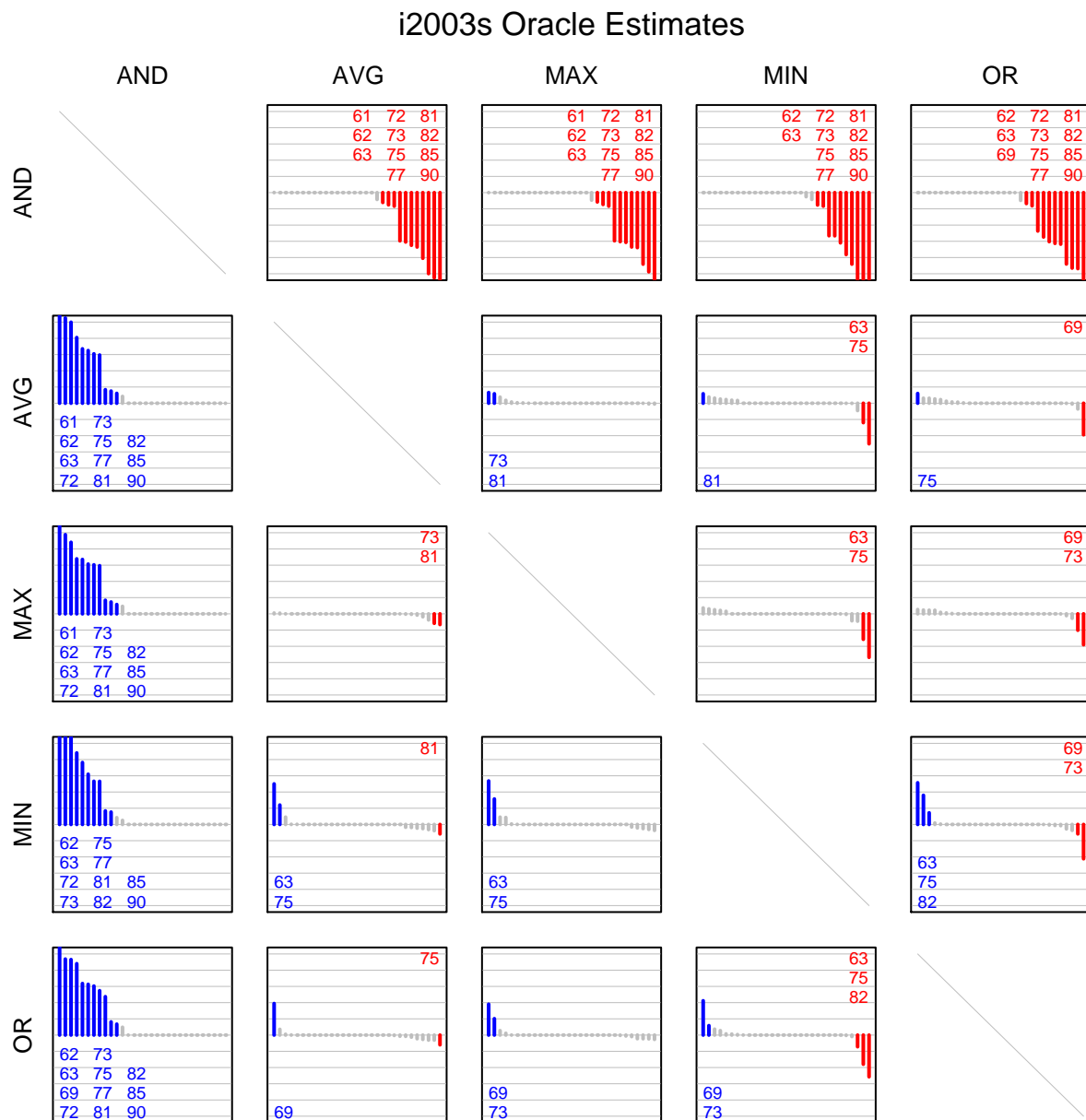


Figure 6.12: A comparison of the nested scope combination methods on i2003s with the best known parameters per query. The gray horizontal lines indicate increments of 0.1. Query numbers in red in the top portion of the charts are queries where the row method performed worse than the column method, while query numbers in blue in the bottom indicate stronger performance by the row method.



Figure 6.13: A comparison of scope combination methods on i2004s with the best known parameters per query. The gray horizontal lines indicate increments of 0.1. Query numbers in red in the top portion of the charts are queries where the row method performed worse than the column method, while query numbers in blue in the bottom indicate stronger performance by the row method.

so in effect we expect for most articles the selected belief will in effect disregard the element or other narrowly defined language models and instead rely on the document and collection language models.

It is interesting to compare Topics 63 and 81; they both request articles and have similar nested scope operators on paragraphs. In both cases we have cause to question whether the paragraph constraints are appropriate for the query, yet the MIN method has a strong advantage over the AVG method for Topic 63 while the MIN method performs worse than the AVG method for Topic 81. At one level, the MIN and AVG methods seem to express similar requests; the MIN method should prefer elements where all of the nested matching elements match the subordinate query, while the AVG method prefers elements where the nested matching elements tend to match the subordinate query well. However, if this were so, we would not expect the dramatic differences in performance observed.

We suspect the difference between the two queries' performance lies in query nodes nested within the scope operators themselves. For Topic 81, the nested scope operators contains query terms: `multi concurrency control` and `algorithm`. It is likely that many paragraphs will match these concept nodes in relevant documents. On the other hand, the nested scope operator of Topic 63 contains three concept nodes, `authorization`, `access control`, and `security`. It is unlikely that many paragraphs will receive high scores for this scope operator in relevant documents, because it is unlikely that paragraphs will contain all of these concepts. We feel it may be the case that averaging the scores of the paragraphs in this case may poorly estimate how well the article is about the topic's information need. Conversely, using the MIN combination for Topic 63 will in effect use beliefs estimated using the document and collection models, effectively ignoring the noisy element scores. The dominance of MIN over other combination methods is strong evidence of a poorly formed query.

However, the MIN combination method performs poorly for queries where the structure is important. For example, the NEXI query for Topic 139 makes heavy use of structural constraints:

```
6.8 //article[
      (about(../bb//au//snm, Bertino)
        or about(../bb//au//snm , Jajodia))
      and about(../bb//atl, security model)
      and about(../bb//atl,
        -"indexing model" -"object oriented model")]
```

The MIN combination method's reliance on the document and collection representations discards any context of bibliography entries in ranking, resulting in the relatively poor performance compared to other combination methods.

The OR scope combination method seems to have limited ability for Topic 135. It appears its query is poorly structured:

```
6.9 //article[about(./at1, summaries)]
      //sec[about(., "Internet security")
            or about(., "network security")]
```

The `about(./at1, summaries)` clause would match all article titles, including those in the bibliography.

The OR combination method will by the nature of its definition be at least as large as the maximum belief, increasing moderately with each additional element that has high belief values and increasing modestly for additional elements with low belief values. When there are few elements matching a scope operator, we would expect the OR method to perform similarly to the MAX method. When there are many elements that match the structural constraints of the scope operator, it is possible that the OR method could be very sensitive to the number of elements that match, even perhaps favoring elements with multiple matches to the structural constraints over ones with fewer elements matching keywords in the query.

If the poor structure of the NEXI queries is the cause of poor performance for the OR combination method, we must wonder why we do not observe a similar inherent limitation for the MAX scope combination method. For Topic 135, the MAX method achieves its best performance when no weight is placed on the element's representation function. The MAX combination method in this case reverts almost entirely to estimating beliefs based on the document and collection representations. The OR method, on the other hand, is sensitive to the number of matching elements; when the weight on the element representation function is zero, the OR method will exhibit a bias to elements where there are multiple matches to the nested scope operator.

### 6.2.3 Summary

The experiments with element retrieval presented in this Section first provided verification on another task that the model has desirable *structure-aware* properties. The strong results for the element retrieval task for both keyword and keyword plus structure queries helped to demonstrate that the *result-universal* adaptations to the retrieval model are effective. Finally, we explored aspects of how the model is also *structure-expressive* using the keyword plus structure queries. These experiments focused on the use of scope combination methods when combining the beliefs for multiple matching elements of nested scope operators.

It is difficult to draw strong conclusions with any certainty on such small sets of topics, but we do see hints of a trend. The AVG and MIN methods seem most resilient to superfluous constraints in the queries, with the AVG method striking a balance between using article and element evidence beliefs and the MIN method essentially ignoring more detailed constraints during belief estimation. The MIN method's reliance on the article evi-

Label	Function	Description
self	s	Text of the element
collection	$C_d$	The collection model
document	d	The text of the document
sentence	$O_{\text{sentence}}$	Text of sentence outside the element

Table 6.9: Functions of graph structure chosen in question answering experiments.

dence could harm performance when there are very specific structural constraints present in the query. The MAX and OR method perform similarly in practice, although the OR method may be sensitive when many elements match the structural constraints present in a nested scope operator.

## 6.3 Annotation Retrieval

This section revisits the experiments in Section 4.3 on the retrieval of answer-bearing sentences. We found previously that the Inference Network model was insufficiently *annotation-robust* to use heavily structured queries. A simple keyword + named entity answer placeholder baseline outperformed the richly structured queries. This section uses a similar query conversion process for each query method, simply replacing the extent retrieval operations with the appropriate #SCOPE operators.

We explore two approaches to improving the *annotation-robustness* of the retrieval model. First, Section 6.3.1 explores the use of a sentence representation to include context during ranking. The second approach, presented in Section 6.3.2, pads elements and annotations with probabilistic observations to include additional context and accommodate annotation boundary errors.

### 6.3.1 Better Representations

One way to improve the robustness of the retrieval model to the annotation errors observed in Section 4.3 is to simply improve the set of representation functions used. In addition to the self function s, the collection function  $C_d$ , and the document representation function d, we add the representation function  $O_{\text{sentence}}$ , which creates a representation from the sentence containing the element in question. These functions are listed in Table 6.9. As with other experiments, we use the maximum likelihood estimator for  $\phi$  and constant  $\lambda_f$  values across all elements.

Parameter	Held-Out Fold				
	1	2	3	4	5
Element	0.1 (0.0, 0.2)	0.1 (0.1, 0.3)	0.1 (0.0, 0.1)	0.1 (0.0, 0.2)	0.1 (0.0, 0.1)
Collection	0.4 (0.2, 0.7)	0.4 (0.2, 0.7)	0.4 (0.3, 0.7)	0.4 (0.2, 0.8)	0.5 (0.4, 0.8)
Document	0.2 (0.1, 0.2)	0.2 (0.1, 0.3)	0.2 (0.1, 0.3)	0.2(0.1, 0.3)	0.2 (0.0, 0.2)
Sentence	0.3 (0.1, 0.4)	0.3 (0.1, 0.4)	0.3 (0.1, 0.3)	0.3(0.1, 0.5)	0.2 (0.1, 0.3)
Length	2.1 (1.2, 2.1)	2.1 (0.0, 2.4)	2.1 (0.0, 2.4)	2.1(0.0, 2.4)	2.1 (0.0, 2.4)

Table 6.10: Estimated parameters for structured queries for the AVG combination method on the training sets corresponding to the held-out folds. A 95% confidence interval for each parameter is shown in gray.

The AVG scope combination method performed best on the training data sets, and Figure 6.14 shows the parameter estimate curves for this method. The improvements gained by using a non-zero weight on the sentence representation function show in this figure suggests that the addition of the sentence representation does improve the *annotation-robustness* of the retrieval model. Table 6.10 shows the parameter estimates on the training folds and their confidence intervals. The relatively high weight on the sentence representation demonstrates its usefulness, while the presence of non-zero weight on the element itself suggests that the structure in the queries can help improve results. Despite the low weight on the element representation function, we see from these curve estimates that structure is important; no weight on the element is clearly worse than a small weight.

The results in Table 6.11 verifies that the inclusion of the sentence representation does indeed provide a boost in the retrieval effectiveness. We see that the addition of the sentence representation yields a MAP of 0.240, which is higher than that of the keyword + named entity baseline. These results are at least as good as the feature-rich learning-to-rank model presented in Chapter 6 of Bilotti [4], which reports a MAP of 0.232. The AVG combination method performs statistically significantly better than the AND and MAX methods, but no statistically significant improvement was detected for the AVG method over the keyword + named-entity baseline, the Bilotti approach, or the MIN and OR combination methods. The failure to detect a statistically significant difference between the keyword + named-entity baseline and the structured query approach using the AVG combination method is a direct result of the large number of shallow questions in the testbed for which both approaches use the same queries (57 of 91). For the deep questions only, the AVG method performed statistically significantly better than the AND method and the keyword + named-entity baseline.

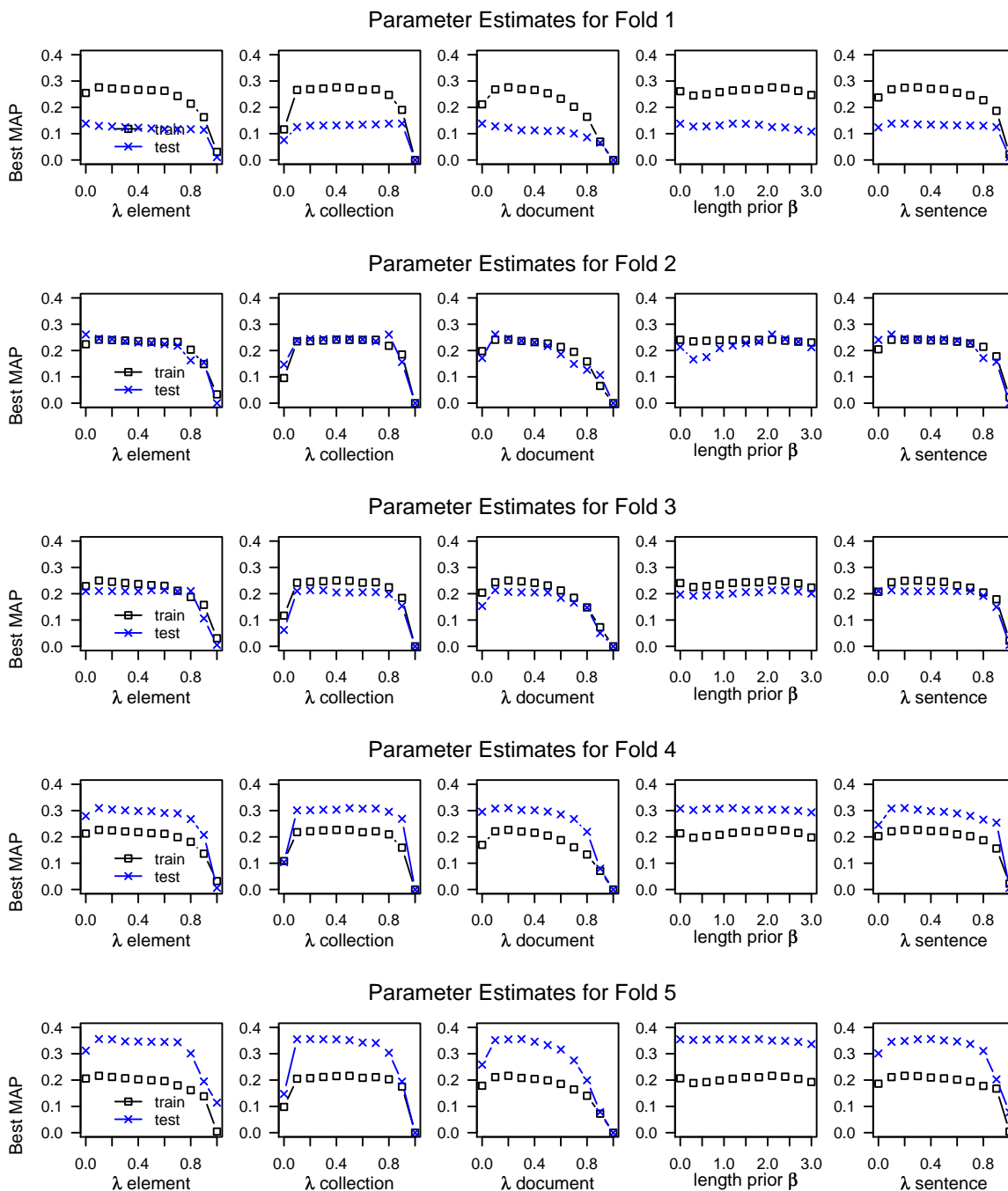


Figure 6.14: Parameter estimates for structured queries on training folds and test folds when using the AVG scope combination method. While the test performance can vary significantly from fold to fold, the shapes of performance of the training folds are quite similar to with the test folds.



Paritition	Keyword + named-entity	AND	AVG	MAX	MIN	OR
All	0.218	0.166	0.240	0.206	0.201	0.213
Shallow	0.197	0.197	0.197	0.197	0.197	0.197
Deep	0.232	0.182	0.303	0.296	0.242	0.270
Combined	0.211	0.191	0.240	0.238	0.215	0.227

Table 6.11: MAP of the structured query approach using the different combination methods. The AVG method performs strongest, outperforming the keyword + named-entity approach. Training separately on shallow and deep questions does not consistently yield an improvement over globally optimized parameters.

The simple addition of a sentence representation greatly improved the *annotation-robustness* of the retrieval model. Without the adaptations proposed in Chapter 5, it would have been difficult to describe and implement a similar improvement. Indeed, implementation of the sentence representation model for these results required extensive modification of the Indri source code. Indri has no notion of field evaluation using a broader extent (other than the document). Without the language to describe representations we proposed in Chapter 5, the use and construction of this representation might not be obvious to other researchers. With the extensions proposed in Chapter 5, the inclusion of the sentence representation becomes a natural choice.

For completeness, Table 6.11 also shows the results when using the other scope operator combination methods. We see again that the AVG combination method provides the largest boosts, verifying the conclusions made in Section 6.2. However, we see greater variation across the combination methods for this dataset, underscoring the need to make the combination method explicit. Training separately on the shallow and deep questions does not seem to provide consistent improvements, although it does not seem to hurt the effectiveness of the structured queries (Table 6.11).

Figure 6.15 shows the query-by-query comparison of the structured query method and AVG combination to the keyword + named-entity baseline results. Questions 1494 and 1453, which were most harmed by using the structured queries with the task-oriented model in Section 4.3, are no longer greatly harmed by the structured queries. For both questions, the structured query method now provides an average precision of 1, ranking the answer-bearing sentences over all other sentences in the collection.

Of the six remaining questions where the structured queries perform worse than the keyword + named entity baseline, the performance for Questions 1463 and 1537 is only worse due to different parameters because these questions have no semantic predicates.

The remaining questions for which the structured queries performed worse than the baseline all had fairly structured queries. For example, Question 1453,

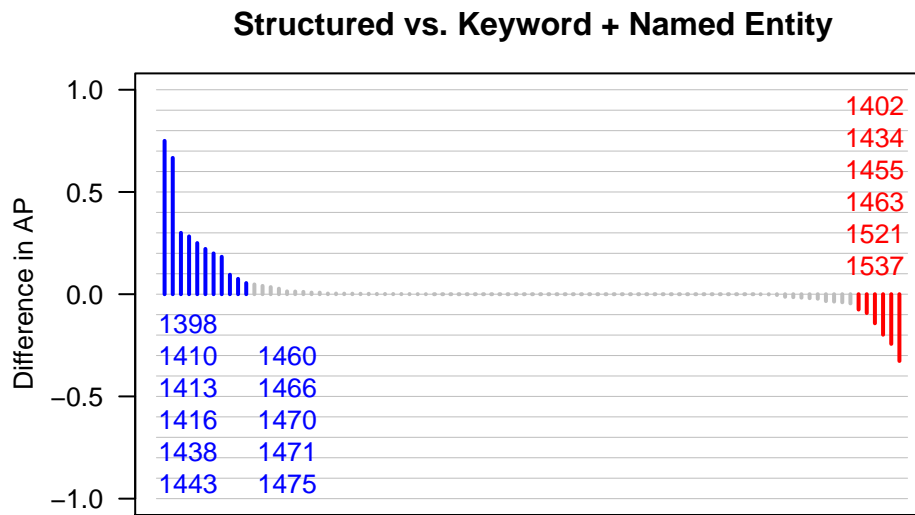


Figure 6.15: The structured query method with the AVG combination method helps more queries than it hurts relative to the keyword + named entity baseline.

What site did Lindbergh begin his flight from in 1927?

was converted into the query

```
6.10 #SCOPE[result:sentence:length]( #AND(
      #SCOPE[avg:target]( #AND(
        begin
        #SCOPE[avg:./argm-loc]( site )
        #SCOPE[avg:./arg0]( lindbergh )
        #SCOPE[avg:./arg1]( flight )
        #SCOPE[avg:./argm-tmp]( 1927 )
      ) )
      #ANY:location
    ) ) .
```

Many of the answer-bearing sentences for this question were quite long, for example:

```
6.11 The pope's visit could attract crowds that may even
surpass the 700,000 who welcomed Charles Lindbergh back
home a few weeks after his record flight from New York to
Paris in 1927.
```

Long sentences like these often have multiple semantic predicates. ASSERT identified three semantic predicates for this sentence:

- 6.12 [ARG0 The pope's visit] [ARGM-MOD could] [TARGET attract]  
[ARG1 crowds that may even surpass the 700,000 who  
welcomed Charles Lindbergh back home a few weeks after  
his record flight from New York to Paris in 1927.]
- 6.13 [ARG0 crowds] [R-ARG0 that] [ARGM-MOD may] [ARGM-ADV even]  
[TARGET surpass] [ARG1 the 700,000 who welcomed Charles  
Lindbergh back home a few weeks after his record flight  
from New York to Paris in 1927.]
- 6.14 [ARG0 the 700,000] [R-ARG0 who] [TARGET welcomed] [ARG1  
Charles Lindbergh] back [ARGM-TMP home a few weeks]  
[ARGM-TMP after his record flight from New York to Paris]  
[ARGM-TMP in 1927.]

BBN's Identifinder correctly identified New York and Paris as locations and Charles Lindbergh as a person.

None of the target verbs match `flight` and none of the ARG0 annotations match `lindbergh`. All that matches the structured in the query is that one of the ARGM-TMP annotations in the third semantic predicate matches `1927`. While this answer-bearing sentence clearly contains the answer, its wording does not yield an obvious semantic predicate closely resembling the one in the question. We do not expect these simple structured queries to do a good job at retrieving answer-bearing sentences when the structure is this different.

### 6.3.2 Probabilistic Observations

Section 5.4.1 proposed two methods for handling annotation boundary errors. Probabilistic containment of elements is not important to the annotation retrieval experiments because the semantic annotation process was performed on sentences themselves; we can be confident that the `target` annotations are contained within the sentences. However, the use of probabilistic observations around the boundary of an annotation may have an impact on the retrieval performance.

We explore an approach to padding the extents with probabilistic observations based on the distance of the observed token or concept from the annotation. This approach uses a pad width, where observations outside the annotation boundaries plus/minus the pad width get zero weight, but observations within the pad region receive a weight proportional to the distance of the observation from the annotation boundary.

Recall that an observed concept vector  $c$  is anchored at a location in text and is a binary vector. The dimensions of the vector are possible concepts and the values indicates the presence or absence of the concept at the observed concept vector's location in the text.

Then, we can estimate the probability that a specific observation vector is located within the set of vertexes (elements or annotations) returned by a representation function using:

$$P(c \in f(v)) = \begin{cases} 1 & \exists v_i \in f(v) : c \text{ is located in } v_i, \\ 1 - \min_{v_i \in f(v)} d(c, v_i)/pad & \exists v_i \in f(v) : d(c, v_i) < pad, \text{ and} \\ 0 & \text{otherwise,} \end{cases} \quad (6.1)$$

where  $d(c, v_i)$  measures the distance between observed concept vector's location and the element  $v_i$ 's location in text:

$$d(c, v_i) = \begin{cases} begin(v_i) - location(c) & \text{if } location(c) < begin(v_i), \\ location(c) - end(v_i) & \text{if } location(c) > end(v_i), \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Using these definitions, the belief estimate  $P(r_i | \phi(f(v)))$  becomes

$$P(r_i | \phi(f(v))) = \frac{\sum_{c:c_i=1} P(c \in f(v))}{\sum_{c'} P(c' \in f(v))}. \quad (6.2)$$

The numerator sums the observation probabilities over all observed concept vectors (locations) for which the concept  $i$  was observed. The denominator sums over all concept vectors in the padding regions to compute a "weighted" length of these padded elements.

We explore the use of these probabilistic observations as an alternative to the use of the sentence representation function used in Section 6.3.1. We consider a wide range of pad widths: 1, 2, 4, 8, 16, 32, 64, 128, 256, and 512. A small pad width provides local smoothing of extents, while a large pad width provides passage-level smoothing.

Figure 6.16 shows the parameter estimates across the five folds for a system using extent padding and the representation functions  $F = \{s, C_d, d\}$ . These figures suggest that pad width 4 or smaller provides little to no benefit to retrieval, while a large pad width is detrimental to the system's ability to retrieve answer-bearing sentences. Indeed, the results in Table 6.12 clearly show that this method of extent padding only provides small improvements, giving a MAP of 0.206, while the task-oriented model achieved a MAP of 0.201. We also explored the use of probabilistic observations in the element retrieval experiments for both structured and keyword queries. For these datasets, we also observed tiny but insignificant improvements for results.

While these results suggest that probabilistic observations may provide modest improvements to a system's annotation-robustness, a better representation function can provide much greater improvements to the robustness of the retrieval model. When we reflect on the nature of the errors found during failure analysis in Section 4.3.2, this

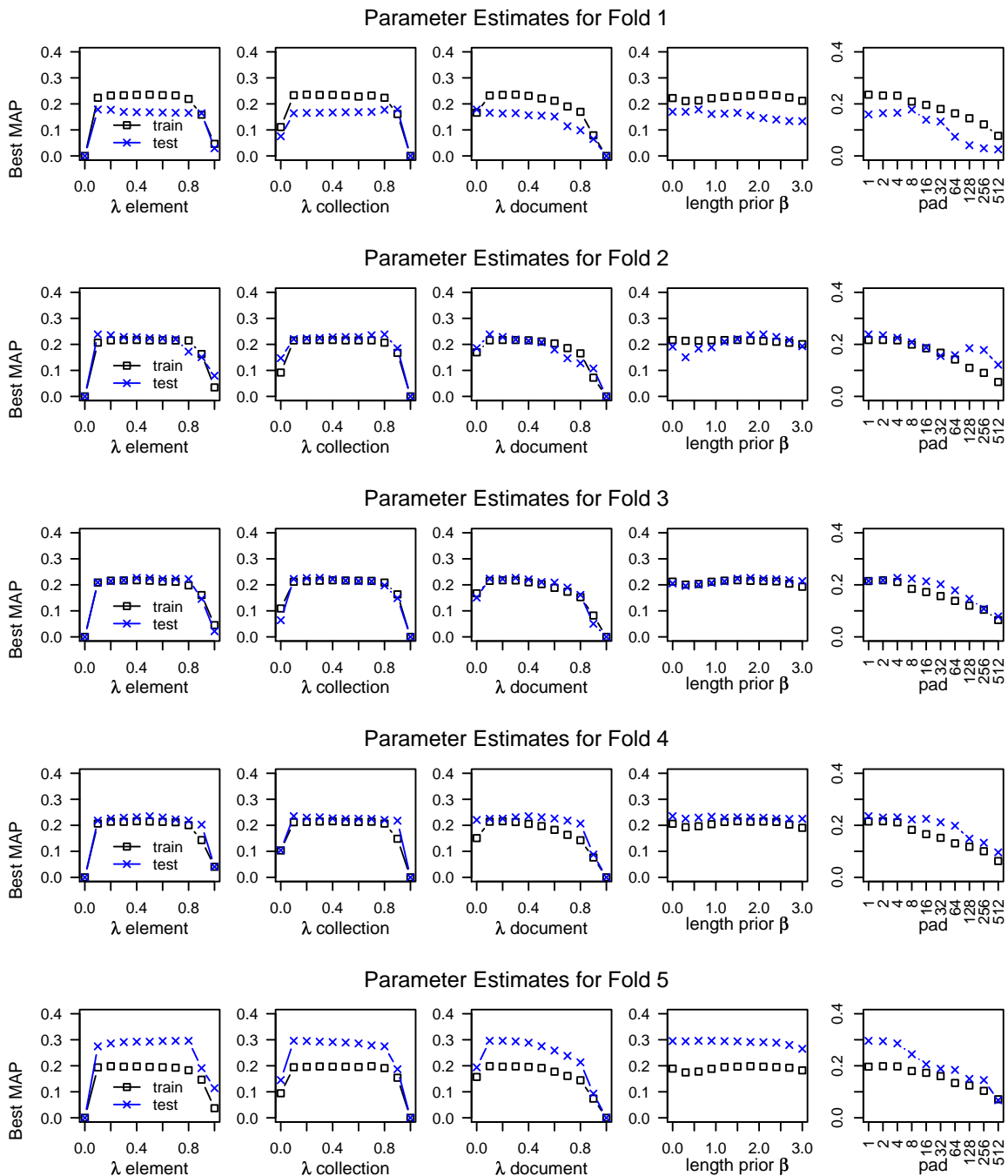


Figure 6.16: Parameter estimates for structured queries on training folds and test folds when using the AVG scope combination method and extent padding.

Partition	Sentence	Padding	Neither
All	0.240	0.206	0.201
Shallow	0.197	0.197	0.197
Deep	0.303	0.210	0.206
Combined	0.240	0.202	0.201

Table 6.12: MAP for retrieving answer-bearing sentences. The first row shows performance on test folds when using parameters optimized across all query types. The rows below show performance when using parameters optimized separately for shallow and deep queries, with the last row showing the combined shallow and deep results. While a weighted padding of extents does not work as well as the inclusion of a sentence representation, padding may provide modest improvements in the absence of the sentence representation.

is perhaps not surprising; many of the failures of the structure retrieval approach for answer-bearing sentence retrieval were a result of mislabeled or missing annotations. It is tempting to hypothesize that the use of annotator confidence estimates and indexing  $n$ -best annotations would help improve robustness for this retrieval task. Unfortunately, few annotation tools provide this output.

It is also tempting to dismiss the use of probabilistic observations as unimportant for retrieval. To do so would be premature. There are many ways one could estimate the observation probabilities, and better estimates and control of the contribution of these observations could result in greater improvements; we do not yet fully understand their use in retrieval. While the specific use probabilistic observations investigated in this section showed only small improvements for the retrieval of answer-bearing sentences, there may be other tasks for which this form of smoothing could be important.

### 6.3.3 Summary

The extensions to the Inference Network model proposed in Chapter 5 facilitated the natural inclusion of a sentence representation function. The use of a sentence representation function greatly improved the performance of the structured query method, enabling it to out-perform the keyword + named-entity baseline method presented in Section 4.3 and be at least as good as the more feature-rich learning-to-rank model investigated in Chapter 6 of Bilotti [4].

The improvement in performance was a direct side-effect of the improved ability for the Inference Network model to provide *annotation-robust* retrieval results. The use of probabilistic observations to pad extents did not result in significant improvements to the

retrieval performance, suggesting that careful choice of representations can have greater impact.

## 6.4 Chapter Summary

This chapter revisited the experiments of Chapter 4 to investigate the extended Inference Network model. The extensions to the Inference Network model yielded performance comparable to or better than each of the task-oriented models. The extended Inference Network model has the benefit of being a single retrieval model easily applied to all three tasks.

The use of the extended Inference Network model for each of the tasks has several specific benefits. A single model provides a shared vocabulary for the presentation and investigation of results. The modifications proposed for the Inference Network model include a clarified query language, where the effects of query construction are more obvious and direct. Using a mixture of multiple-Bernoullis driven by representation functions for inference network belief estimation resulted in a much cleaner model construction.

It is helpful to highlight the differences between the inference networks and queries used for the three retrieval tasks evaluated in this chapter. The only differences were the choice of representation functions and prior probabilities of relevance:

- $F = \{d, l, de_{\text{title}}, de_{\text{header}}, de_{\text{meta}}, C_d\}$  and a URL-type prior for known-item finding,
- $F = \{s, C_d, d, de_{\text{st}}, de_{\text{fgc}}\}$ , and result length prior for XML element retrieval, and
- $F = \{s, C_d, d, O_{\text{sentence}}\}$  and a result length prior for answer-bearing sentence retrieval.

The use of a shared model across all tasks allowed us to focus on the investigation of model properties, delivering results at least as good as strong task-specific models.

The experiments in this chapter have shown that the model is *structure-aware*, through the ability to choose an appropriate and useful set of representation functions to combine evidence from multiple representations. We have demonstrated the model is *structure-expressive* through the shared query language appropriate for all three tasks. The experiments on element retrieval and answer-bearing sentence retrieval found that the AVG scope combination method is a robust way to combine evidence when multiple elements match a nested scope operation. This chapter shows how the model is *result-universal* through the structure-expressivity of the query language, enabling the restriction results to requested element types, and the use of prior probabilities. This property is verified through the strong results for element retrieval and answer-bearing sentence retrieval. Finally, although probabilistic observations provided minimal improvements to

the annotation-robustness of the model, we have found that an appropriate choice of representation functions can greatly improve the *annotation-robustness* of rankings. Specifically, we showed that a sentence representation function can improve results when the structure of the query only partially matches that of the answer-bearing sentences.



# Conclusions

---

This chapter summarizes the work presented in the dissertation, considers the major contributions, and highlights directions for future research.

## 7.1 Summary

This dissertation took a holistic view of retrieval of documents with structure and annotations, considering a wider variety of retrieval tasks than prior work. This resulted in the articulation of several properties important for retrieval in these scenarios. Specifically, any retrieval system hoping for generality and broad applicability to tasks of retrieving documents with structure and annotations should be *structure-aware*, *structure-expressive*, *result-universal*, and *annotation-robust*. The structure-aware property asserts that the retrieval model should be able to use evidence about relevance from multiple representations of the content of an element within a document. A structure-expressive query language should have the ability to specify constraints on the relationships between keywords and elements within a document. A result-universal retrieval model should be able to rank and mix results of any result type. Retrieval models that are annotation-robust can recover from errors in annotations to effectively make use of the structure present in the query and documents. To the best of our knowledge, no prior work has explicitly identified all four of these properties as important.

In Chapter 4, we performed experiments on known-item finding, element retrieval, and the retrieval of answer-bearing sentences for question-answering. These experiments used retrieval models designed for these tasks. These statistical language modeling and inference network approaches are representative in spirit and effectiveness of the approaches used in recent research. The mixture of multinomials used for known-item finding closely resembled the model used by Ogilvie and Callan [63], which first investigated the differences between *in-model combination* and *score-combination*. While the presented approaches are similar models, they made different choices for smoothing (smoothing using Dirichlet priors vs. Jelinek-Mercer). The inference networks used in the question-answering experiments closely resembled our work in Bilotti et al [5] and [60], which highlighted the potential for using semantic structure during retrieval to better serve question-answering systems. However, the forcing of the mixture of multinomial be-

belief estimates into the Inference Network model to gain result-universality and structure-expressiveness was ad-hoc and unjustified. Chapter 4 also discussed the use of grid search for parameter estimation, demonstrating that a fairly coarse grid search can yield good parameter estimates for length priors and mixture model parameters.

Chapter 5 proposed an extended inference network model specifically designed to address each of the four properties for supporting structure and annotations.

In order to support the *structure-aware* property within the Inference Network model, we showed how different representations for an element may be combined using a mixture of multiple-Bernoullis, similar to the mixture of multinomials within the language modeling framework. We formalized the process of creating representations through the use of representation functions. These representation functions operate on elements in the collection's structure and annotations to group related elements into representations. The choice of functions explicitly encodes our assumptions about which elements, annotations, or representations are related to the element whose score is being estimated. The introduction of the representation functions also simplifies the construction of queries in the most common uses cases for mixture models within the Inference Network retrieval model, removing the need for the weighted sum operator and the use of field evaluation when the representations and their weights are the same for each query term or concept.

While prior work with Inference Network models had provided some support for *structure-expressive* and *result-universal* retrieval, the query languages provided limited support for structural constraints and were unclear about the proper use of priors and belief estimation with nested extent restrictions. We proposed the replacement of arbitrarily located extent restrictions with the #SCOPE operator. The scope operator allows extended support for structural constraints and make explicit the method used for the combination of beliefs. The requirement that priors be placed within the specification of a scope operation also clarified the use of prior probabilities in ranking.

Finally, we discussed the extended Inference Network model to make it more *annotation-robust*. We described how the model could be made more robust to annotation boundary errors through probabilistic containment and probabilistic observations. We also described how annotator confidence estimates could be easily used during ranking within the Inference Network model.

Chapter 6 explored properties of the extended Inference Network model with experiments on known-item finding, element retrieval, and the retrieval of answer-bearing sentences. We applied the model to achieve state-of-the art performance rivaling that of the best published results for these tasks, verifying the model's *structure-aware*, *structure-expressive*, and *result-universal* properties. The use of the maximum likelihood estimator for the multiple-Bernoulli is reasonable, provided the choice of representation functions include a collection function for smoothing. We verified that using  $\lambda_f$  parameters tuned using grid-search to combine the multiple-Bernoullis can achieve state-of-the-art retrieval effectiveness. Experiments on the element retrieval and answer-bearing sentence retrieval

suggested that averaging the beliefs of multiple extents matching a nested structural constraint provides good results for element retrieval. Finally, we showed how the extended Inference Network model allows for the easy inclusion of a sentence representation, improving the annotation-robustness of the model for answer-bearing sentence retrieval.

The experiments in Chapters 4 and 6 also demonstrated previously unknown properties of mixtures of language models with respect to retrieval effectiveness. The parameter estimate curves enabled by the grid search show remarkably smooth and flat curves. This highlights the robustness of the mixture models and in-model combination, as well as underscoring the fact that only a fairly coarse grid search is required for good estimates.

## 7.2 Contributions

As the field of Information Retrieval has matured, researchers have considered a wider variety of search tasks. It is true that researchers have studied the use of multiple representations and document structure for years, but it has only been recently that large reusable test collections have been available for tasks such as these. The Text Retrieval Conference (TREC) and the Initiative for the Evaluation of XML Retrieval (INEX) have recently created large Web and XML test corpora which has drawn extra attention to tasks and techniques that can benefit from the use of document structure and multiple representations. In parallel, the rapid growth and success of commercial search engines on very large samples of the World Wide Web has renewed researcher's beliefs that document structure and multiple representations are crucial for effective search in very large scale systems.

With this renewed interest in the use of structure and multiple representations, researchers within the field of Information Retrieval have proposed numerous modeling and ranking techniques. The effectiveness of these techniques demonstrated the usefulness of structure and representations for ranking. However, techniques are often heuristic adaptations specific to one problem domain. There has been little work to develop a model that is flexible and effective for a wide variety of search tasks that leverage document structure and annotations.

This dissertation fills the hole by providing a well-motivated theoretical model of the structure of information in a collection and demonstrates how it can be leveraged for ranking across a variety of problems. This work builds on top of proven techniques for statistical language modeling and the Inference Network model. The model makes explicit researchers' assumptions about which structural information is related. The parameters estimated for the model directly reflect the accuracy of the researchers' assumptions. The effectiveness of a coarse grid search removes the burden of estimation from other researchers and allows them to focus on which features of the structure are important for effective ranking with their specific tasks and corpora.

Even if other researchers choose not to work within the extended Inference Network model presented in this dissertation, we have identified four important properties for retrieval of documents with structure and annotations. We feel that explicit recognition of these properties will help guide other's research in these tasks.

Apart from the applications of XML element retrieval, known-item finding of Web documents, and answer-bearing sentence retrieval investigated in this dissertation, the model is easily applicable and adaptable to other applications where structure and multiple representations could be helpful in modeling and ranking. There is no requirement within the model that the information or beliefs be estimated on text representations; for example, measures of image similarity could be used for comparing image elements within documents. Other measures of similarity could be used for numeric elements, names, or dates. Example applications where the model could be applied include multi-media text retrieval, search of medical records, and legal document retrieval.

The work in this dissertation also highlights that the information needs of natural language processing applications are different than those of a human user. Natural language processing applications often have much more precise expectations about the form of useful information. That is, the applications have greater sensitivity to the structure of the text itself. In many cases, these expectations are expressible in terms of constraints on the relationships between keywords and annotations of the linguistic structure of the text.

The extensions to the Inference Network model also made important strides in reducing the sensitivity to badly formed queries and noisy document structure. Our failure analysis in the task-oriented retrieval models showed that there was often a mismatch between the structured query and the actual information need in the element retrieval experiments. The use of the AVG combination method for nested scope operators provided a good balance between effectively using good structural constraints, while being robust and providing reasonable results when there was a poorly chosen structural constraint. The strength of averaging the beliefs of multiple matching elements was more pronounced in our answer-bearing sentence retrieval experiments, where there may be mismatch between the annotations in the documents and the structure of the query. This mismatch may be a result of difference in the language itself, but also common errors in the output of natural language processing tools. We have shown that the AVG combination method used in conjunction with a good set of representations can provide consistent and strong improvements for the use of structure in queries.

Some of our early work in this area [61] proposed a more complex hierarchical model to create structure aware retrieval models. This model smoothed elements using the hierarchy present in each document. One goal at the time was to develop a structure aware retrieval model that hid many of the smoothing decisions from the system administrator or information retrieval researcher. However, this approach proved difficult to interpret and had many unspecified retrieval model parameters. The use of representation functions, which exposes smoothing decisions to the researcher in a simple and interpretable man-

ner, is in some ways more powerful and more transparent. We still rely on the researcher to choose a good set of candidate representations in order to narrow the parameter search; it behooves us to ensure that these choices will be clear to other researchers. Similarly, the choice of belief combination for nested extent retrieval in structure-expressive queries exposes complexity to the query authors during retrieval. Averaging the beliefs may prove to be a good default, but the different methods do result in different behavior during ranking. As we learn more about these retrieval tasks, we may be able to hide more of the retrieval model decisions from the experimenters, but at this time it will serve us best to expose these decisions as simply and as transparently as possible.

The extensions and modifications to the Inference Network model proposed in this dissertation are driven by desires to simplify the setup of experiments for other researchers. Specifically, the introduction of the model representation layer in the Inference Network (Section 5.1) enables moving the in-model combination out of the query language and into system parameters. Thus, one should no longer need to write long queries with duplicated query terms and field evaluations in `#WSUM` operators. This use multiple representations within Inference Networks is frequent enough that it deserves to be handled in a simple way that reduces the burden on other researchers, improving the likelihood they will write effective queries. Similarly, the suggested introduction of the `#SCOPE` operator is intended to clarify the appropriate construction of queries with nested structural constraints. This operator makes the use of prior probabilities and belief combination methods explicit, thus reducing the likelihood that an researcher may omit an important combination method or fail to understand the impact of nested extent retrieval during ranking. The proposed modifications are important. Misuse of nested extent retrieval is a common source of errors, and the `#SCOPE` operator clarifies extent retrieval. Moving representations and their weights from queries into system parameters also makes the construction of inference networks using multiple representations much more straightforward, thus reducing the risk of poorly formed queries.

## 7.3 Future Work

This section describes areas of future work that build upon the results of this dissertation.

**Scaling structured queries and multiple representations to larger collections** As more test collections grow beyond a terabyte of text, it is crucial to scale the methods of constructing representations and evaluating structured queries. In general, search on a pre-built index is at worst case linear in the number of ranked documents and in practice sub-linear. Yet there are cases where the indexing costs are non-linear. Similarly, the cost per candidate result during retrieval may be high.

The choice of representation functions one may wish to use during retrieval can have an impact on the organization of the index structures. It is common practice to group the text of all representations one may wish to use within an “expanded document.” This allows fast access to these representations during retrieval, as well as the use of structured queries and extent retrieval restricted within a representation. The construction of the expanded documents requires a pre-processing pass of the text in a collection and a large sort of the alternative representations in order to group the text of these representations with the document. With the exception of the in-link text of web documents, this dissertation largely focused on representations that are formed from elements within a document. As we wish to explore more representations that exist outside a document, it will be important understand and improve the indexing and retrieval structures and algorithms to ease experimentation.

Furthermore, nested extent retrieval in large collections can be expensive. For example, consider the case where we wish to place additional constraints on the text of in-links to a document. Some documents will have many links. Current implementations of the structured query operators require computing the beliefs over all links, resulting in very slow retrieval performance. Despite retrieval sub-linear in the number of documents in the collection, the cost per document can be high.

Now that we have a better sense of what representation creation functions are important and common effective uses of structure in queries, we can use this knowledge to focus our efforts in optimization and approximation. Our link example may be improved by an approximation to the AVG combination method. Other optimizations to structured query operators and the use of representations will be important to their practicality in large test collections.

**Scaling to many representations** Currently it is practical to estimate parameters for up to around seven representations. The exponential nature of the grid search makes it difficult to scale to larger representation sets and this approach relies upon a researcher to choose a set of promising representations. While for many retrieval tasks only a small number of representations may be important, search at web scale may benefit from a larger number of representations. There are two primary approaches to considering a larger number of representations. The first would be to automatically reduce the number of representations used during ranking; the second would be to use parameter estimation methods that are not exponential in the number of representations.

The representation functions provide a language for the construction of possible representations. These representation functions are analogous to features in machine learning. We believe that simple analysis of a large set of potential representations on a test collection may be sufficient to automatically select a small set of promising representations. For example, one could choose the representation functions that yield the best retrieval per-

formance when combined with collection representation. Or more sophisticated methods, drawing from feature selection literature in machine learning or statistics could yield better choices of representations. Understanding correlations between representations could also help the process of choosing a good set of representations.

Once a promising set of representations has been selected, it may be important to use a more scalable method than grid search. While grid search may remain the most reliable method of parameter estimation, we may use grid search on a smaller number of representations to validate the optimality of other parameter estimation methods. To the best of our knowledge, no prior work exists on comparing methods for the estimation of mixture model parameters such as line search, approximations of gradient ascent, using regression parameters, or learning to rank. The use of a grid search as a reference across many test collections would provide valuable insight to the stability of parameters these search methods employ, such as search step sizes or regularization parameters.

**Using knowledge to improve annotation retrieval** The failure analysis of answer-bearing sentence retrieval demonstrated that mismatch between language use resulted in queries that did not match the language and structure of the answer-bearing sentences. There are multiple ways we believe using additional knowledge about language structure could improve the results on annotation retrieval.

The mismatch resulting in annotation errors could be addressed through indexing multiple annotation structures per semantic predicate, weighted by belief of accuracy. This data would need to come from the annotation tools themselves, but having the alternative annotations indexed could greatly improve the robustness of these methods.

Mismatch in word use may be partially addressed through the use of synonymy. Careful, weighted expansion, such as expanding the target verb with a list of synonyms may enable effective use of knowledge about meaning. Since such an expansion would be anchored to the target verb of a specific semantic predicate, this use of meaning may be controlled enough to overcome the limitations of prior attempts to use synonymy in text retrieval systems.

A similar form of knowledge would be the understanding of other equivalent ways to structure the semantic predicate that may contain an answer. One could then create multiple queries, one for each potential answer-bearing semantic structure.

Ultimately, the best way to improve the annotation-robustness of retrieval systems will be through more careful failure analysis of the errors annotators make and how those errors interact with retrieval models. Additional careful data analysis should be used to identify which methods of annotation-robustness seem the most promising.

# Bibliography

---

- [1] S. Beitzel, E. Jensen, R. Cathey, L. Ma, D. Grossman, and O. Frieder. Iit at trec 2003, task classification and document structure for known-item search. In *The Twelfth Text Retrieval Conference (TREC 2003)*, pages 311–320, 2004.
- [2] D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. An algorithm that learns whats in a name. *Machine Learning*, 34(1-3), 1999.
- [3] M. Bilotti. Query expansion techniques for question answering. Master’s thesis, Massachusetts Institute of Technology, 2004.
- [4] M. Bilotti. *Linguistic and Semantic Passage Retrieval Strategies for Question Answering*. PhD thesis, Carnegie Mellon University, 2009.
- [5] M. W. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. Structured retrieval for question answering. In *SIGIR ’07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–358, New York, NY, USA, 2007. ACM.
- [6] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.
- [7] F. J. Burkowski. Retrieval activities in a database consisting of heterogeneous collections of structured text. In *Proceedings of the Fifteenth Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 112–125, 1992.
- [8] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching xml documents via xml fragments. In *SIGIR ’03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 151–158, New York, NY, USA, 2003. ACM Press.
- [9] V. R. Carvalho, J. L. Elsas, W. W. Cohen, and J. G. Carbonell. A meta-learning approach for robust rank learning. In *Proceedings of the SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.
- [10] V. R. Carvalho, J. L. Elsas, W. W. Cohen, and J. G. Carbonell. Suppressing outliers in pairwise preference ranking. In *Proceedings of the 17th Annual ACM Conference on Information and Knowledge Management (CIKM 2008)*. ACM Press, 2008.



- [11] J. Chu-Carroll, J. Prager, K. Czuba, D. Ferrucci, and P. Duboue. Semantic search via xml fragments: a high-precision approach to ir. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 445–452, New York, NY, USA, 2006. ACM Press.
- [12] C. L. A. Clarke, G. V. Cormack, and F. J. Burkowski. Schema-independent retrieval from heterogeneous structured text. In *Fourth Annual Symposium on Document Analysis and Retrieval, Las Vegas, NV*, pages 279–290, 1995.
- [13] C. L. A. Clarke, G. V. Cormack, and C. R. Palmer. An overview of multitext. *SIGIR Forum*, 32(2):14–15, 1998.
- [14] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.
- [15] K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press, 2001.
- [16] N. Craswell and D. Hawking. Overview of the trec-2002 web track. In *The Eleventh Text REtrieval Conference*, pages 86–95. NIST Special Publication 500-251, 2002.
- [17] N. Craswell and D. Hawking. Overview of the trec 2003 web track. In *The Twelfth Text REtrieval Conference*, pages 78–92. NIST Special Publication 500-255, 2003.
- [18] N. Craswell and D. Hawking. Overview of the trec 2004 web track. In *The Thirteenth Text REtrieval Conference*. NIST Special Publication 500-261, 2004.
- [19] E. A. Fox. Composite document extended retrieval. In *Proceedings of the Eighth Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 42–53, 1985.
- [20] E. A. Fox and R. K. France. A knowledge-based system for composite document analysis and retrieval: Design issues in the CODER project. Technical Report TR-86-6, Virginia Tech, 1986.
- [21] E. A. Fox and J. A. Shaw. Combination of multiple searches. In *Proceedings of the Second Text REtrieval Conference (TREC)*, pages 243–249, 1994.
- [22] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.
- [23] N. Fuhr, N. Goevert, G. Kazai, and M. Lalmas, editors. *Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*. ERCIM, 2003.

- [24] N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors. *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2005)*, volume 3977 of *Lecture Notes in Computer Science*. Springer Verlag, Heidelberg, 2006.
- [25] S. Fujita. More reflections on “aboutness” TREC-2001 evaluation experiments at just-system. In *The Tenth Text REtrieval Conference (TREC 2001)*, pages 331–338, 2002.
- [26] M. Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkerson. Structured answers for a large structured document collection. In *Proceedings of the Sixteenth Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 204–213, 1993.
- [27] D. Graff. *The AQUAINT Corpus of English News Text*, volume Cat. No. LDC2002T31. Linguistic Data Consortium (LDC), 2002.
- [28] S. Harter. A probabilistic approach to automatic keyword indexing. *Journal of the American Society for Information Science*, 26:197–206, 280–289, 1975.
- [29] D. Hawking and N. Craswell. Overview of the trec-2001 web track. In *The Tenth Text REtrieval Conference*, pages 61–67. NIST Special Publication 500-250, 2001.
- [30] D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. In *Proceedings of the Second European Conference on Research and Advanced Technologies for Digital Libraries*, pages 569–584, 1998.
- [31] D. Hiemstra. *Using language models for information retrieval*. PhD thesis, University of Twente, 2001.
- [32] T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.
- [33] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length normalization in xml retrieval. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 80–87, New York, NY, USA, 2004. ACM Press.
- [34] J. kamps, M. de Rijke, and B. Sigurbjörnsson. The importance of length normalization for xml retrieval. *Information Retrieval Journal*, 8:631–654, 2005.
- [35] J. Kamps, G. Mishne, and M. de Rijke. Language models for searching in web corpora. In *The Thirteenth Text REtrieval Conference*. NIST Special Publication 500-261, 2004.

- [36] J. Kim, X. Xue, and W. Croft. A probabilistic retrieval model for semistructured data. In *Proceedings of the 31st European Conference on Information Retrieval*, pages 228–239. Springer, 2008.
- [37] P. Kingsbury, M. Palmer, and M. Marcus. Adding semantic annotation to the penn treebank. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT 2002)*, 2002.
- [38] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 27–34. ACM, 2002.
- [39] J. Lafferty and C. Zhai. *Language Modeling in Information Retrieval*, chapter Probabilistic Relevance Models Based on Document and Query Generation. Kluwer International Series on Information Retrieval. Kluwer, 2003.
- [40] R. Larson. A fusion approach xml structured document retrieval. *Information Retrieval Journal*, 2005.
- [41] V. Lavrenko. *A Generative Theory of Relevance*. PhD thesis, University of Massachusetts at Amherst, 2004.
- [42] J. Lin and B. Katz. Building a reusable test collection for question answering. *Journal of the American Society for Information Science and Technology*, 57(7):851–861, 2006.
- [43] J. List, V. Mihajlović, A. P. de Vries, G. Ramirez, and D. Hiemstra. The tijah xml-ir system at inex 2003. In *INEX 2003 Workshop Proceedings*, pages 102–109, 2003.
- [44] W. Lu, S. Robertson, and A. MacFarlane. Field-weighted xml retrieval based on bm25. *Advances in XML Information Retrieval and Evaluation (INEX 2005)*, LNCS 3977:161–171, 2006.
- [45] Y. Lu, J. Hu, and F. Ma. SjtU at trec 2004: Web track experiments. In *The Thirteenth Text Retrieval Conference (TREC 2004)*, 2005.
- [46] Y. Mass and M. Mandelbrod. Retrieving the most relevant xml components. In *Proceedings of the INEX Workshop 2003*, pages 58–64, 2003.
- [47] Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for XML retrieval. In *Advances in XML Information Retrieval (INEX 2005)*, pages 73–84, 2005.

- [48] Y. Mass and M. Mandelbrod. Using the inex environment as a test bed for various user models for xml retrieval. In *Advances in XML Information Retrieval and Evaluation (INEX 2005)*, pages 187–195, 2006.
- [49] Y. Mass, M. Mandelbrod, E. Amitay, D. Carmel, Y. S. Maarek, and A. Soffer. Juruxml - an xml retrieval system at inex'02. In *Proceedings of the INEX Workshop 2002*, pages 73–80, 2003.
- [50] D. Metzler. Direct maximization of rank-based metrics. Technical report, Center for Intelligent Information Retrieval, University of Massachusetts, 2005.
- [51] D. Metzler. Using gradient descent to optimize language modeling smoothing parameters. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 687–688. ACM Press, 2007.
- [52] D. Metzler and W. Croft. Combining the language model and inference network approaches to retrieval. *Information Processing and Management*, 40:745–750, 2004.
- [53] D. Metzler, V. Lavrenko, and W. Croft. Formal multiple-bernoulli models for language modeling. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 540–541, New York, NY, USA, 2004. ACM Press.
- [54] D. Metzler, T. Strohman, H. Turtle, and W. Croft. Indri at trec 2004: Terabyte track. In *Proceedings of 2004 Text REtrieval Conference (TREC 2004)*, 2005.
- [55] D. Metzler, T. Strohman, Y. Zhou, and W. Croft. Indri at trec 2005: Terabyte track. In *Proceedings of 2005 Text REtrieval Conference (TREC 2005)*, 2006.
- [56] V. Mihajlović, G. Ramírez, A. P. de Vries, D. Hiemstra, and H. E. Blok. Tijah at inex 2004 modeling phrases and relevance feedback. In *Advances in XML Information Retrieval (INEX 2004)*, pages 276–291, 2004.
- [57] M. Montague and J. A. Aslam. Relevance score normalization for metasearch. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 427–433, New York, NY, USA, 2001. ACM Press.
- [58] S. Myaeng, D. Jang, M. Kim, and Z. Zhoo. A flexible model for retrieval of SGML documents. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 138–145. ACM Press, 1998.
- [59] G. Navarro and R. Baeza-Yates. Proximal nodes: a model to query document databases by content and structure. *ACM Trans. Inf. Syst.*, 15(4):400–435, 1997.

- [60] P. Ogilvie. Retrieval using structure for question answering. In *Proceedings of the First Twente Data Management Workshop*, pages 15–23, 2004.
- [61] P. Ogilvie and J. Callan. Hierarchical language models for xml component retrieval. In *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004*, pages 224–237. Springer-Verlag, 2005.
- [62] P. Ogilvie and J. P. Callan. Experiments using the lemur toolkit. In *The Tenth Text REtrieval Conf. (TREC-10), NIST SP 500-250*, pages 103–108, 2002.
- [63] P. Ogilvie and J. P. Callan. Combining document representations for known-item search. In *Proc. of the 26th annual int. ACM SIGIR conf. on Research and development in informaion retrieval (SIGIR-03)*, pages 143–150, New York, July 28– Aug. –1 2003. ACM Press.
- [64] P. Ogilvie and J. P. Callan. Combining structural information and the use of priors in mixed named-page and homepage finding. In *The Twelfth Text REtrieval Conf. (TREC-12)*, 2004.
- [65] PaulOgilvie and J. Callan. Experiments with language models for known-item finding of e-mail messages. In *The Fourteenth Text Retrieval Conference (TREC-14)*, 2006.
- [66] D. Petkova and W. Croft. Hierarchical language models for expert finding in enterprise corpora. In *Proceedings of the 18th International Conference on Tools with Artificial Intelligence*, pages 599–606, 2006.
- [67] J. Ponte and W. Croft. A language modeling approach for information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281. ACM Press, 1998.
- [68] S. Pradhan, W. Ward, K. Hacioglu, J. H. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of NAACL-HLT 2004*, 2004.
- [69] V. V. Raghavan and S. K. M. Wong. A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, 37(5):279–287, 1986.
- [70] J. C. Reynar and A. Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 1997.
- [71] S. Robertson. The probability ranking principle in ir. *Journal of Documentation*, 33:294–304, 1977.

- [72] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–127, 1994.
- [73] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York, NY, USA, 2004. ACM Press.
- [74] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.
- [75] G. Salton, E. A. Fox, and H. Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036, 1983.
- [76] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.
- [77] B. Sigurbjörnsson and J. Kamps. The effect of structured queries and selective indexing on xml retrieval. In *Advances in XML Information Retrieval and Evaluation (INEX 2005)*, pages 104–118, 2006.
- [78] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An element-based approach to xml retrieval. In *The INEX 2003 Workshop Proceedings*, pages 19–24, 2004.
- [79] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Mixture models, overlap, and structural hints in xml element retrieval. In *Advances in XML Information Retrieval (INEX 2004)*, pages 104–109, 2004.
- [80] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Processing content-oriented xpath queries. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 371–380, New York, NY, USA, 2004. ACM.
- [81] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the Nineteenth Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, 1996.
- [82] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 623–632, New York, NY, USA, 2007. ACM.

- [83] R. Song, J. Wen, S. Shi, G. Xin, T. Liu, T. Qin, X. Zheng, J. Zhang, G. Xue, and W. Ma. Microsoft research asia at web track and terabyte track of trec 2004. In *The Thirteenth Text REtrieval Conference*. NIST Special Publication 500-261, 2004.
- [84] T. Strohman, D. Metzler, and W. Croft. Indri: A language model-based search engine for complex queries (extended version). Technical report, Center for Intelligent Information Retrieval, University of Massachusetts, 2005.
- [85] A. Trotman and B. Sigurbjörnsson. Narrow Extended XPath I. Technical report, INEX, 2004. Available at <http://inex.is.informatik.uni-duisburg.de:2004/>.
- [86] H. Turtle and W. Croft. Inference networks for document retrieval. In *Proceedings of the Thirteenth Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 1–24, 1990.
- [87] J.-N. Vittaut and P. Gallinari. Machine learning ranking and inex’05. *Advances in XML Information Retrieval and Evaluation (INEX 2005)*, LNCS 3977:336–343, 2006.
- [88] J.-N. Vittaut, B. Piwowarski, and P. Gallinari. An algebra for structure queries in bayesian networks. *Advances in XML Information Retrieval and Evaluation (INEX 2004)*, 2005.
- [89] L. Wasserman. *All of Statistics*. Springer, 2004.
- [90] T. Westerveld, W. Kraaij, and D. Hiemstra. Retrieving web pages using content, links, URLs, and anchors. In *The Tenth Text REtrieval Conf. (TREC-10)*, NIST SP 500-250, pages 663–672, 2002.
- [91] R. Wilkinson. Effective retrieval of structured documents. In *Proceedings of the Seventeenth Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 311–317, 1994.
- [92] J. Wolff, H. Flörke, and A. Cremers. Searching and browsing collections of structural information. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries (ADL)*, 2000.
- [93] J. Xu and J. Callan. Cluster based language models for distributed retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 254–261. ACM Press, 1999.
- [94] H. Zaragoza, N. Craswell, M. Taylor, S. Sarria, and S. Robertson. Microsoft cambridge at trec 13: Web and hard tracks. In *The Thirteenth Text REtrieval Conference*. NIST Special Publication 500-261, 2004.

- [95] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342. ACM Press, 2001.
- [96] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 2(2), April 2004.
- [97] L. Zhao and J. Callan. A generative retrieval model for structured documents. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1163–1172, New York, NY, USA, 2008. ACM.



# NEXI query language

The content-oriented XPath query syntax described here is called NEXI [85]. Although other filters such as numeric operations exist in NEXI, the discussion here of NEXI focuses on the `about` relevance filter as it is more intimately related to ranking than the other filters. Paths prefixed `xp` take the form `(//type)+` where `type` can be any of the element types in the corpus or `*` indicating that any element type is satisfactory. The `//` denotes a descendant relationship.

In pattern XP1, the query expresses the request that elements matching path `xp1` with descendant elements matching path `xpA` that are about the query `qtA` be returned to the user. For example, the query

```
A.1 //front_matter[ about(./title, bombings) ]
```

requests that the system return `front_matter` elements that have a `title` element about `bombings`. Any path within an `about` filter may simply be the character `'.'` which indicates that the query expressed in the filter should match the outer element itself. The query

```
A.2 //paragraph[ about(., number suicide bombings 2006) ]
```

requests that paragraphs discussing the number of suicide bombings in 2006 be ranked and returned to the user.

---

Label	Pattern	Description
XP1	<code>xp1[about(.xpA, qtA)]</code>	elements matching <code>xp1</code> containing element <code>xpA</code> which is about <code>qtA</code>
XP2	<code>xp1[about(.xpA, qtA) op about(.xpB, qtB)]</code>	<code>xp1</code> must have <code>xpA</code> about <code>qtA</code> and/or <code>xpB</code> about <code>qtB</code>
XP3	<code>xp1[about(.xpA, qtA)]xp2[about(.xpB, qtB)]</code>	<code>xp1.xp2</code> having <code>xpB</code> about <code>qtB</code> and <code>xp1</code> has <code>xpA</code> about <code>qtA</code>

---

Table A.1: Summary of some content-oriented XPath query patterns.

Pattern XP2 shows how conjunctions of about filters may be expressed in NEXI. The operator *op* may be either a Boolean AND or an OR. The query

```
A.3    //article[ about(., bombings) AND  
        about(./image/caption, investigators) ]
```

requests that **article elements about bombings having image captions about investigators** be ranked and returned to the user.

Pattern XP3 shows how elements may be ranked with the consideration of related elements higher up in the document structure hierarchy. The query

```
A.4    //article[ about(., bombings)]//image[  
        about(./caption, investigators) ]
```

requests that **image elements with a caption about investigators in an article about bombings** be ranked and returned.

# *IEEE Elements Retrieved*

---

We restrict our experiments in the IEEE collections to retrieving a subset of all XML element types for efficiency reasons. The list was constructed by taking all element types that had at least 100 relevant elements across the test collections. We found in our experiments that adding more element types only improved the MAP of the systems, but felt that this list created a reasonable compromise between efficiency and MAP. These element types are sorted by descending frequency: *p, sec, article, it, tmath, ss1, ref, ip1, bb, art, bdy, b, st, sub, item, fig, vt, abs, ss2, bm, tf, fm, li, atl, label, scp, fgc, list, ti, obi, au, en, entry, fnm, url, bib, bibl, snm, yr, super, app, pdt, no, enum, lc, pp, ss, tt, index-entry, row, tbl, loc, cty, p1, mo, tig, la, math, term.*