

# Annotation Persistence over Dynamic Documents

by

Shaomin Wang

Submitted to the Department of Civil and Environmental Engineering  
in Partial Fulfillment of the Requirements for the Degree of

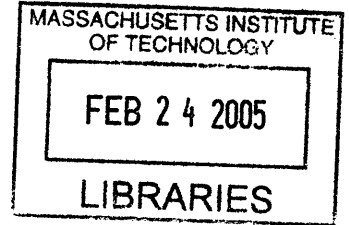
**BARKER**

Doctor of Philosophy in the Field of Information Technology

at the

Massachusetts Institute of Technology

February 2005



© 2005 Massachusetts Institute of Technology  
All right reserved

Signature of Author .....  
Department of Civil and Environmental Engineering  
October 5, 2004

Certified by .....  
Steven R. Lerman  
Professor of Civil and Environmental Engineering  
Thesis Co-supervisor

Certified by .....  
V. Judson Harward  
Principal Research Scientist, Center for Educational Computing Initiatives  
Thesis Co-Supervisor

Accepted by .....  
Andrew Whittle  
Chairman, Departmental Committee on Graduate Studies



# Annotation Persistence over Dynamic Documents

by  
Shaomin Wang

Submitted to the Department of Civil and Environmental Engineering  
On October 5, 2004, in Partial Fulfillment of the Requirement for the Degree of  
Doctor of Philosophy in the Field of Information Technology

## Abstract

Annotations, as a routine practice of actively engaging with reading materials, are heavily used in the paper world to augment the usefulness of documents. By annotation, we include a large variety of creative manipulations by which the otherwise passive reader becomes actively involved in a document. Annotations in digital form possess many benefits paper annotations do not enjoy, such as annotation searching, annotation multi-referencing, and annotation sharing. The digital form also introduces challenges to the process of annotation. This study looks at one of them, *annotation persistence over dynamic documents*.

With the development of annotation software, users now have the opportunity to annotate documents which they don't own, or to which they don't have write permission. In annotation software, annotations are normally created and saved independently of the document. The owners of the documents being annotated may have no knowledge of the fact that third parties are annotating their documents' contents. When document contents are modified, annotation software faces a difficult situation where annotations need to be reattached. Reattaching annotations in a revised version of a document is a crucial component in annotation system design.

Annotation persistence over document versions is a complicated and challenging problem, as documents can go through various changes between versions. In this thesis, we treat annotation persistence over dynamic documents as a specialized information retrieval problem. We then design a scheme to reposition annotations between versions by three mechanisms: the meta-structure information match, the keywords match, and content semantics match. Content semantics matching is the determining factor in our annotation persistence scheme design. Latent Semantic Analysis, an innovative information retrieval model, is used to extract and compare document semantics.

Two editions of an introductory computer science textbook are used to evaluate the annotation persistence scheme proposed in this study. The evaluation provides substantial evidence that the annotation persistence scheme proposed in this thesis is able to make the right decisions on repositioning annotations based on their degree of modifications, i.e. to reattach annotations if modifications are light, and to orphan annotations if modifications are heavy.

Thesis Co-supervisor: Steven L. Lerman  
Title: Professor, Civil and Environmental Engineering

Thesis Co-supervisor: V. Judson Harward  
Title: Principal Research Scientist, Center for Educational Computing Initiatives



# Acknowledgements

---

I would like to thank the following people for their generous support which made this work possible.

Professor Steve Lerman for his wise guidance, encouraging advice, and patience; but most important for being there for me during my stay at MIT.

Dr. Jud Harward for his excellent advice, direction and support. His suggestions on potential thesis topics led me to this very interesting work.

The members of my thesis committee Professor Jerome Connor and Professor Suzanne Flynn for their encouragement and support.

Ms. Ellen Faran, director of MIT Press, for her help on finding books with multiple editorial versions.

The people at the Center for Educational Computing Initiatives.

My parents and my brother for their love and encouragement.

Finally, I wish to acknowledge the intangible contribution of my wife, Jun Yan, to whom this thesis is dedicated.



*This thesis is dedicated to my wife,*

*Jun*





# Contents

---

<b>Abstract</b>	<b>3</b>
<b>Acknowledgements</b>	<b>5</b>
<b>1 Introduction</b>	<b>11</b>
1.1 Annotations: from Papers to Digital Documents	11
1.1.1 Paper Annotation Taxonomy – Forms and Functions	11
1.1.2 The Benefits Digital Format Brings to Digital Annotations	17
1.1.3 The Challenges Digital Annotations Face	18
1.1.4 Digital Annotation Representation	20
1.2 Annotation Systems and Architecture	23
1.2.1 Overview of State-of-Art Annotation Systems and Architecture	24
1.2.2 Representative Annotation Systems	26
1.3 Annotation Persistence Mechanism	34
1.3.1 Document Modifications	34
1.3.2 Robust Criteria of <i>Intra-Document Locations</i>	36
1.3.3 Related Works on Annotation Persistence Mechanism	38
1.4 Annotation Persistence over Dynamic Documents – a Specialized Information Retrieval Problem	48
1.4.1 A Specialized Information Retrieval Problem	48
1.4.2 IR Evaluations and IR Models	49
1.4.3 Problem Revisited – Concepts and Keywords	55
1.4.4 Strategies and Solutions	59
1.5 Summary	63
<b>2 Natural Language Statistics and Entropy Measure of Keywords</b>	<b>70</b>
2.1 Statistical Nature of Natural Language	70
2.1.1 Zipf’s Law	70
2.1.2 Impact of Zips’s Law on Information Retrieval	78
2.2 Information Theory and Entropy Measure of Keywords	80
2.2.1 Information Theory and Shannon’s Entropy	81
2.2.2 Normalized Word Entropy	86
2.3 Summary	88
<b>3 Latent Semantic Analysis</b>	<b>91</b>
3.1 Introduction	92
3.2 Theory Background and Methodologies	95
3.2.1 Term-Document Matrix	95

3.2.2 Singular Value Decomposition .....	100
3.2.3 Words and Documents Representations and Comparisons .....	107
3.2.4 Singular Value Decomposition of Sparse Matrices .....	108
3.3 Applications .....	111
3.4 LSA – Evaluation .....	117
3.4.1 Software Tools .....	118
3.4.2 LSA Flow Chart .....	119
3.4.3 The Corpus .....	122
3.4.4 Computation Cost .....	123
3.4.5 Effectiveness of Retrieval .....	124
3.5 Summary .....	134
<b>4 Design and Evaluation of Robust Annotation Persistence Scheme</b>	<b>136</b>
4.1 Design of Robust Annotation Persistence Scheme .....	139
4.1.1 Annotation Anchor Formulation .....	139
4.1.2 Reattachment Confidence Index .....	145
4.1.3 Design and Calibration of Reattachment Algorithm .....	146
4.2 Evaluation of Robust Annotation Persistence Scheme .....	168
<b>5 Conclusion</b>	<b>188</b>
5.1 Thesis Summary .....	188
5.2 Future Research .....	193
<b>Appendix: Sample Document Retrieval Results of LSA</b>	<b>195</b>
<b>References</b>	<b>212</b>

# Chapter 1

## Introduction

---

### 1.1 Annotations: from Papers to Digital Documents

Annotations, as a routine practice of actively engaging with reading materials, are heavily used in the paper world to augment the usefulness of documents. By annotation, we include a large variety of creative manipulations by which the otherwise passive reader becomes actively involved in a document.

Annotations in digital form possess many benefits paper annotations do not enjoy, such as annotation searching, annotation multi-referencing, and annotation sharing. The digital form also introduces challenges to the process of annotation. This study looks at one of them, *annotation persistence over dynamic documents*.

In the following, we first review the practice of making annotations on paper, the taxonomy of annotation forms, and their functions. We then review the benefits digital format brings to annotations as well as the challenges digital annotations face. We elaborate one of the challenges, which is the focus of this study, *annotation persistence over dynamic documents*. Lastly, we present a generic definition of digital annotation based on Marshall's classification which will be used through out this study.

#### 1.1.1 Paper Annotation Taxonomy – Forms and Functions

There have been a number of studies to examine the practice of paper annotations, in terms of their forms and functions. For example, in a study comparing reading paper and online documents, O'Hara and Sellen (1997) outlined several general annotation forms and functions from the usage perspective. In their opinion, annotation markings could serve as signals of direct short cut into the content for readers when re-reading the documents. They are used to extract key points or structures when re-reading. The very act of making such marks also aids the understanding and remembering the reading

materials. Brief notes written separately for later reference provides “*a pool of text and ideas*”. In a study by Brown and Brown (2003), the general mechanism of annotation is viewed as a key concept of integrating reading with writing.

The most comprehensive study on the taxonomy of paper annotations, though, is from Ovsianikov et. al. (1999) and Marshall (1997), although they approached the problem from very different perspective.

As part of USC’s Brian project, Ovsianikov et. al. studied the taxonomy of paper annotations in a questionnaire targeted mainly at researchers and those in an academic environment. The respondents were graduate and undergraduate students, professors and a few professionals. Three main questions were asked. How do people annotate papers? How are the annotations used once they have been created? What are the features ideal annotation software must have?

Marshall approached the problem from a different perspective. She examined the used textbooks of college students from a cross section of courses and disciplines.

In the following, we summarize their findings.

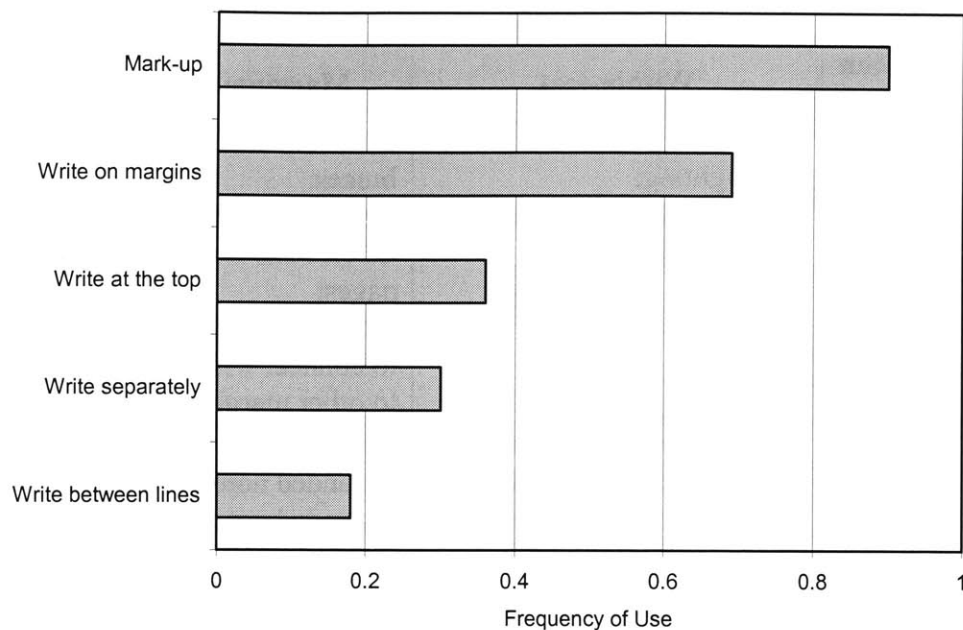
### **Annotation Forms**

After the examination of fifteen different sets of used college student textbooks, over 150 books in all, Marshall classified paper based annotations into four groups based on whether the annotation is within-text (e.g. highlighting text, circled words) or it is in the margins (e.g. scribbled notes in a margin, asterisks and stars) and whether the annotation is explicit in meaning (e.g. brief notes) or opaque personal coding (e.g. red underlining indicating importance). Table 1.1 is the annotation taxonomy developed by Marshall in terms of annotation forms.

<b>Location Content</b>	<b>Within-text</b>	<b>Marginal or blank space</b>
<b>Telegraphic</b>	Underlining; Highlighting; Circles and boxes around words and phrases	Brackets, angle brackets, and braces; Asterisks, and stars; Circles and boxes around whole pages; Arrows and other deictic devices to connect within-text markings to other marginal markings
<b>Explicit</b>	Brief notes written between lines, especially translations of words in foreign language texts	Short phrases in margin; Extended notes in margin; Extended notes on blank pages in the front of the book; Problems worked in margins

**Table 1.1** Form of annotations written in books (Marshall, 1997)

The survey conducted by Ovsiannikov et. al. (1999) revealed that the common types of the annotations on papers, rated by the frequency of usages, are text mark up, writing on margins, writing at the top, writing separately from the paper, and writing between lines (Fig. 1.1)



**Figure 1.1** Annotation forms (Ovsiannikov et. al., 1999)

The most common annotation type is to highlight portions of text with a marker. It is used to highlight key ideas and concepts in the paper. It helps reader to memorize the contents. In the future, when readers re-read the paper, it draws the reader's attention and helps to quickly recollect the paper's main ideas.

Writing on the margins comes as the close second in terms of usage. In the process of reading a paper, readers can come up with their own ideas, critical remarks, questions and notes reflecting their opinions on the subject. Margins have space to record readers' thoughts next to the annotated text.

The three less popular ways of making annotations are writing at the top of the documents, making notes separately from the original paper and writing between lines. Writing on the top of the documents or making notes separately often serves as a summary of a paper. The notes are further distanced from the local context of the paper, thus requiring a higher level of engagement with the paper and additional mental effort, which explains its lower popularity in practice.

Writing between the lines or crossing out phrases is very common in the paper authoring and editing process.

Although Marshall did not rate the importance or preferences of each annotation form in her taxonomy, she presented a classification which separates the annotation contents from annotation locations. As each annotation is a marking made on a document at a particular place, a generic annotation representation can be composed by its content and its location. Annotation content, from the taxonomy of Marshall, can be implicit or explicit, while its location can be within-text or marginal. We will give a generic definition of digital annotation representation in the end of this section based on Marshall's annotation form classification.

### **Annotation Functions**

Marshall reconstructed annotation functions from the material evidence of the used textbook annotations. She summarized that annotation usually serves one of the following functions:

- *procedural signals for future references*
- *place markings and aids to memory*
- *in situ locations for problem working*
- *a record of interpretive activity*
- *a visible trace of the reader's attention*

Marshall also mapped the annotation forms into functions shown in Table 1.2

Form	Function
Underlining or highlighting higher-level structures (like section headings); telegraphic marginal symbols like asterisks; cross outs.	Procedural signaling for future attention
Short highlighting; circled words or phrases; other within-text markings; marginal markings like asterisks.	Place marking and aiding memory
Appropriate notation in margins or near figures or equations	Problem working
Short notes in the margins; longer notes in other textual interstices; words or phrases between lines of text.	Interpretation
Extended highlighting or underlining	Tracing progress through difficult narrative
Notes, doodling, drawings, and other such markings unrelated to the materials themselves	Incidental reflection of the material circumstances of reading

**Table 1.2** Mapping annotation form into function (Marshall, 1997)

The survey conducted by Ovsianikov et. al. revealed four primary annotation usages after annotations are created: to remember, to think, to clarify and to share.

People tend to forget the contents of papers. Annotations can serve as “indexes” to quickly recollect the main points of the paper. They help readers memorize the paper’s contents.

In the process of reading a paper, readers can come up with their own ideas, critical remarks, questions and notes reflecting their opinion on the subject. The very act of writing those ideas out on the margin helps people to think.

Apart from markups helping readers to memorize and writing on margins helping readers to think, the author argues that sometimes readers are interested in rephrasing the contents into their own words by writing in margins, thus personifying contents by clarification.

The fourth usage of annotations is annotation sharing, which is common in group collaboration. This is very brief given that the usage is so different and that digital annotation makes this more effective.



It is interesting to see that although Marshall and Ovsianikov come up with different lists of annotation functions, they are indeed equivalent. They are just ways of making a taxonomy of annotation functions from different perspectives.

### **1.1.2 The Benefits Digital Format Brings to Digital Annotations**

Having documents in digital format has brought unprecedented benefits over their paper counterparts. Digital documents are easier to edit, reproduce, distribute and search.

Having annotations in digital form will confer upon documents many benefits that paper annotations do not enjoy. The following are particularly important.

- Annotations can be searched.

As we reviewed earlier, annotations are used as means to quickly recollect the main ideas or structures of the documents. They serve as “indexes” to quickly find the important ideas and keywords, much like a book index, although they are dispersed throughout the document. For digital annotations, searching annotations is an automated and convenient process. When searchable, annotations can be accessed not by their positions in a paper, but rather by their content.

- Annotations can themselves be annotated.

When in digital format, annotation itself can become a thread of interest, thus become a target for annotation. This could become very useful in group collaborations.

- Annotations can point to multiple locations in a document, or even multiple locations in multiple documents.

When we read papers or books, we see similar keywords, and related concepts can appear in multiple places in the same document. Paper annotations restrict the ability to reference our annotations to multiple places of interests. In digital annotation, this restriction no longer exists. Annotations can reference not only multiple places in one document, but also can reference multiple places in many documents. This function greatly expands our reach when making annotations. It puts the annotation in the context of the entire set of documents of our interest, rather than the context of one location in one document.

- Annotation itself can be multimedia.

Since we are only interested in annotations on text in this study, however, the annotation itself can be multimedia. It could include text, images, or even video clips.

- Annotations can contain hypertext links.

In paper annotation, we are restricted by the space of the physical material. Our annotations have to be concise or even telegraphic. In digital annotations, this restriction no longer exists. Not only can we put much larger contents into our annotations, but also our annotation contents can include hypertext links to link to outside materials of interests.

- Annotations can be shared.

In paper annotations, sharing can only happen with the physical passing or copying of the paper material. In digital annotations, sharing becomes a convenient feature. In fact, many annotation systems developed so far have annotation sharing as an important goal.

### **1.1.3 The Challenges Digital Annotations Face**

Just as digital format can bring unprecedented benefits to annotations over their paper counterparts, digitization also brings unique challenges to digital annotations. Such challenges include the following:

Digital annotations need to be highly expressive. Annotations on paper are highly expressive and individual in form; digital annotation should respect this fluidity. This goes against the idea of using a palette of common symbols, colors and pen types. It suggests a more freeform capability is needed.

Digital annotations should be format independent. As more and more document formats are developed, a robust digital annotation system should be immune to document format change.

The focus of this study, another challenge, is *annotation persistence over dynamic documents*. In the following, we elaborate on this issue.

### **Annotation Persistence over Dynamic Documents**

With the development of annotation software, quite different from paper annotations, users now have the opportunities to annotate documents which they don't own, or to

which they don't have write permission. This creates both benefits and challenges. Unlike paper annotations where only one person can annotate his own document at a time, in digital annotation, anyone can annotate the same document at the same time without interference. Theoretically, any user can annotate any document within his/her reach (in the networked sense) without owning the document. In annotation software, annotations can be created and saved independently of the document. The documents being annotated could have no knowledge of the fact that third parties are annotating their contents. A document intra-location referencing mechanism is usually created in the third party annotation software, which is used to position annotations in the documents when the annotation software merges the annotations with the documents.

This is very similar to hypertext linking, where the webpage link targets have no knowledge of the pages that link to them. Thus when a target changes its URL, the links to them become broken. In digital annotations, when document contents are edited, the initial text and surrounding context that third party annotations reference could also be changed, which can lead annotations to become "unattached". When annotation fails to be reattached to the document when document text is edited or changed, the annotation is "orphaned". We name this problem as "annotation persistence over dynamic documents", which is the focus of this study.

In paper books, we often make a considerable number of annotations as we actively engage with the materials. If a second version of the same book is published, we face a frustrating situation. We have to keep both versions because the first version contains our valuable notes on the main ideas and concepts of our interest. If the books are in digital form, could we move annotations on the first version automatically and "intelligently" over to the second version?

World Wide Web presents us enormous amount of information. We make annotations on some of the webpages of interest. When the contents of the webpages are edited, could we reattach our annotations "gracefully" to the newer contents of the webpages without losing our original annotations?

This is the problem we are going to address. We assume the document changes are uncoordinated, that is we have no knowledge when or where the document contents are changed. Thus it is infeasible to save the record of all the changes the document has gone

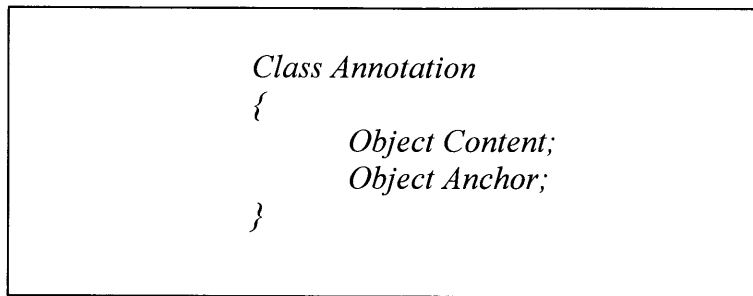
through. At the time of the annotation reattachment, we only know two states of the document, the initial state when the annotations were made and the current state when the document contents have been changed. The solution to this problem is to develop a “robust” mechanism that could “intelligently and gracefully” reattach annotations when document contents are changed.

Many existing annotation systems silently leave the “unattachable” annotations “orphaned”, either display them at the end of the document or put them at the bottom of the browser.

A large-scale annotation software study was conducted by Cadiz J.J. et. al. (2000) using Microsoft Office 2000. Approximately 450 people created 9000 shared annotations on about 1250 documents over 10 months. When studying the factors that influence system usage, it turned out that the primary reason people stopped using the entire system was annotation orphaning. Annotation orphaning is an understandably frustrating problem. As put it by Cadiz J.J. et. al. (2000) in his paper, *“The power of annotations stems from being context-based, and they are worded with the context assumed. Without the context, many annotations are useless. From the annotator’s standpoint, it can be extremely frustrating to take the time to comment on a document, only to see the comments become meaningless through orphaning.”*

#### **1.1.4 Digital Annotation Representation**

Annotation on multimedia requires different techniques for different media type. In this study, we are only interested in annotation on text. An annotation is a marking made on a document at a particular place. A generic annotation representation can be composed by its content and its location (or anchor) inside a document. Annotation content, based on the taxonomy of Marshall, can be implicit or explicit, while its location can be within-text or marginal. Here we use “anchor” instead “location” to represent the information that is used to address into the document. Adopting object oriented programming semantics, an object of “annotation” contains an object of “content” and an object of “anchor” (Figure 1.2)

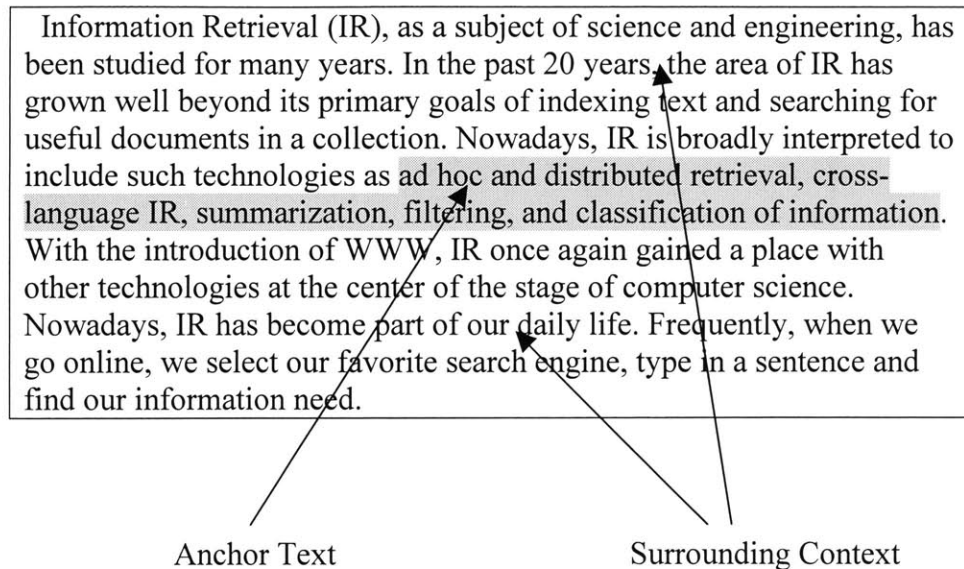


**Figure 1.2** Annotation class compositions

Each anchor, based on whether it is within-text or marginal, can contain anchor text and surrounding context.

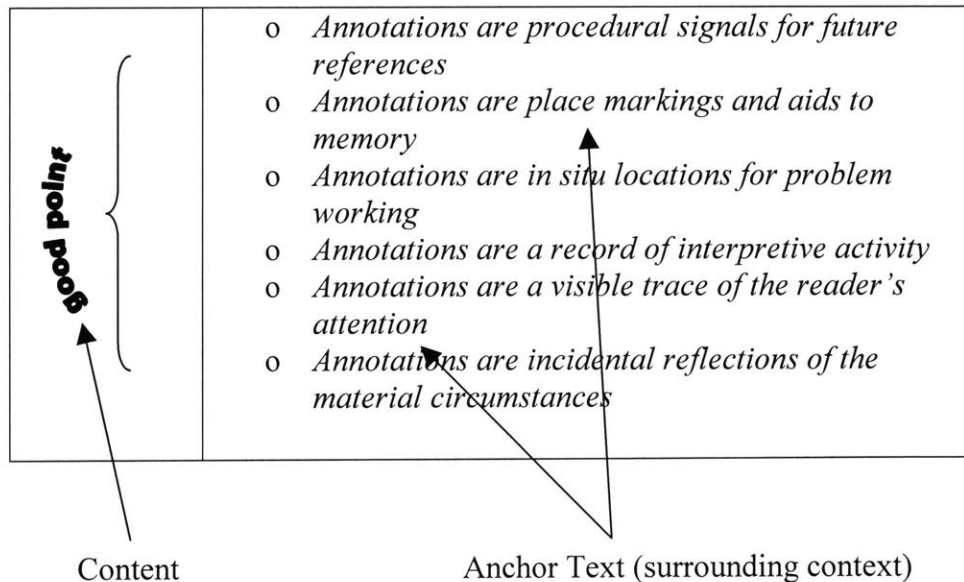
Figure 1.3 is an example of annotation with implicit content (highlighting) and within-text anchor (highlighted text). We define the highlighted text as “anchor text” and surrounding text as “surrounding context”.

Highlight with within-text anchor:



**Figure 1.3** Annotation example: highlight with within-text anchor

Figure 1.4 is an example of an annotation with explicit content (margin notes) and marginal anchor (curly prentices).

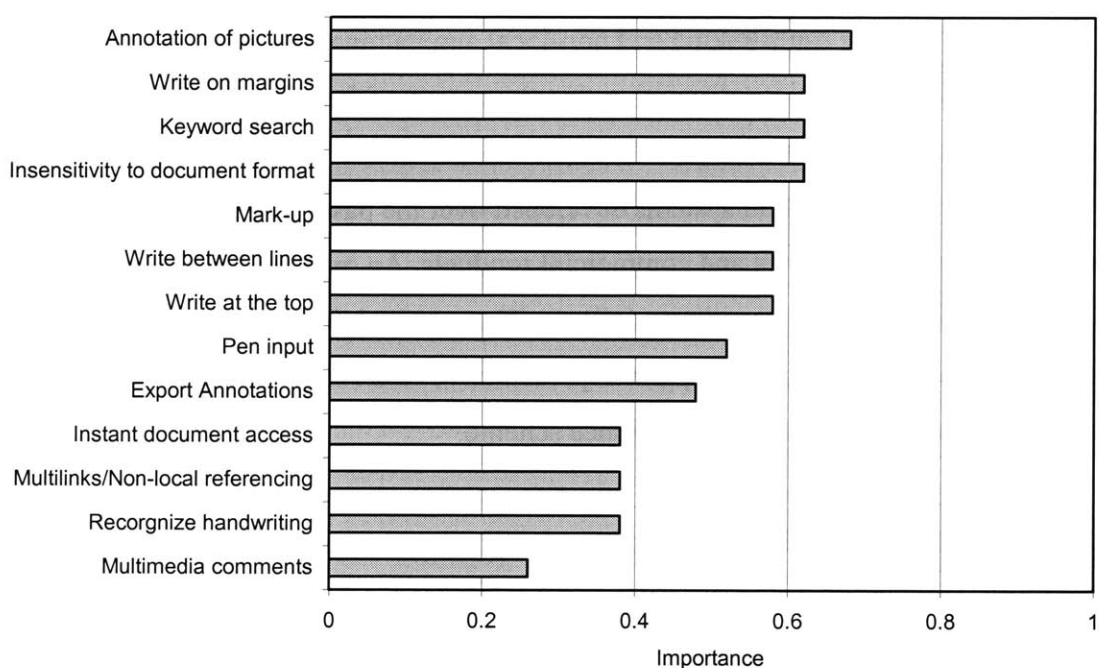


**Figure 1.4** Annotation example: comments with margin anchor

## 1.2 Annotation Systems and Architecture

What features should ideal annotation software have? We expect digital annotations to present at least the equivalent visual expressive power to their paper counterparts. In functionality, we expect digital annotations to enjoy many benefits that paper annotation lacks.

Figure 1.4 shows the comparison of various annotation features favored by the respondents in the survey conducted by Ovsianikov et. al. (1999).



**Figure 1.4** Important annotation software features (Ovsianikov et. al. 1999)

The ability to markup pictures or figures tops the respondents' lists of preferred functions. Pictures or figures usually carry important semantic information about the document. On paper, readers can easily annotate pictures or figures along with the text, but annotating pictures in software requires different technology and algorithms than annotating text. These methods remain to be developed. Hence in this study, we confine our interest to annotating text only.

Writing on the margin, the ability to search annotations, and being able to handle multiple document formats together come as close seconds in the survey.

The next three features are the abilities to markup text, write between lines and write at the top of the document.

It is interesting to note the ability for pen-based input and handwriting recognition don't come near the top of the list, since handwriting comments and drawings are very common and convenient in paper annotations. The author attributes this to the fact that proficient text typing may often be faster and easier than handwriting in a computer environment.

The ability to make multilinking and non-local annotations also appears on the list. It is consistent with our previous discussion that this is part of the advantages of digital annotations over paper annotations.

There are many annotation systems developed over the past ten years, both in academic researching settings and commercial products. An overview of the literature on the state-of-the-art annotation systems follows. Then we look in more detail at a few annotation systems, which are unique or representative either in system architecture design or their robust annotation persistence scheme.

### **1.2.1 Overview of State-of-Art Annotation Systems and Architecture**

Virtually all commercial document-processing software (e.g. Microsoft Word, Lotus Notes, Adobe Reader) supports some form of annotations. Microsoft Word allows users to highlight a portion of the contiguous text and to use the command "Insert-comment" to add footnote-like annotations in a separate window. When a user points the mouse to the highlighted text, a pop-up box shows the name of the user who created the annotation and the annotation contents. In Microsoft Word, annotations are stored within the document file. Users must have the write permission to add annotations. Collaborations can only be achieved by passing the document file to other users. Lotus Notes allow discussions around a document over a network, but comments can only be made on the document as a whole, and not to individual sentences or paragraphs.

Similar to Microsoft Word, Re:mark (AMBIA, 1996) is a commercial plug-in to Adobe Reader to allow annotations on PDF files. With Re:mark, users can add text notes,



draw, color-highlight and strikeout right on the document. Users can link files, such as sound clips, to the document. All linked files and comments become part of the document.

With the web, several companies and research institutes have created client-server systems that provide the ability to annotate web pages (Roscheisen et. al., 1995; Davis and Huttenlocher, 1995; Ovsianiko et. al., 1999; Kahan and Koivunen 2001; Phelps and Wilensky, 1997; Yee, K-P; Third Voice; HyperNix; NovaWiz; uTok; Zadu;E-quill)

One of the early works on annotation systems, which is built on NCSA Mosaic, is the ComMentor (Roscheisen et. al., 1995), developed by the Stanford Integrated Digital Library Project. In this system, annotations are considered as the third-party, lightweight meta-information. The meta-information is stored separately from the main documents in a so-called meta-information server. For the user, ComMentor provides the ability to highlight a piece of text in a page and attach a small image with the user's face on it. This image serves as a link to the annotation.

To avoid potential modification in web pages, CoNote (Davis and Huttenlocher, 1995) developed at Cornell University, allows users to annotate only predefined positions. CoNote requires inserting special HTML-like markup tags before a document can be annotated. In CoNote, the user's permissions to create, delete, read or reply to annotations are determined by the user's role, such as viewer, reader, user or author. The system also supports authentication and annotation search. The authors provide evidence that CoNote use improved class performance and established a greater sense of community among students.

CritLink (Yee, K-P) uses a proxy-based approach to render annotations on the web. In a proxy-based approach, annotations are stored and merged with a web document by a proxy server; the browser user only sees the result of the merge. Typically, presentation of the annotations is limited to the presentation styles available through HTML. The proxy approach inherently restricts the presentation styles that can be used for annotations.

More systems use a browser-based approach to render annotations. In a browser-based approach, the browser is enhanced (either by an external application or plug-in) to merge the document and the annotation data just prior to presenting the content to the

user. Annotation data are stored in a proxy or a separate annotation server. Work referenced in Kahan and Koivunen 2001; Phelps and Wilensky, 1997 and Third Voice all fall into this category.

One company, Third Voice, drew considerable initial attention with its software, but has been hindered by concern that their system allows undesirable graffiti to be posted on major web sites. Third Voice uses plug-ins to enhance the web browsers. Annotations are displayed as side notes to the web page. Users can annotate any web page or some text on the page with discussions on selected topics. The discussion can be closed to a group of participants or open to anyone. Annotation representation and hosting are proprietary to the company.

Another recent system, which allows annotations for streaming video content on the web, is MRAS from Microsoft Research (Barger et. al., 1999). The system allows controlled sharing based on annotation sets and user groups. It supports text and audio annotations and it uses email for notification.

There are many commercial annotation systems developed during the Internet bubble times (HyperNix; NovaWiz; uTok; Zadu; Equil). They now either no longer exist (HyperNix; NovaWiz; uTok; Zadu) or are being acquired (Equil). Their systems are like the systems we have reviewed above in terms of the functioning and system design.

### **1.2.2 Representative Annotation Systems**

We now turn our attention to a few systems which are unique in their design principles and represent the forefront research in annotation systems.

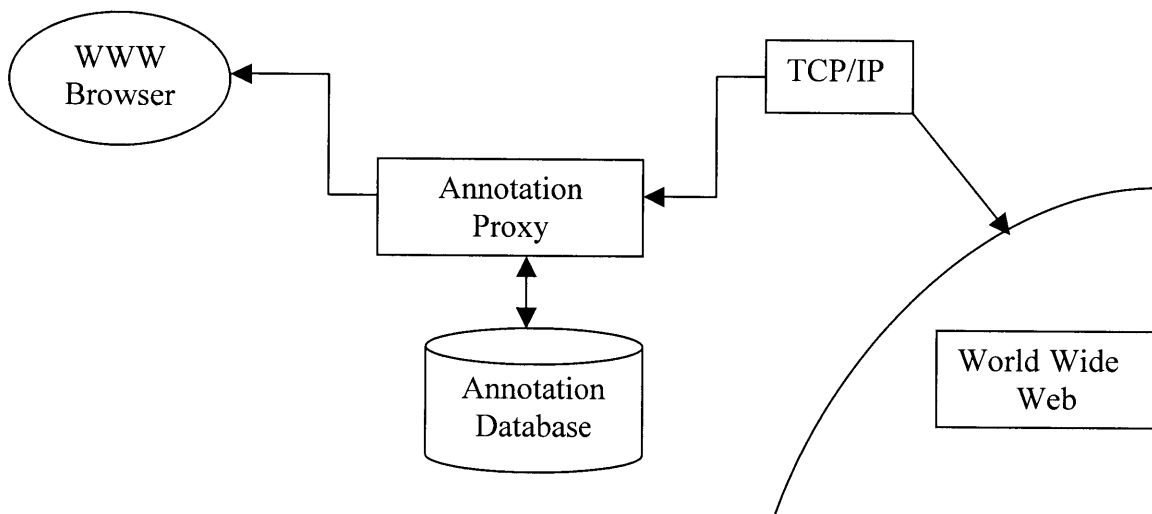
#### **Annotation technology**

As part of the Brain Project in the University of Southern California, a software prototype “Annotator” was developed to make on-line annotations on the World Wide Web (Ovsianniko et. al., 1999).

Annotator adopts a proxy-based architecture as shown in the following Figure 1.5. A Java plug-in must be installed on Netscape or Internet Explorer to view the web documents along side the annotations.

The client browser is configured to send all its HTTP requests to the annotation proxy, which forwards the requests to the appropriate web server. At the same time, the proxy contacts the annotation database to fetch the related annotation records, if any. The annotation records are merged with the returned web documents on the fly and are returned to the client browser.

When the client browser posts to the proxy server trying to add or modify annotations, the proxy parses the file, and extracts and saves the related annotation records to annotation database.



**Figure 1.5** Proxy-based architecture

Users of the “Annotator” can perform the following functions with annotations:

- Creating, modifying and deleting annotations.
- Viewing annotations along with documents
- Indexing and browsing annotations
- Searching annotations
- Referencing one continuous chunk of document text or multiple discontinuous and non-local document texts in a annotation
- Sharing annotations

Each Annotation record contains an annotation content record and multiple (in the case of multilinking and non-local referencing) annotation anchor records.

To implement the principle of document format independence, the anchor representation does not rely on document markup or structures; it contains only a portion of the text in the anchor's proximity (Figure 1.6). The text strings are subsequently hashed and saved as part of an annotation anchor record. A string or sub-string match of anchor text with the document is performed to position the annotation anchors.

Unique ID	2591
URL	<a href="http://bsl.usc.edu/papers/bg.html">http://bsl.usc.edu/papers/bg.html</a>
Color	0000FF
Font Style	Italic
Text	"Our proposal that the basal ganglia ..."
Keywords	Basal ganglia, inhibition

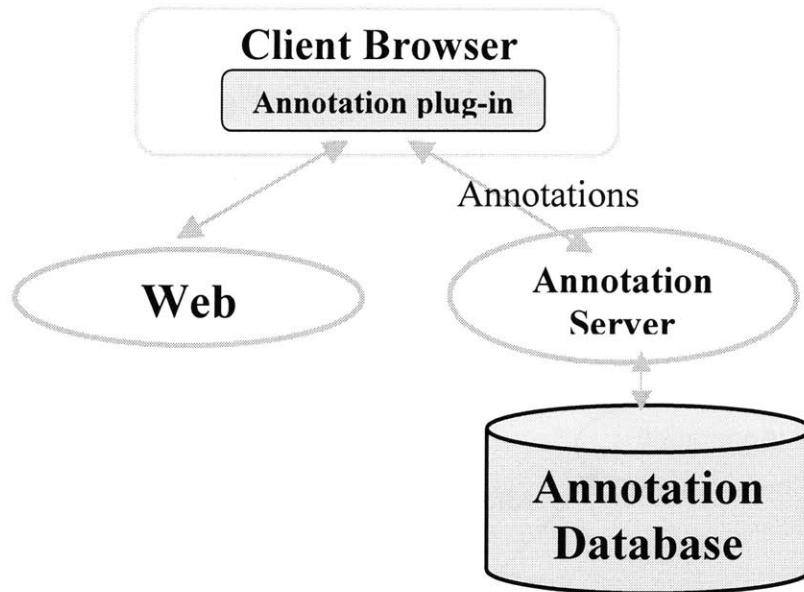
**Figure 1.6** "Annotator" annotation records

### **Annotea**

"Annotea" (Kahan and Koivunen, 2001) is a W3C collaborative web annotation project based on general-purpose open metadata infrastructure where the annotations are modeled as a class of metadata. It uses mostly W3C open source technologies, such as RDF, XLink, XPointer and HTTP.

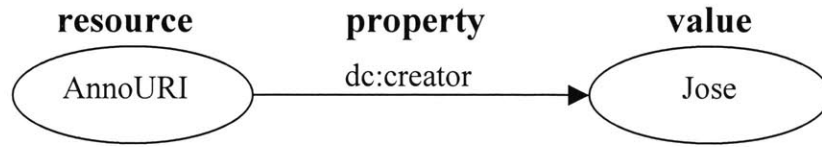
The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web. XLink is a language which allows elements to be inserted into XML documents in order to create and describe links between resources. It uses XML syntax to create structures that can describe links similar to the simple unidirectional hyperlinks of today's HTML, as well as more sophisticated links. XPointer, which is based on the XML Path Language (XPath), supports addressing into the internal structures of XML documents. It allows for examination of a hierarchical document structure and choice of its internal parts based on various properties, such as element types, attribute values, character content, and relative position.

“Annotea” adopts a browser-based architecture where the browser is modified to merge the web documents and the annotation data just prior to presenting the content to the user (Figure 1.7). The client browser uses the Amaya editor/browser, an open source program developed by W3C that supports HTML and a variety of XML markup schemas.



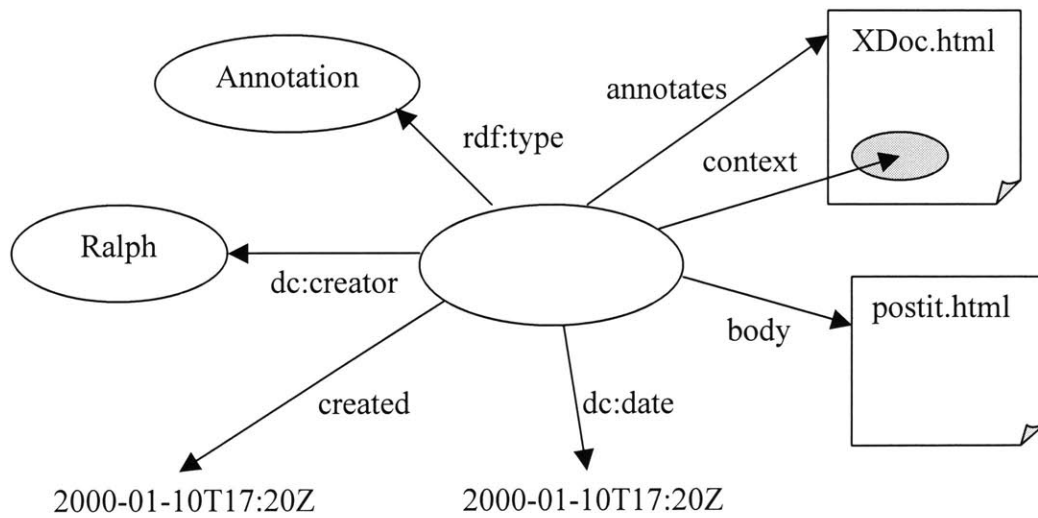
**Figure 1.7** Browser-based architecture

In “Annotea”, annotations are treated as the statements made by a third party about a web resource. Thus they are metadata about a document. Resource Description Framework (RDF) is used as the metadata language to represent annotations. RDF is a language for representing information about resources in the WWW. At its most simple level, RDF provides (resource, property, value) triples (Figure 1.8). A single triple is a statement that indicates that a *resource* has a given *property* with a given *value*. Figure 1.8 could read as: resource “AnnoURI” has a property of “creator” with a value of “Jose”.



**Figure 1.8** RDF triple model

A RDF representation of the annotations is presented in Figure 1.9. It is a type of “Annotation” defined in RDF namespace. It is created by “Ralph” at the date of “1/10/2001”. It annotates the context of “XXX” of the document of “Xdoc.html”. The annotation body (content) is “postit.html”.



**Figure 1.9** The RDF model of an annotation

The schema definition for properties of RDF models is defined in Table 1.3

<i>rdf:type</i>	An indication of the creator's intention when making an annotation; the values should be of <i>rdf:type Annotation</i> or any of its subclasses
<i>annotates</i>	The relation between an annotation resource and the resource to which the annotation applies
<i>body</i>	The content of the annotation
<i>context</i>	Context within the resource named in <i>annotates</i> to which the annotation most directly applies. Eventually this will be an XPointer. It may include a location range too.
<i>dc:creator</i>	The creator of the annotation
<i>created</i>	The date and time on which the annotation was created
<i>dc:date</i>	The date and time on which the annotation was last modified
<i>related</i>	A relation between an annotation and a (collection of) resource(s) that augment the resource that is the <i>body</i> of the annotation. This may point to related issues, discussion threads, etc.

**Table 1.3** The basic annotation properties

Annotea defined a general annotation super class in RDF schema (<http://www.w3.org/2000/10/annotation-ns#Annotation>), and several sample subclasses are defined as well based on the annotation super class (Table 1.4). Users or groups may need to create new subclasses based on their needs.

<i>Annotation</i>	A super class describing the common features of annotations
<i>Advice</i>	A subclass of <i>Annotation</i> representing advice to the reader
<i>Change</i>	A subclass of <i>Annotation</i> describing <i>annotations</i> that document or propose a change to the source document
<i>Comment</i>	A subclass of <i>Annotation</i> describing annotations that are comments
<i>Example</i>	A subclass of <i>Annotation</i> representing examples
<i>Explanations</i>	A subclass of <i>Annotation</i> representing explanations of content
<i>Question</i>	A subclass of <i>Annotation</i> representing questions about the content
<i>SeeAlso</i>	A subclass of <i>Annotation</i> representing a reference to another resource

**Table 1.4** Basic annotation classes

Users of the Annotea can perform the following functions with annotations:

- Creation, modification and deletion;
- Browsing;

Note also that:

- Annotations are typed in the sense that annotations could have further semantic classification such as annotation can represent an “advice”, a “comment”, a “question” etc.
- Annotations can be filtered by types, author name and annotation server.

In Annotea, the anchor mechanism to address into the documents depends on XPointer, thus the documents need to be in highly structured format, such as XML.

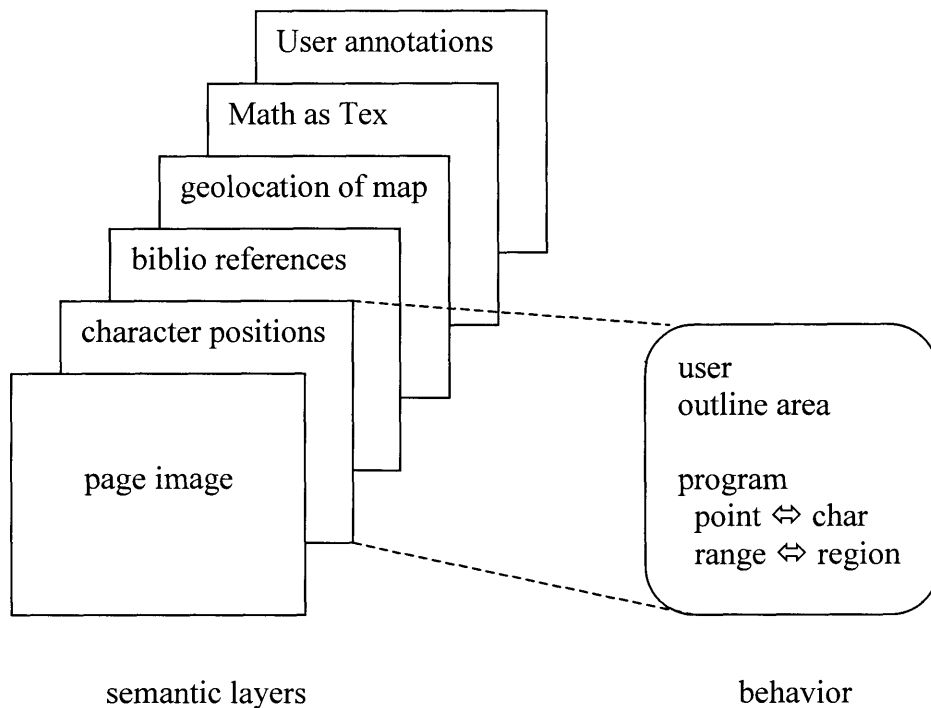
### **Multivalent Annotations**

Built on top of the Multivalent Document Model developed in UC Berkeley (Phelps and Wilensky, 1996, 1997). Multivalent Annotations is yet another representative work trying to bring annotations to digital documents.

Rather than packing all possible content types of a document in a single specification, the Multivalent Document Model slices a document into layers of homogeneous content, to which additional layers maybe added at a later stage. Each layer can be associated with multi-agents or “behaviors” which provide interactions with other layers and users.

The following figure shows the semantic layers added to a scanned image with user annotations as the last layer.





**Figure 1.10** Semantic layers of a scanned image

In the Multivalent Model, the content and functional behavior of each layer can be located on different servers. In response to a request to view a given conceptual document, the client queries relevant servers, loading content and functional behavior essential to the requested operations. Multivalent Annotations can be categorized as a browser-based architecture.

To provide a robust way of reattaching annotations in the event of document changes, multivalent annotation anchor representations are composed of three types of “location descriptors”: a unique identifier; a tree walk; and context strings.

We will investigate in more detail the robustness of the re-anchoring mechanism of its approach in the next section.

## **1.3 Annotation Persistence Mechanism**

As we stated earlier, one of the most challenging and interesting problems annotation software faces is to develop a sound annotation persistence scheme over document versions. Citing from the study of a large-scale usage of annotation systems (Cardiz et. al., 2000), annotation orphaning or bad annotation re-anchoring can become the prime cause that leads people to give up using the entire annotation system.

In this section, we first look at the different types of document modifications between versions. We then answer the question, “What are the criteria for a robust annotation persistence mechanism?” Lastly, we review and evaluate the existing literature on the various annotation persistence methodologies.

### **1.3.1 Document Modifications**

Documents may be modified in a variety ways between versions. We are especially interested in the changes the author made independently of (or unaware of) annotations made by a third party. This is quite common when readers annotate the first edition of a digital document while the author makes changes to it and publishes a newer version thereafter.

In our study of annotation persistence mechanism, we do not pay attention to the annotation content, i.e. what we put into our annotations; they could be a highlight, a paragraph of comment, or even a video clip. What we are interested in is the underlying text our annotation rests upon, i.e. the anchor text and surrounding context as defined in Figure 1.2 and Figure 1.3.

What kind of changes could anchor text and surrounding context go through in the face of document modifications? They could be rewording, moving text, or more drastically, deleting the anchor text and surrounding context.

Microsoft Research (Bernheim et. al., 2001) developed a modification classification scheme based on the annotation’s anchor text. Table 1.5 below presents different types of modifications that an annotation’s anchor text may undergo in the document modification process.

Modification Type	Modification	Description
Delete	Minor Delete	Between 1 character and half of the anchor is deleted.
	Medium Delete	More than half of the anchor is deleted.
	Total Delete	Entire anchor is deleted.
Reword	Minor Reword	Between 1 character and half the anchor is reworded.
	Medium Reword	More than half the anchor is reworded, reorganized, or split into multiple pieces.
	Total Reword	Complete anchor is reorganized. Typically only a few key words remain.
Move Anchor Text	Anchor Text Indirect	Anchor text itself doesn't change, but the text around it does.
	Anchor Text Direct	Anchor text moves within the paragraph or changes paragraphs.
Move Paragraph	Paragraph Indirect	The paragraph in front of the annotation's paragraph changes.
	Paragraph Direct	The paragraph containing the annotation moves forward or backward.

**Table 1.5** Annotation anchor modification types (Bernhein et. al. 2001)

In our study, we also pay significant attention to the annotation anchor's surrounding context. As we believe surrounding context provides additional assistance in positioning annotation anchors for the following reasons:

1. When users make annotations, they are putting their mindset in a particular semantic context. Annotations are more meaningful and compelling in this semantic space. The semantic context is reflected in both the annotation's anchor text and surrounding text.
2. Anchor texts may be repetitive in the document, as most often people annotate important words, which are distributed through the entire document. The surrounding context differentiates the original anchor from the other occurrences of those words.
3. The exact document text related to an annotation is often ambiguous. Marshall (1998) suggests that people frequently place their annotations carelessly.

We also could argue that the annotation's surrounding context could have the same modification classification scheme as anchor text has, it could also go through modifications like rewording, moving and deleting or combination of all three.

The changes for both anchor text and surrounding text can be more complex, as annotation anchor text can go through one set of changes while the annotation's surrounding context goes through another. For example, the annotation text might be reworded while the surrounding context could be partially deleted.

### **1.3.2 Robustness Criteria of *Intra-Document Locations***

In a more general picture, if we allow annotations (made by third parties) to be in arbitrary locations in a digital document, is there a way we can robustly refer these document locations in the future in the event of independent document modifications?

Or more specifically, suppose we made annotations to a particular location in a document. After the document goes through a series of changes, from minor (e.g. some rewording) to more drastic changes (e.g. deleting), could we meaningfully recover that location within the document with some confidence?

In a study by Thomas and Wilensky (2000) of UC Berkeley, the term "intra-document locations" is used to refer to an object within a document in need of positioning. They define "locations" as "indices into document content". By robust, they mean that "one should be able to make an intra-document reference to a location within an arbitrary resource, save this description, and then re-establish the location in the future, after a document has undergone some class of mutations".

In their study, Phelps and Wilensky further claim that "... to achieve robustness, two elements are needed: *a location descriptor*, which describes a location, and *a reattachment algorithm*, which attempts to reposition the descriptor within a possibly mutated target resource". We adopt their terminology in this thesis.

Phelps and Wilensky suggested in their study that intra-document location descriptors and their associated reattachment algorithms should provide the following robust mechanism:

1. "*Robust to common changes in the referenced document*"

2. *“Gracefully degrading in the face of increasing change to the document”* - Reattachment should proceed only if the computed location is likely to be the same location semantically as before the mutation. Hence minor changes should not pose much threat to reattachment; larger changes should cause reattachment to fail proportionally to the degree of change. If the change is too great, reattachment should fail. The reattachment algorithms should measure the likely quality of matches, and have a means to report failure to the user.
3. *“Based on document content”* - Location descriptors should be based on document content only, independent of presentation characteristics. Basing location descriptors on presentations (e.g. geometric location of the interface or browser) proves to be unreliable as geometric location will change in light of document mutations (e.g. a Adobe’s PDF file) or geometric location will change for a “flowed” document such as HTML when users resize the browser interface.
4. *“Work with uncooperative servers”* - A cooperative server is a server that is aware of the third party references to their documents. It will track, store all document changes related to third party references (or annotations) and may even notify the interested parties when related documents get changed. Given that the majority of the web servers are behaving as uncooperative servers, robust strategies should assume no server cooperation and should be able to calculate all useful location descriptions for possible future location reattachments without the cooperation of the source document server.
5. *“Extensible, to multimedia and various document types”* - Phelps and Wilensky argue in their study that “documents may contain multimedia elements, such as images and video, and new document types are developed regularly. A robust location mechanism should extend naturally to accommodate these and new varied types in the future, without breaking existing locations.”
6. *“Relatively small”* - In order for the reattachment mechanism to be practical, location descriptors should be relatively small. Saving the complete trail of editing of documents should be ruled out.
7. *“Straightforward to implement”* - The simplicity and ease of implementation should be vital for the wide support of the robust location mechanism.

In addition to the criteria laid out by Phelps and Wilensky, we propose another important criterion:

8. *Document Format Neutrality* - As more and more documents are published in various networked digital formats, a robust mechanism should allow the *location descriptors* to be document format neutral. Internal document structure differs significantly from one document format to another. Including internal document structure in the location descriptors from one type of document format could limit the method's applicability to other document formats. Since we can't make any assumption about the internal document structure, we must base our anchor location algorithm **only** on the **document's text**. This rules out the possibility of including a popular document structure like XML or HTML (or XPointers) in the scheme design. In upholding the rule of *document format neutrality*, we add additional benefits such as interoperability across different document formats. Our scheme could allow annotating a document in one format and later being able to view and edit the annotations on the same document in a different one.

### **1.3.3 Related Works on Annotation Persistence Mechanism**

As we pointed out earlier, annotation persistence mechanism is an essential part of annotation software development. All annotation software developed so far has, one way or the other, tried to address or solve this problem. Earlier, we reviewed a document modification classification scheme based on possible modifications made to anchor text and surrounding context; we then reviewed the robustness criteria given by Phelps and Wilensky. In the following, we will look at all the annotation software systems developed so far, review in detail their location descriptors and reattachment algorithms, and evaluate their robustness in term of the document modifications and robust criteria.

#### **Annotating “Frozen” Documents**

Adobe Acrobat Reader and Microsoft eBook Reader are among the systems which assume that annotated digital documents will never change. In these systems, annotations are typically positioned by very simple means, such as geometric locations (page number

+ a place coordination), or character offsets. Documents will never be modified, so annotations will never need to be repositioned.

Many other systems do not explicitly require documents to be frozen, but work best when there are no document modifications. In these systems, annotations are typically positioned by calculating a digital signature from the content of the page to which the annotation belongs. When the document indeed does change, these systems either silently dispose of the annotations or put them in a separate window as orphaned annotations. E-Quill, Third Voice, and Microsoft Office Web Discussions are commercial systems that have taken this approach.

### **Annotating Predefined Locations**

To compensate for potential changes in the web pages, some systems allow users to annotate only predefined locations. CoNote (Davis and Huttenlocher, 1995) inserts special HTML-like markup tags; annotations can only be made on the locations where those tags exist. By limiting the locations where annotations can be placed, the system has better control in the face of document modifications.

### **More Complex Annotation Descriptors and Reattachment Algorithms**

Many systems use more complex algorithms to re-position annotations. They all store a combination of annotated text, surrounding text, and internal document structure information so that the annotation may be repositioned later. Annotator (Ovsianniko et. al., 1999), ComMentor (Roscheisen et. al., 1995), WebVise (Gronbak et. al., 1999), Robust Locations (Thomas and Wilensky, 2000), and Annotations using Keywords (Bernheim and Barger, 2001; Bernheim et. al., 2001) are systems that take this approach. ComMentor stores key words that attempt to uniquely identify annotated text. WebVise stores a “locSpec” for each annotation that includes a bookmark or HTML target name, annotated text, surrounding text and a character count of the start position. In the following, we review in more detail a few systems that are robust to varying degrees. Each system tries to solve the problem using different re-positioning algorithms.

- **Annotea**

Earlier, we reviewed Annotea's architecture and system design. Annotea is a W3C collaborative web annotation project based on general-purpose open metadata infrastructure where the annotations are modeled as a class of metadata. It uses mostly W3C open source technologies, such as RDF, XLink, XPointer and HTTP.

In Annotea, annotated documents need to be well-formed structured documents, such as XML. XPointer is used to address into the annotated context within XML/HTML documents. XPointer proves to be an unreliable intra-document location mechanism because it can handle only limited document modifications, and it can only be used for well-structured documents. The annotation mechanism used in Annotea violates our robust criteria rule number eight. Using XPointer as the intra-document location mechanism proves to fail for many common document changes.

- **Annotator**

In Annotator, a location descriptor records a portion of text in the anchor's proximity. The string to be remembered must be long enough to be unique in the whole document. Its length is determined by a heuristic algorithm, which verifies the uniqueness conditions. In reality, the search sub-string can be as short as several characters, or as long as almost the whole document.

Annotator argues that it is against the copyright law to store copies of data from copyrighted material in a publicly accessible database without explicit permission of the publisher. Because of the copyright problem and uncertainty of the length of the location descriptor strings, Annotator hashes the sub-string. In the implementation, the string need not be positioned at the very beginning of the anchor, but can be off a few characters from these points. Annotator also records the few initial characters of these strings as a hint to improve the speed of search for the anchor points within a document.

Annotator essentially adopts a text matching algorithm to re-attach annotations. The literature has no explanation on how the string is hashed and how the reattachment algorithm works when documents undergo changes, even slight ones. But simple string matching cannot survive even moderate modifications of the anchor text.

- **Multivalent Annotation**



Building on the Multivalent Document Model, Phelps and Wilensky (2000) developed a strategy called “robust locations” to address the annotation re-anchoring problem and to try to meet the robust criteria defined earlier in this review.

The core of their strategy is to,

1. “provide automatically generated location descriptors, which comprise multiple descriptions of a location, each of which captures different aspects of the documents.”
2. “use heuristics when a descriptor does not resolve directly to a location to hypothesize the intended location, along with a measure of the degree of confidence in the hypothesized location”

In Multivalent Annotation, anchor information includes three location descriptors

- o A unique identifier (UID)
- o A tree walk
- o Context.

A unique identifier (UID) is a name unique within the document, as per ID attributes in SGML/XML. UIDs are put into the document by the document owner. “They normally survive even the most violent document modifications, except their own deletion”, Phelps and Wilensky argue.

A tree walk (similar to the concept of XPath in XML) describes the path from the root of the document, through internal structural nodes, to a point at a leaf. Phelps and Wilensky use “tree walk” as the central component of the robust location strategy. They argue that “tree walks incrementally refine the structural position in the document as the walk proceeds from root to leaf, they are robust to deletions of content that defeat unique ID and context locations”

Context is a small amount of previous and following information from the document tree. Multivalent Annotation proposes a context record containing a sequence of document content prior to the location, and a sequence of document content following the location. Context records could be in arbitrary length. In their implementation, 25 characters are used as context text.

In Multivalent Annotations, the robust location reattachment algorithm is performed in three steps. It first uses the unique identifier, then the tree walk descriptor, and then the

context descriptor, continuing only if the previously tried methods are ineffective. Each of these steps is described below.

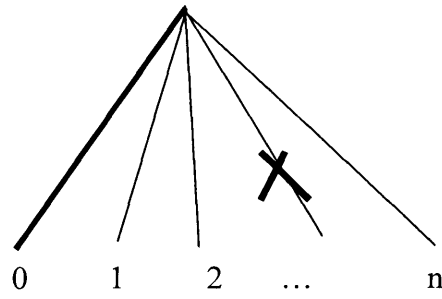
1. Unique ID sub-method

If the location has a UID, and the same UID is found in the revised document, presumably the location is resolved. If no identical UID is found, proceed to the tree walk sub-method.

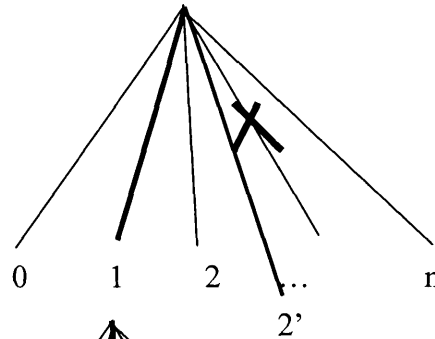
2. Tree walk sub-method

Tree walk is used as the main reattachment method in Multivalent Annotation because of the likely sparcity of UIDs. As show in the following figure (Figure 1.11), a tree walk is immune to the changes to the following sibling subtrees. The walk is also safe with non-structural changes to previous sibling trees.

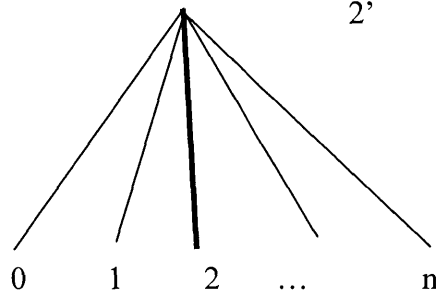
**Book**



**Chapters**



**Sections**



**Figure 1.11** Tree walk sub-method

Tree walk, however, fails when a new previous sibling is introduced or one is deleted. In Multivalent Annotation, other than the child node numbers, the node names are also recorded. They will be used when tree walk fails after the initial walk.

Suppose the matching a tree walk of a robust location against the actual running tree successes up to the leaf, but the leaf is of the wrong type. The matching is tried on the following sibling node, then on the previous sibling, and then on the second following, the second previous and so on until the no names match. If the match fails previously, the same strategy is employed. If no match can be found, Multivalent Annotation assumes that a new level of hierarchy was introduced, and skips a level of the tree. If still no match was found, they hypothesize that a level of hierarchy was removed, and skip the

current location node/offset descriptor pair. If none of these attempts yields a match, the match attempt fails.

Each departure from the original description adds a weighted value to an overall “unreliability” factor for the match.

This algorithm makes saved tree paths robust against any number and combination of sibling insertions, deletions and reordering at any or numerous levels of the document tree. The search pattern prefers siblings closest to the saved location, rather than, say, searching from the parent’s first child to its last, reasoning that closet match is most likely the best match.

### 3. Context sub-method

If the structural document tree has been changed too much for the tree walk’s tactics to recover an annotation’s location, the context method is used.

As with tree walk, the closest match to the original position is the preferred one. A search is done forward and backward, with the nearer match chosen. If neither direction matches, more and more of the context is searched until a match is found, or until the length of the string used for the search drops below a threshold.

The initial search position is set from the furthest extent that the tree walk described above could be resolved. If no match is found within the subtree, the search climbs up the tree by one node and tries again within that subtree until a match is found or it has climbed to the root.

As with tree walk matching, an overall unreliability factor for the match is computed. If some but not all of the context is matched, unreliability is increased.

The algorithm used in Multivalent Annotation depends highly on the structure of the document format. It failed to pass our robustness criteria rule number eight.

The tree walk method works in some circumstances, but we suspect the tree walk method will not be able to survive the complications in the ways the document could be modified. The claim of the method satisfying rule number one still needs to be questioned.

- **Robust Anchoring Annotations Using Keywords**

Microsoft researchers (Bernheim and Barger, 2001; Bernheim et. al., 2001) approach the annotation reattachment problem from the user's point of view. In a study performed by Microsoft Research, a group of users were recruited to evaluate a simple text matching annotation-positioning algorithm. The algorithm saved only the anchor text selected by the user and then used partial string matching to find a position in the modified document. No information about the surrounding context of the annotations was saved.

The study result is interesting; it suggests that participants paid *little attention* to the surrounding context of an annotation, indicating that users might not consider the surrounding context very important for annotations made on a range of text. Results suggest that algorithms may want to give the surrounding context relatively little weight when determining an annotation's position.

The evaluation further revealed that unique words in the vicinity of an annotation are distinguishing anchor characteristics, which should be tracked among successive versions of a document.

Based on the user's evaluation, Microsoft introduced keyword anchoring, a robust anchoring method designed based on what users expect to happen to annotations when the documents change. The algorithm primarily uses unique words from the annotated document to anchor and re-position annotations, and it ignores any specific internal document structure.

Location descriptors includes:

- o *HMTL book mark for the selection*: an IE specific string used to quickly anchor annotation in documents that have not changed (can be ignored to ensure the anchor is completely independent of the document format)
- o *Offset from start of document*
- o *Length of the anchor text*
- o *Information start and end points of the anchor text*: a small of amount of text from the document surrounding the start and end of the anchor text
- o *Information about the keywords in the anchor text*: a list of unique words from the anchor text and their locations within the anchor text

### **Reattachment Algorithm:**

The reattachment algorithm first assumes the document has not changed. The bookmark and document offset information are used to find the anchor. When the initial attempt fails, the algorithm looks for keywords from the anchor in the modified document.

The algorithm loops through all keywords in the original anchor and creates each “candidate anchor” whenever there is a keyword match (called “seed” keyword in this case). Each “candidate anchor” is assigned a base confidence score. The confidence scores of the “candidate anchors” will be raised by performing the following steps:

1. Looking for keywords in the vicinity

If there are other keywords in the vicinity of the original anchor (within two times of the initial anchor length), the algorithm extends the candidate anchor to include it. The algorithm also measures the distance between the newly found keyword and the seed keyword. The confidence score is modified based on the relative change in distance between the seed keyword and newly added keyword. The higher the confidence score is raised, the more similar the distribution of the keywords between the two document versions.

2. Looking for the start and end points of the anchor text

After including as many keywords as possible in each candidate anchor, the algorithm tries to match start and end points of the original anchor to each candidate anchor. If there is a match, the confidence score is raised. Again the increase of the confidence score is based on how closely the new distance from the seed keyword to the start/end point matches the distance in the original anchor.

3. Comparing the length and offset of the found candidate anchor to the original anchor.

The method further compares each candidate anchor (after steps 1 and 2), to the length of the original anchor and its location in the document. Confidence scores are then increased or decreased depending on if the found anchor’s length and location compares favorably with those of the original one.

4. Looking for surrounding text from end points

Finally, for the cases when there are many multiple matches of the anchor text in the modified documents, the surrounding text is used to differentiate among them.

Confidence scores are then modified based on this comparison.

After going through all the steps, the candidate anchor with the highest confidence score is used to reposition the annotation.

In summary, the algorithm seeks the best match in terms of the number of matching keywords, their relative distribution in the anchor text, the anchor text's initial and ending text match and the anchor's relative position in the whole document.

### **Keywords Selection:**

The keywords are determined by selecting the words in the anchor text that are most unique with respect to the rest of the document. For a particular document, a map of word frequencies is calculated. When the user creates an annotation, all words in the anchor text that only occur once in the document are selected. If there are fewer than three words in the anchor text that occur only once in the document, words with increasing frequency are selected until at least three keywords have been found.

The method developed in Microsoft Research upheld the rules that document format should not matter and that reattachment algorithm depends only on document text. The method used in selecting keywords sounds simple, though we foresee problems will arise when the document grows large. The semantic significance of keywords depends on the distribution of the keywords in the documents rather than the frequency alone. The algorithm essentially seeks the best match in terms of the number of matching keywords, their relative distribution in the anchor text, the anchor text's initial and ending text match and the anchor's relative position in the whole document. In many cases, the method would fail to properly position the annotation,

1. If the anchor texts are reworded in a way that it contains quite similar semantic contents, but uses different keywords.
2. If the anchor texts are reworded in a way that word sequences are changed although they express exactly the same meaning.

We also believe that reattachment algorithms depending only on keyword match is fundamentally flawed, as many times, users make annotations because they are

particularly interested in a specific concept, which might not include any significant keyword at all.

## **1.4 Annotation Persistence over Dynamic Documents – a Specialized Information Retrieval Problem**

### **1.4.1 A Specialized Information Retrieval Problem**

The problem we are facing in this study is a very interesting and unique one. It is an Information Retrieval (IR) problem. We call it *a specialized IR problem* because of its well-defined user queries and information collection once the documents of interest are pre-defined.

In our study, what we are interested in is to transfer annotations “robustly” between document versions. We pay no attention to what is written into the annotation itself, but rather consider solely the document text underlying the annotation (“anchor text and surrounding context”). In our problem, the user query is well defined. The user query is the “anchor text + surrounding context” in the original version of the document, and the information collection is the revised version of the document. Our IR system should attempt to find the most “relevant” piece of text in the revised version of the document in response to the user query (“anchor text + surrounding context”) in the original version of the document. If we assume users can make annotations anywhere in a document, the study boils down to ***using any piece of text in the original edition of a document to find the most “relevant” piece of text in the revised edition of the document.***

Information Retrieval (IR), as a subject of science and engineering, has been studied for many years. In the past 20 years, the area of IR has grown well beyond its primary goals of indexing text and searching for useful documents in a collection. Nowadays, IR is broadly interpreted to include such technologies as ad hoc and distributed retrieval, cross-language IR, summarization, filtering, and classification of information. With the introduction of WWW, IR once again gained a place with other technologies at the center of the stage of computer science. Nowadays, IR has become part of our daily life. Frequently, when we go online, we select our favorite search engine, type in a sentence and find the information need.



IR has proven to be a hard subject. The difficulties are attributable to mainly three reasons.

1. Because of the ambiguities inherent in natural language, the information is often misinterpreted.
2. Characterization of the user information needs is not a simple problem. Often, the information need is imprecisely or vaguely stated by the user.
3. Unlike most of the algorithms in computer science that have a “right answer”, IR techniques are essentially heuristics because there is no right answer, or we don’t know the right answer. As *relevance* is at the center of the information retrieval, to be effective in its attempt to satisfy the user information need, a IR system must somehow ‘interpret’ the contents of the documents and rank them according to a degree of relevance to the user’s query.

Since we can define our problem as a specialized IR problem, in the following, we first review the three retrieval evaluation criteria widely used in IR research – *Precision, Recall and Ranking*. We then review the three classic categories of IR models, *Boolean Model, Vector Model and Probabilistic Model*, and their extensions. Later, we revisit our original problem. We look at the differences between “conceptual search” and “mechanical word search” when performed by humans and computing machines, and their relationships to the reattachment problems under common human annotation practices. We also review Latent Semantic Analysis, an extension to vector model, which is an automated method to compare documents and queries based on their semantic similarities. Lastly, we formally define our problem and lay out strategies to solve it.

#### **1.4.2 IR Evaluations and IR Models**

##### **IR Evaluation Criteria – Precision, Recall and Ranking**

Before we turn our attention to review the IR models, we first look at the three measures to evaluate retrieval performances - *precision, recall and ranking*.

In talking about retrieval models and how to improve them, it helps to remember what distinguishes an effective retrieval from a fruitless one. To be truly effective, there are generally three things we want from a retrieval system.

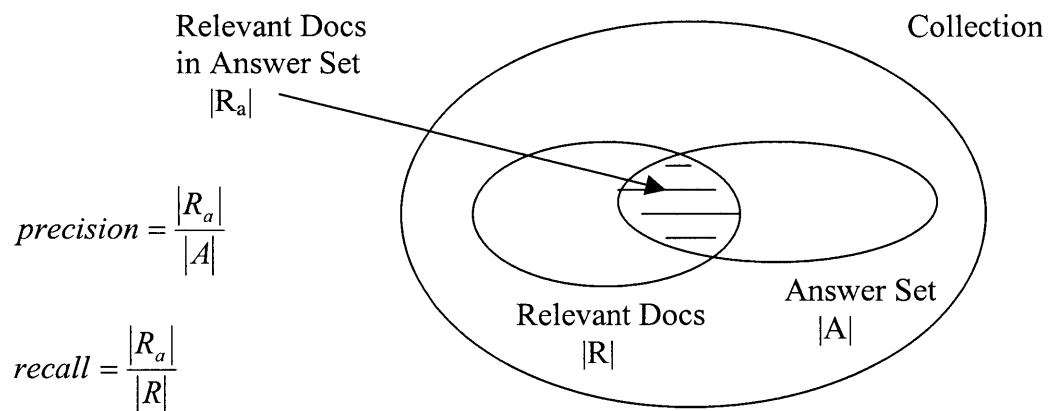
1. We want it to give us *all* of the relevant information available on our topic.
2. We want it to give us *only* information that is relevant to our search.
3. We want the information ordered so that most of the relevant results come first.

The first criterion, getting all the relevant information available – is called *recall*.

Without good recall, we have no guarantee that valid, interesting results won't be left out of our result set. We want the rate of false negatives – relevant results that we never see – to be as low as possible.

The second criterion – the proportion of documents in our result set that is relevant to our search – is called *precision*. With too little precision, our useful results get diluted by irrelevancies, and we are left with the task of sifting through a large set of documents to find what we want. High precision means the lowest possible rate of false positives.

The following figure shows the concept of recall and precision given an information collection and an information request.



**Figure 1.12** Precision and recall for a given example of information request

There is an inevitable tradeoff between precision and recall. Search results generally lie on the continuum of relevancy, so there is no distinct place where relevant results stop and extraneous ones begin. The wider we cast our net, the less precise our result set becomes. This is why the third criterion, *ranking*, is so important. Ranking has to do with whether the result set is ordered in a way that matches our intuitive understanding of what is more and what is less relevant.

## IR Models

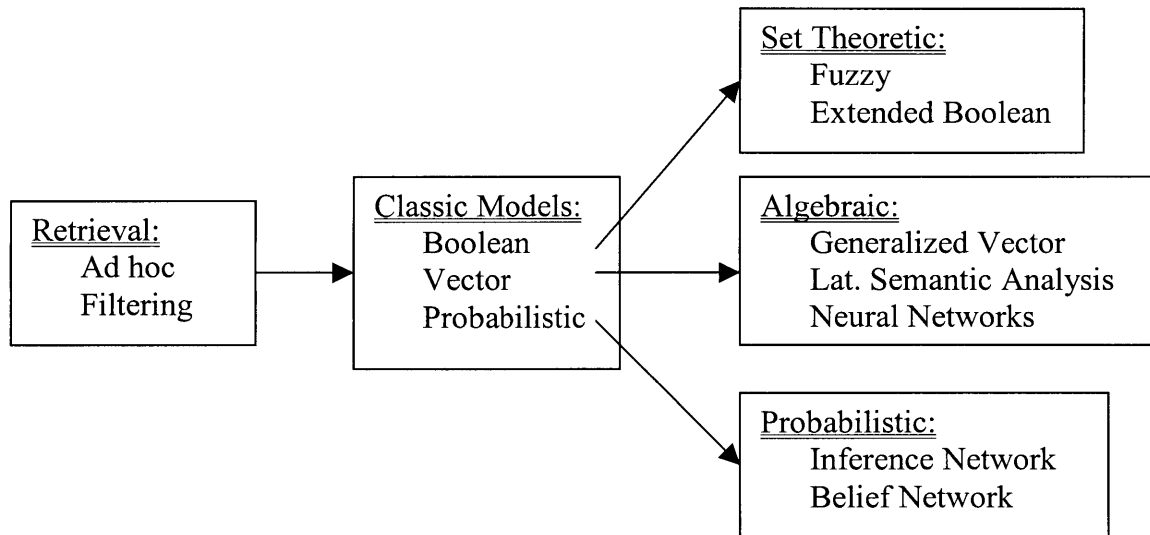
Interest in information retrieval has existed long before the Internet. There have been many models proposed over the years. In their classic IR book, “Modern Information Retrieval” (Baeza-Yates and Ribeiro-Neto, 1999), Ricardo Baeza-Yates and Berthier Ribeiro-Neto defined the following formal characterization of IR models

**Definition** *An information retrieval model is a quadruple  $[D, Q, F, R(q_i, d_j)]$*

where

- 1)  *$D$  is a set composed of logical views (or representations) for the documents in a collection*
- 2)  *$Q$  is a set composed of logical views (or representations) for the user information needs. Such representations are called queries.*
- 3)  *$F$  is a framework for modeling document representations, queries, and their relationships.*
- 4)  *$R(q_i, d_j)$  is a ranking function which associates a real number with a query  $q_i \in Q$  and a document representation  $d_j \in D$ . Such ranking defines an ordering among the documents with regard to the query  $q_i$ .*

All IR models can be classified into three classic categories, *Boolean, Vector and Probabilistic*. In the Boolean Model, documents and queries are represented as sets of index terms. In the Vector Model, documents and queries are represented as vectors in a multi-dimensional space. In Probabilistic Model, probability of relevance is calculated based on the assumption that the terms are distributed differently in relevant and non relevant documents, Over the years, extensions and refinements have been suggested to each classic model. Figure 1.13 is taxonomy of information retrieval models from the book of “Model Information Retrieval” (Baeza-Yates and Ribeiro-Neto, 1999).



**Figure 1.13** Taxonomy of information retrieval models  
(Baeza-Yates & Ribeiro-Neto, 1999)

### *Boolean Model*

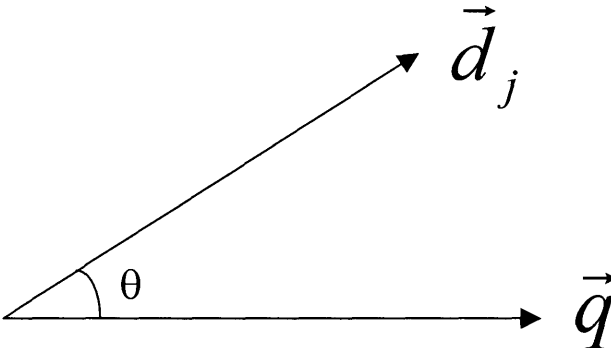
The Boolean model is the most simple of these retrieval models. It is based on set theory and Boolean algebra. The terms in a query are linked together with AND, OR and NOT. This method is often used in search engines on the Internet because it is fast and simple.

Unfortunately, the Boolean model suffers major drawbacks. First, users have to have some knowledge to the search topic for the search to be efficient; a wrong word in a query could rank a relevant document non relevant. Secondly, frequently it is difficult to use simple Boolean expressions to express information needs. Lastly, since Boolean model is based on a binary decision criterion, the retrieved documents are all equally ranked with respect to relevance.

Alternative models have been proposed to extend and refine the classic Boolean Model to solve these problems. Expanded term weighting operations make ranking of documents possible, where the terms in the document are weighted according to their frequency in the document (Salton 1983). Fuzzy operators are used in place of Boolean operators (Lee et. al., 1993). Weighted query can be expanded to include synonyms using a thesaurus (Kwon et. al., 1994).

### Vector Model

Recognizing the use of binary weights in Boolean Model leads to poor relevance evaluation, the Vector Model assigns non-binary weights to index terms in queries and documents. As each document and user query is represented by weighted index terms, mathematically, they are represented as t-dimensional vectors. The degree of similarities of between the document vector ( $\vec{d}_j$ ) and user query vector ( $\vec{q}$ ) can be computed as the correlation between the two vectors. This correlation can be quantified, for instance, by the cosine of the angle between these two vectors (Figure 1.14).

$$sim(d_j, q) = \frac{\vec{d}_j * \vec{q}}{|\vec{d}_j| \times |\vec{q}|}$$


The diagram illustrates two vectors,  $\vec{d}_j$  and  $\vec{q}$ , originating from a common point. Vector  $\vec{q}$  is horizontal and points to the right. Vector  $\vec{d}_j$  points upwards and to the right. The angle between the two vectors is labeled  $\theta$ .

**Figure 1.14** Cosine of two vectors

The vector model procedure can be divided into three stages. The first stage is the document indexing, where content bearing terms are extracted from the document text. The second stage is the weighting of the indexed terms to enhance retrieval of documents relevant to the user. The last stage ranks the document with respect to the query according to a similarity measure.

- Document Indexing

Many words don't carry semantic meanings, such as *the* and *is*. In the process of document indexing, those non-significant words (function words) are removed from the document vector, so the document will only be represented by content bearing words. The indexing can be based on term frequency, where terms that have both high and low frequency within a document don't carry much document association power. In practice, term frequency has been difficult to implement in automatic indexing. Instead, a stop list that holds common words is used to remove high frequency words (stop words). In

general, 40-50% of the total number of words in a document is removed with the help of stop list.

- Term Weighting

Index term weights can be calculated in many different ways, but the main idea behind the most effective term-weighting techniques is based on the principles that support clustering techniques. Term weighting can be explained by controlling the exhaustivity and specificity of the search, where the exhaustivity is related to recall and specificity to precision. The term weighting for the vector model has entirely been based on single term statistics (or with an assumption that terms are not related). There are two main factors that contribute to term weighting: term frequency factor and collection frequency factor.

The term frequency factor measures the frequency of a term  $t_i$  inside a document  $d_j$ . The term frequency factor provides for quantification of intra-cluster similarity, which provides one measure of how well that term describes the document content.

The collection frequency factor measures the inverse of the frequency of a term  $t_i$  among the documents in the collection. The collection frequency factor provides for quantification of inter-cluster dissimilarity, which assumes that the terms that appear in many documents are not very useful for distinguishing a relevant document from a non-relevant one. Experimentally it has been shown that these document discrimination factors lead to a more effective retrieval, i.e. an improvement in precision and recall (Salton and Buckley, 1996).

A classic term weighting strategy is called  $tf - idf$  scheme. Assume  $N$  be the total number of documents in the collection and  $n_i$  be the number of documents in which the index term  $k_i$  appears. Let  $freq_{i,j}$  be the raw frequency of term  $k_i$  in the document  $d_j$  (i.e., the number of times the term  $k_i$  is mentioned in the text of the document  $d_j$ ). Then the  $tf_{i,j}$  of term  $k_i$  in document  $d_j$  is given by:

$$tf_{i,j} = \frac{freq_{i,j}}{\max_l(freq_{l,j})}$$

Where the maximum is computed over all terms that are mentioned in the text of the document  $d_j$ .

Let  $idf_i$  be the inverse document frequency for  $k_i$ . It is given by

$$idf_i = \log \frac{N}{n_i}$$

$tf - idf$  term-weighting is defined as:

$$w_{i,j} = tf_{i,j} \times idf_i = \frac{freq_{i,j}}{\max_l(freq_{l,j})} \times \log \frac{N}{n_i}$$

- Ranking scheme

The similarity in vector space models is determined by using associative coefficients based on the inner product of the document vector and query vector, where word overlap indicates similarity. The inner product is usually normalized. The most popular similarity measure is the cosine coefficient, which measures the angle between the document vector and the query vector. There are other measures, such as Jaccard and Dice coefficients (Salton 1988).

#### *Probabilistic Model*

Another classic retrieval method is probabilistic retrieval, where the probability that a specific document will be judged relevant to a specific query is based on the assumption that the terms are distributed differently in relevant and irrelevant documents. The probability formula is usually derived from Bayes' theorem.

### **1.4.3 Problem Revisited – Concepts and Keywords**

Earlier in this Chapter we reviewed the state-of-art research in annotation systems and architecture. We concluded that the function of reattaching annotations among document versions is a required element for any annotation system to be successful. We also concluded that annotation persistence over document versions is a complicated and

challenging problem, as documents can go through various changes among versions. Before we reviewed the state-of-art annotation re-anchoring methods, we defined robust criteria for annotation re-anchoring mechanism. We concluded in our review that none of the current mechanisms proposed in the literature fully satisfies our robust criteria. In the last part of this Chapter, we revisited our problem and stated that the study of annotation persistence over dynamic documents can be formulated as a specialized information retrieval problem.

In the following, we look at the problem from another point of view. We look at the differences between two common ways of searching, “Conceptual search” and “Mechanical word search”. One (“conceptual search”) is used most often by human brains and the other (“mechanical word search”) is well-suited for computer automation. After the brief discussion of the two types of searches, we review a new retrieval models, *Latent Semantic Analysis*, which allows a computer to do human work, i.e. it retrieve documents based on semantic concept matching. Next, we look at the two common practices of annotations, annotation on concept and annotation on important words, and their relationship to the search types of conceptual search and mechanical word search as we develop our annotation reattachment strategies.

### **Conceptual Search vs. Mechanical Word Search**

When a human is given a task to find all articles having to do with *Egyptian Civilization* from a stack of newspapers and magazines, it is very unlikely he would read through each article word-by-word, looking for that exact phrase. Instead, he would probably skim through each article’s headline searching for those that might have to do with ancient art or history, and then read through the ones he finds where there might be a connection to the topic of interest.

If, however, he is given a task of finding information about a term such as *singular value decomposition* from highly technical journals or text books, the chances are high (unless he is a mathematician) that he would have to go through each article line-by-line, looking for “*singular value decomposition*” to appear in the sea of jargon.

The two searches would yield very different results. In the first one, we might find many articles that are very relevant to *Egyptian Civilization*, even though the exact words



might not appear in the article at all. Articles about Metropolitan Museum of Art, New York, and the Tomb of Tutankhamen or even article on archaeology might surface.

With the math articles, we might find all the articles with the exact phrase *Singular Value Decomposition*. Unless we knew about relevant mathematics, it is unlikely we would pick articles about matrix algebra that did not contain the search phrase, even though a mathematician might find those articles very relevant.

The two searches represent opposite ways of searching a document collection. The first is a conceptual search, based on a higher-level understanding of the user's need and the search space, including all kinds of contextual knowledge. The second is a purely mechanical search, based on an exhaustive matching between the search phrase and the collection of documents. It requires no understanding of either the query or the document.

Computers are perfect for doing mechanical comparisons. Human beings can never take a purely mechanical approach to a text search problem, because human beings can't help but notice semantics, structures and implications. Computers know nothing about context, and excel at performing repetitive tasks quickly.

Current full text search engines, no matter how complex, find their results using just such a mechanical method of exhaustive searching. While the technique it uses to rank the results may be very sophisticated (Google is an example of innovation in choosing a system for ranking), the actual search is based entirely on keywords, with no higher-level understanding of the query or documents.

Could a computer retrieve documents based on semantic concept matching between query and documents? In the following we review one promising approach, called *Latent Semantic Analysis (LSA)*, an innovative extension to IR Vector Model. LSA was first developed at Bellcore in the late 1980's, and is the object of active research.

### **Latent Semantic Analysis**

Latent semantic analysis adds an important step to the vector model. In addition to recording which content bearing words a document contains, the method examines the document collection as a whole, to see which other documents contain some of those same content bearing words. LSA considers documents that have many words in

common to be semantically close, and ones with few words in common to be semantically distant.

Retrieval models suffer from two well-known language related problems called synonymy and polysemy. Synonymy means that an object can be referred to in many ways, i.e., people use different words to search for the same object. An example of this is the words *car* and *automobile*. Polysemy is the problem of words having more than one specific meaning. An example of this is the word *jaguar* that could mean a well-known car type or an animal. The prevalence of synonyms tends to decrease the recall performance of retrieval systems. Polysemy is one factor underlying poor precision.

LSA offers a dampening of synonymy. By using Singular Value Decomposition (SVD) on a term by document matrix of term frequency, the dimension of the transformed space is reduced by selection of the highest singular values, where the most of the variance of the original space is. By using SVD the major associative patterns are extracted from the document space and the small patterns are ignored. The query terms can also be transformed into this subspace and can lie close to documents where the term does not appear. The advantage of LSA is that it is fully automatic and does not use language expertise.

Intuitively, the reduced dimension ( $\sim 100 - 300$  orthogonal space) may be viewed as artificial semantic concepts; they represent extracted common components of many different words and documents. Each term or document is then characterized by a vector of weights indicating its strength of association with each of these underlying semantic concepts. Similarities between terms and documents are then measured by how closely they contain similar semantic concepts.

Empirical studies of LSA have demonstrated LSA has better recall and precision performance than traditional vector models. LSA has also been examined analytically. By comparing LSA to multidimensional scaling it has been shown that LSA preserves the document space optimally when using the inner product similarity function (Bartell et. al. 1992). By using Bayesian regression model it is shown that by removing the small singular values, statistically dubious information are being removed and also specification errors are reduced (Story, 1996).

In Chapter 3, we elaborate on LSA methodology and evaluate the method against a collection of Wall Street Journal articles from the late 1980s.

### **Annotation on Concept and Annotation on Domain Semantically Significant Words**

While humans most often conduct “conceptual searches” when they are trying to find the information they need, they often avoid “mechanical keyword searches”. When it comes to annotation, the story is different.

Humans make annotations on both important words and important concepts. Very often, when we read articles, books, especially in scientific studies, we annotate domain semantically significant words, because they represent new jargon within a certain conceptual context.

At some other times, our annotations address key concepts. We often highlight a paragraph because it introduces new ideas; we underline a list of directions because it represents procedures to finish certain tasks. In many cases, whole paragraphs might not contain any semantically significant words at all.

It is generally difficult to automatically identify domain semantically significant words. We need to point out that semantically significant words are usually not the same as rare occurring words. In object oriented programming context, the word “polymorphism” is usually both rare occurring word and semantically significant word; while “class” is semantically significant word, but not rare occurring word.

In this study, we don’t make special efforts to identify semantically significant words. We will, however, take the advantages of our ability to extract document semantic concepts and our ability to automatically identify rare occurring words to help reposition annotations.

#### **1.4.4 Strategies and Solutions**

As we reach the concluding part of the first chapter, we are now ready to layout our strategies to solve the “annotation persistence over dynamic documents” problem. In the following, we present our design of annotation anchor formulation and specify the guidelines of reattachment algorithm.

## Anchor Formulation

In section 1.1.4, we presented a generic annotation definition. By adopting the object oriented programming convention, an object of “annotation” contains an object of “content” and an object of “anchor”. “Anchor” contains location referencing information to address into a document and position the annotation properly.

To uphold the rule of document format neutrality, our method needs to depend on document text only. This rules out the possibility of using internal document structures, such as XML. However this does not mean we can’t use the semantic meta-structure of the documents, such as the anchor text is located within “*Chapter 1 Introduction/1.4 Annotation Persistence over Dynamic Documents – A Specialized Information Retrieval Problem/1.4.4 Strategies and Solutions*“. The meta-structure information could be readily parsed and generated automatically (or with very light human intervention). We include meta-structure information of “anchor text and surrounding context” as one location descriptor in our anchor formulation.

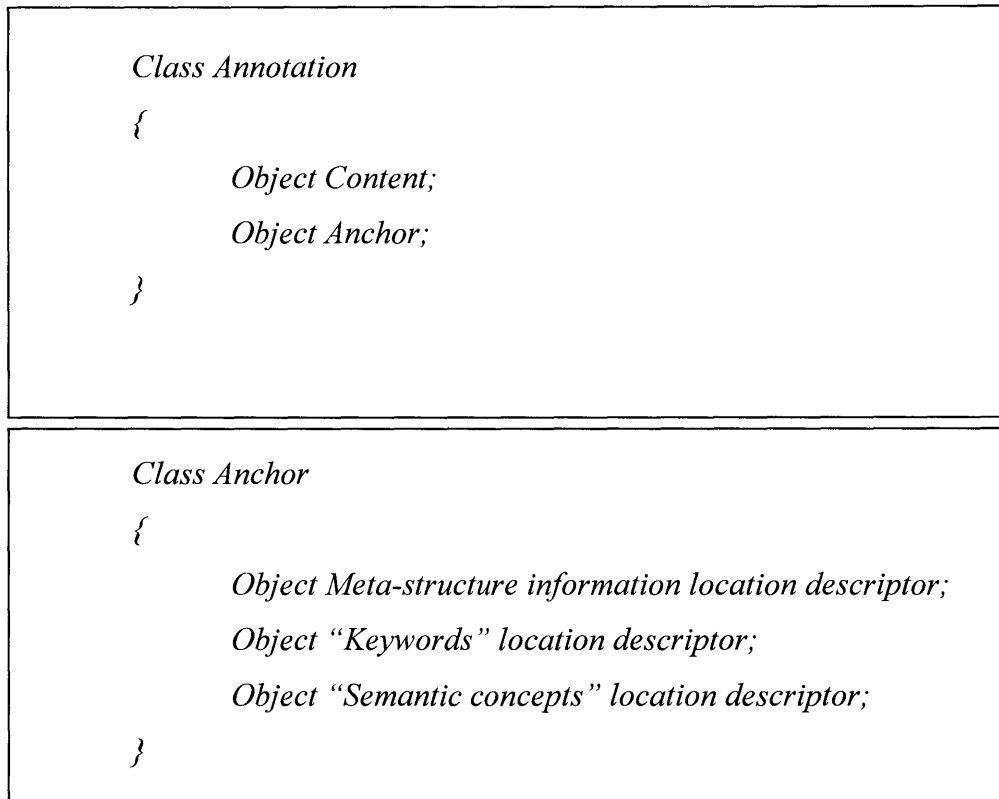
To address the need of maintaining concepts which anchor text and surrounding context contains, the anchor formulation contains a location descriptor which describes the semantics of the anchor text and surrounding context (thus “concepts”).

We also like to add another location descriptor which contains a lists of rare occurring words (we call them “keywords” in the future references). The problem we are studying is a specialized information retrieval problem in that we are looking for similarities between document versions. That fact that rare occurring words appearing in both versions of the document indicate that there is a strong likelihood that the paragraphs where these words appear are related.

What words are *keywords*? Or put another way, what words in a document are statistically more significant so that such words appearing in one version of a document could be used as reliable tokens to uniquely identify the “same” texts in another version of the document? We call the ability of words to resolve document locations as words’ indexing power. Keywords are rare occurring words.

Figure 1.15 presents the full definition of *Annotation Class*; each object of “annotation” is composed by an object of “content” and an object of “anchor”. Each

“anchor” has three location descriptor object, -- meta-structure location descriptor; keywords location descriptor; and concept location descriptor.



**Figure 1.15** Annotation Class Representation

### **Reattachment Algorithm**

The anchor formulation and reattachment algorithm in this study are designed with a goal of satisfying the robust criteria stated in section 1.3.2.

In the case when there are only minor modifications made to annotation’s anchor text and surrounding context, we expect the reattachment algorithm to pick the right location in the revised document with high confidence. In the case when there are increasing changes to the documents, the reattachment algorithm should present a list of possible answers with a ranking with confidence scores. In the case when the document changes are radical, the reattachment algorithm should orphan the annotations rather than reattach

them with low confidence. We will elaborate our reattachment algorithms in detail in Chapter 4.

## 1.5 Summary

In Chapter One, we started by reviewing the practice of making annotations on paper, the taxonomy of annotation forms and their functions. By adopting Marshall's classification, annotations fall into four groups based on whether the annotation locations are within-text (e.g. highlighting text, circled words) or they are in the margins (e.g. scribbled notes in a margin, asterisks and stars) and whether the annotation contents are explicit in meaning (e.g. brief notes) or opaque personal coding (e.g. red underlining indicating importance).

We identified the benefits digital format brings to annotations. The list includes such benefits as: annotations can be searched; annotations can themselves be annotated; annotations could reference multiple locations; annotation contents could be multimedia; annotation could be shared.

We also identified many challenges digital annotations face. We elaborated one of the challenges, which is the focus of this study, "*Annotation persistence over dynamic documents*". We made an observation that *Annotation persistence over dynamic documents* is a common problem in the digital annotation world, as documents, such as digital books and the WWW, are most often "incorporative". By citing from a large-scale annotation software study conducted at Microsoft, a failed annotation persistence scheme over dynamic documents in annotation software can be costly. It was the primary reason people stopped using the annotation software system.

We presented a generic definition of digital annotation based on Marshall's classification. A digital annotation object is composed of the two properties, of annotation content and annotation anchor, where annotation anchor contains information to address into a document to position the annotation properly.

In the second section of Chapter One, we first presented an overview of the current state-of-art annotation systems and architecture. We then elaborated on the design of a few systems which represent the forefront of research in annotation system development.

In commercial document-processing software, annotations are normally stored within document files, and users need to have write permissions to add annotations. There are a few systems which limit the annotations to only predefined document positions. Most

research annotation systems, however, allow users to annotate any arbitrary web documents.

The two main categories of annotation system architecture are *proxy-based* and *browser-based*. In a *proxy-based* approach, annotations are stored and merged with a web document by a proxy server; the browser user only sees the result of the merge. Typically, presentation of the annotations is limited to the presentation styles available through HTML. The proxy approach inherently restricts the presentation styles that can be used for annotations. In a *browser-based* approach, the browser is enhanced (either by an external application or plug-in) to merge the document and the annotation data just prior to presenting the content to the user. Annotation data are stored in a proxy or a separate annotation server.

We then reviewed in detail three systems, Annotator, Annotea and Multivalent Annotations. They are all developed as research projects. In Annotator, system adopts a *proxy-based* architecture. It allows annotation functions such as annotation creation, modification and deletion; annotation indexing and searching; annotation multi-referencing; annotation sharing. To keep the principle of document format independence, the anchor representation in Annotator does not rely on document markup or structures, it contains only a portion of the text in the anchor's proximity. The text strings are subsequently hashed and saved as part of annotation anchor record. A string or sub-string match of anchor text with the document is performed to position the annotation anchors.

Annotea adopts a browser-based architecture. It uses mostly W3C open source technologies, such as RDF, XLink, XPointer and HTTP. The client browser uses the Amaya editor/browser, an open source program developed by W3C that supports HTML and a variety of XML markup schemas. In Annotea, Resource Description Framework (RDF) is selected as the metadata language to represent annotations. A general annotation super class in RDF schema is defined and several sample subclasses are defined as well to showcase the flexibility of extending annotations to include more annotation sub-types. In Annotea, the anchor mechanism to address into the documents depends on XPointer, thus the documents needs to be in highly structured format, such as XML.

Multivalent Model can be categorized as another *browser-based* architecture. Rather than packing all possible content types of a document in a single specification,



Multivalent Document Model slices a document into layers of homogeneous content, to which additional layers maybe added at a later stage. Each layer could be associated with multi-agents or “behaviors” which provides interactions with other layers and users. User annotations are treated as one layer. To provide a robust way of reattaching annotations in the event of document changes, multivalent annotation anchor representations are composed of three types of “location descriptors”, a unique identifier; a tree walk; and context strings.

In the third part of Chapter One, we first looked at the different types of text modifications document could undergo between versions. We then answered the question “ what are the criteria for a robust annotation persistence mechanism?” Lastly, we reviewed and evaluated the existing literature of the various annotation persistence methodologies.

Document text may be modified in a variety ways between versions. Text modifications can include rewording, moving, or more drastically, deleting, as well as the combination of all three. In this study, we not only pay attention to the anchor text, we feel the semantics of the surrounding context also play an important part in positioning annotations. We argue the changes for both anchor text and the text of surrounding context could be more complex, since the annotation anchor text could go through different kinds of changes from the annotation’s surrounding context.

All annotation software developed so far has, one way or the other, tried to address or solve the annotation persistence problem. How do we measure the effectiveness or robustness of those methods? What are the guidelines annotation persistence mechanism development should follow? In this section, we reviewed the robustness criteria given by Phelps and Wilensky in their study. Besides the criteria proposed by Phelps and Wilensky, such as, “*Robust to common changes in the referenced document*”; “*Gracefully degrading in the face of increasing change to the document*”; “*Based on document content*”; “*Work with uncooperative servers*” etc., we also emphasize the need to uphold the criterion of *Document Format Neutrality*. This rule requires us that we should not make any assumptions of the internal document structure and our location descriptor should be based on document text only. If adopted, this criterion rules out the

possibility of including a popular document structure like XML or HTML (or XPointers) in the scheme design.

In the last part of this section, we reviewed in more detail a few systems that are robust to varying degrees. Each system tries to solve the annotation persistence problem using different location descriptors and re-attachment algorithms.

In Annotea, annotated documents need to be well-formed structured documents, such as XML. XPointer is used to address into the annotated context within XML/HTML documents. This limits the document format and we believe XPointer is not robust for many common document changes.

Annotator argues the importance of document format neutrality. In Annotator, the location descriptor is based on document text only. It records a portion of text in the anchor's proximity. Its length is determined by a heuristic algorithm, which verifies the uniqueness conditions. Annotator argues that it is against the copyright law to store copies of data from copyrighted material in a publicly accessible database without explicit permission of the publisher. Because of the copyright problem and uncertainty of the length of the location descriptor strings, Annotator hashes the sub-string. Annotator essentially adopts a text matching algorithm to re-attach annotations. The literature has no explanation on how the string is hashed and how the reattachment algorithm works when documents undergoes changes, even a slight one. But simple string matching cannot survive even the moderate modifications of the anchor text.

In Multivalent Annotation, three location descriptors are used to position annotations, a unique identifier (UID); a tree walk; and context. Multivalent annotation assumes document in structured form (XML/SGML), thus it does not meet our robust criterion of document format neutrality. Multivalent annotation uses tree walk as the workhorse for its algorithm. Just like XPointer, we suspect the ability of tree walk as a robust mechanism to survive the complications of significant document modifications.

Microsoft research developed a robust anchor mechanism using keywords. A survey based on the trial of an annotation algorithm suggested that users pay "*little attention*" to the surrounding context of an annotation, indicating that users might not consider the surrounding context very important for annotations made on a range of text. The survey further revealed that unique words in the vicinity of an annotation are distinguishing

anchor characteristics, which should be tracked among successive versions of a document. Acknowledging the need to base the location descriptor only on document text, the algorithm primarily uses unique words from the annotated document to anchor and re-position annotations, and it ignores any specific internal document structure. In their method, the anchor includes such information as: offset from start of document, length of the anchor text, start and end points text of the anchor text, and a list of keywords in the anchor text. The algorithm essentially seeks the best match in terms of the number of matching keywords, their relative distribution in anchor text, the anchor text's initial and ending text match and the anchor's relative position in the whole document. While, we feel the method used in selecting keywords sounds simplistic, we foresee problems will arise when document grows large. The semantic significance of keywords depends on the distribution of the keywords in the documents rather than the frequency alone. In many cases, the method would fail to properly position the annotation. For example, when anchor texts are reworded in a way that it contains quite similar semantic contents, but uses different keywords, adopting this method will fail to provide a match. In another instance, if the anchor texts are reworded in a way that word sequences are changed significantly although they express exactly the same meaning, the matching will give very bad confidence score. We also believe that reattachment algorithms depending only on keyword match are fundamentally flawed, as many times, users make annotations because they are particularly interested in a specific concept, which might not include any significant keyword at all.

As we reach the end of our evaluation, although all systems and annotation persistence mechanisms proposed so far are robust to varying degrees, none fully satisfies our robust criteria. The design of a new mechanism is called for.

In the fourth section of Chapter One, we first defined the problem of annotation persistence over dynamic documents as *a specialized information retrieval* problem. We call it *a specialized IR problem* because of its well-defined user queries and information collection once the documents of interest are pre-defined. In this problem, the user query is the “anchor text + surrounding context” in the original version of the document, and the information collection is the revised version of the document. The IR system should attempt to find the most “relevant” piece of text in the revised version of the document in

response to the user query (“anchor text + surrounding context”) in the original version of the document. If we assume users can make annotations anywhere in a document, the study boils down to *using any piece of text in the original edition of a document to find the most “relevant” piece of text in the revised edition of the document.*

Before we turned our attention to review the IR models, we looked at the three measures to evaluate retrieval performance - *precision, recall and ranking*. Precision is a criterion which measures the proportion of documents in the retrieval set that is relevant to the search. Recall is a measure of the proportion of relevant documents in the retrieval set over the total relevant documents in the document collection. Ranking has to do with whether the result set is ordered in a way that matches the intuitive understanding of what is more and what is less relevant. We expect a good IR models to provide high recall, high precision retrieval with good ranking mechanism.

All IR models can be classified into three classic categories, *Boolean, Vector and Probabilistic*. In the Boolean Model, documents and queries are represented as sets of index terms. In the Vector Model, documents and queries are represented as vectors in a multi-dimensional space. In Probabilistic Model, probability of relevance is calculated based on the assumption that the terms are distributed differently in relevant and non relevant documents.

Before we presented our design of anchor formulation and reattachment algorithm, we took a step back by looking at the problem from another point of view. We looked at the differences between two common ways of searching, “Conceptual search” and “Mechanical word search”. One (“conceptual search”) is used most often by human brains and the other (“mechanical word search”) is well-suited for computer automation. After the brief discussion of the two types of searches, we reviewed one of the retrieval models, *Latent Semantic Analysis*, which allows a computer to do human work, i.e. it retrieve documents based on semantic concept matching.

We studied the human annotation practice. We observed that humans most often make annotations on both important words and important concepts. In terms of annotation reattachment when documents are modified, in the case of important words annotations, we would like to reattach annotations to those important words, although we prefer to select those instances where the context of these words are as semantically close

to the original context as possible. In the case of annotations on concept, when documents are modified, we would like to reposition the annotations to the semantically most similar text in the second version of the documents. In more complicated cases where semantically significant words and concepts are both being addressed in the annotation, our solution should match both semantically significant words and concepts in the second version of the documents. It is difficult to identify domain semantically significant words, in this study, we take semantic concept into consideration only.

In the last part of this section, we presented the strategies to solve annotation persistence over dynamic documents problem. In the design, the anchor formulation includes three location descriptors which address three different aspects of the annotation anchor and surrounding context characteristic. “Meta-structure information location descriptor” tries to capture the Macro-semantic structure context location of the annotation’s anchor; “keywords location descriptor” includes all the “keywords” within annotation’s anchor text; “semantic concept location descriptor” captures the “concepts” annotation’s anchor text and surrounding context contains.

The annotation reattachment robust criteria require that the reattachment algorithm needs to follow the following guidelines. In the case when there are only minor modifications made to annotation’s anchor text and surrounding context, the reattachment algorithm should pick the right location in the revised document with high confidence. In the case when there are increasing changes to the documents, the reattachment algorithm should present a list of possible answers with a ranking with confidence scores. In the case when the document changes are radical, the reattachment algorithm should orphan the annotations rather than reattach them with low confidence.

## Chapter 2

# Natural Language Statistics and Entropy Measure of Keywords

---

In the first chapter, after we made an extensive review of digital annotation classification and practice, we identified the problem of annotation persistence over dynamic documents as a specialized information retrieval problem. To meet the criteria of a robust annotation persistence scheme, we designed our annotation's anchor to include three location descriptors to capture the three different aspects of the annotation's anchor information. One of the location descriptors is to include all the *keywords* annotation's anchor text contains.

What words are *keywords*? Or put another way, what words in a document are statistically more significant so that such words appearing in one version of a document could be used as reliable tokens to uniquely identify the "same" texts in another version of the document? We call the ability of words to resolve document locations as words' indexing power.

Before we delve into the explanation of the entropy measure of *keywords*, we first look at one of the important characteristics of the statistical nature of natural language - the distribution of word occurrences in a text corpus and its governing law - Zipf's Law.

## 2.1 Statistical nature of natural language

### 2.1.1 Zipf's Law

Some words appear more frequently than others. Some words appear in nearly all documents. Many words are infrequent. If we rank all words in decreasing order of

frequency of occurrences, is there a rule for the frequency of occurrences of each word as a function of its rank? The relationship, which was first noticed in the early 1930s by a Harvard linguistic professor George Kingsley Zipf, which is now called Zipf's Law.

There are many ways to state Zipf's Law but the simplest is procedural. Take all the words in a body of text, for example a collection of Wall Street Journal articles, and count the number of times each word appears. If the resulting histogram is sorted by rank, with the most frequently appearing word first, and so on ("a", "the", "for", "by", "and"...), then the shape of the curve is a "Zipf curve" for that text. If the Zipf curve is plotted on a log-log scale, it appears as a straight line with a slope of -1 (Figure 2.1).

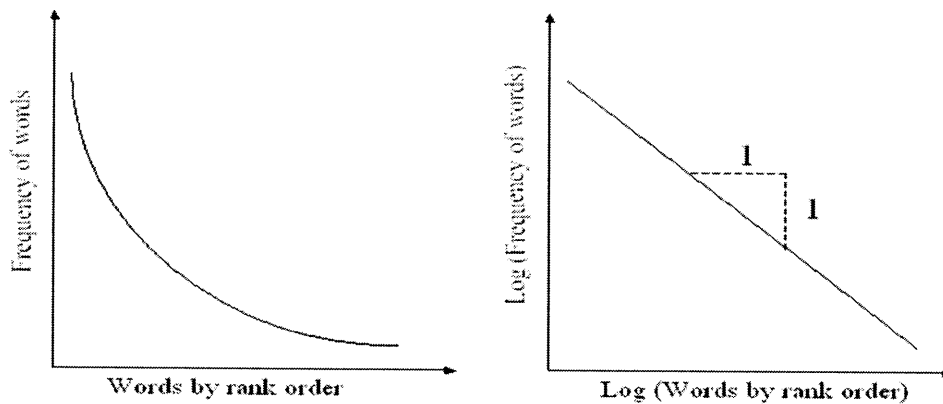


Figure 2.1 Distribution of sorted word frequencies

Zipf stated in his initial publication (Zipf, 1949) that the frequency of the  $r$ -th most frequent word is  $\frac{1}{r}$  times that of the most frequent word.

If we define the following in an English text corpus,

- $r$  : rank of a word;
- $N$  : total number of words in the corpus;
- $D$  : total number of unique words
- $p_r$  : frequency of word with rank  $r$  ( = number of occurrences of word of rank  $r$  divided by  $N$  )

Zipf's Law states:

$$r * p_r = A; \tag{2.1}$$

where  $A \approx 0.1$

In the following, we examined Zipf's Law against a corpus of a collection of three year of Wall Street Journal articles.

Some statistics about the corpus are shown in Table 2.1

<b>Number of documents</b>	150,981
<b>Average length of each document (words)</b>	245
<b>Total number of word occurrences</b>	36,920,947
<b>Number of unique words</b>	164,799

**Table 2.1** Statistics of the corpus of Wall Street Journal articles (1987-1990)

We examined the top 50 most frequent words from the corpus. We ranked them in Table 2.2 and calculated  $r * p_r$  for each ranked word. It is interesting to notice that the most frequent word is “the” and the second most frequent word is “of”. The two most frequent words “the” and “of” account for 8% of total word occurrences. The top 50 words account for 40%.  $r * p_r \approx 0.1$ , although the value of  $r * p_r$  deviates more from 0.1 for the top most frequent words.



Word	Freq	r	$p_r$	$r^*p_r$
the	2040063	1	5.53%	0.055
of	965217	2	2.61%	0.052
to	932307	3	2.53%	0.076
a	845958	4	2.29%	0.092
and	818144	5	2.22%	0.111
in	705402	6	1.91%	0.115
that	364976	7	0.99%	0.069
for	359572	8	0.97%	0.078
one	296023	9	0.80%	0.072
is	278717	10	0.75%	0.075
said	265137	11	0.72%	0.079
point	242651	12	0.66%	0.079
dollars	240667	13	0.65%	0.085
it	225245	14	0.61%	0.085
ten	216988	15	0.59%	0.088
two	205844	16	0.56%	0.089
five	200273	17	0.54%	0.092
as	196998	18	0.53%	0.096
hundred	195534	19	0.53%	0.101
mr	187770	20	0.51%	0.102
by	187717	21	0.51%	0.107
with	186689	22	0.51%	0.111
at	185168	23	0.50%	0.115
trom	177589	24	0.48%	0.115
million	170179	25	0.46%	0.115

Word	Freq	r	$p_r$	$r^*p_r$
percent	170106	26	0.46%	0.120
he	162955	27	0.44%	0.119
its	160000	28	0.43%	0.121
be	156233	29	0.42%	0.123
three	155748	30	0.42%	0.127
was	152985	31	0.41%	0.128
an	140618	32	0.38%	0.122
has	137102	33	0.37%	0.123
are	136416	34	0.37%	0.126
company	134821	35	0.37%	0.128
but	132712	36	0.36%	0.129
have	127041	37	0.34%	0.127
nineteen	126625	38	0.34%	0.130
will	126185	39	0.34%	0.133
seven	125687	40	0.34%	0.136
six	121407	41	0.33%	0.135
four	117019	42	0.32%	0.133
eighty	113837	43	0.31%	0.133
year	113079	44	0.31%	0.135
eight	112377	45	0.30%	0.137
new	110955	46	0.30%	0.138
twenty	100618	47	0.27%	0.128
or	100542	48	0.27%	0.131
about	96949	49	0.26%	0.129
this	93217	50	0.25%	0.126

**Table 2.2** Examination of Zipf's Law against a corpus of three years of Wall Street Journal articles with 36,920,947 total word occurrences; 164,799 unique words

**Predicting Occurrence Frequencies Based on Zipf's Law**

Based on Zipf's Law, we can make some interesting deductions.

If we assume a word that occurs  $n$  times as having a rank of  $r_n$ , then since  $p_r = n/N$ , equation (2.1) becomes,

$$r_n = AN/n \tag{2.2}$$

Since several words may occur  $n$  times, assume rank given by  $r_n$  applies to last of the words that occur  $n$  times. Then we can make this statement,

*$r_n$  words occur more than  $n$  times,  $r_{n+1}$  words occur more than  $n + 1$  times.*

Assume  $I_n$  is the number of words that occur exactly  $n$  times, then we have

$$I_n = r_n - r_{n+1} = \frac{AN}{n} - \frac{AN}{n+1} = \frac{AN}{n(n+1)} \quad (2.3)$$

If the corpus contains  $D$  unique words, then the highest ranking term occurs once and has rank

$$D = AN/1 = AN \quad (2.4)$$

The proportion of words with frequency  $n$  is,

$$I_n/D = \frac{1}{n(n+1)} \quad (2.5)$$

From equation 2.5, we can deduce that:

- the proportion of words occurring once is 1/2
- the proportion of words occurring twice is 1/6
- the proportion of words occurring three times is 1/12

Table 2.3 summarizes the predicted proportion of occurrences compared to the actual proportion of occurrences in the corpus of Wall Street Journal articles.

Number of occurrences ( $n$ )	Predicted proportion of occurrences $\frac{1}{n(n+1)}$	Actual proportion occurring $n$ times $I_n/D$	Actual number of words occurring $n$ times
1	0.500	0.379	57,202
2	0.167	0.140	21,171
3	0.083	0.081	12,209
4	0.050	0.053	8,077
5	0.033	0.039	5,824
6	0.024	0.029	4,396
7	0.018	0.023	3,517
8	0.014	0.019	2,894
9	0.011	0.017	2,504
10	0.009	0.014	2,110

**Table 2.3** Frequencies from a corpus of three year of Wall Street Journal articles with 36,920,947 total word occurrences; 164,799 unique words.

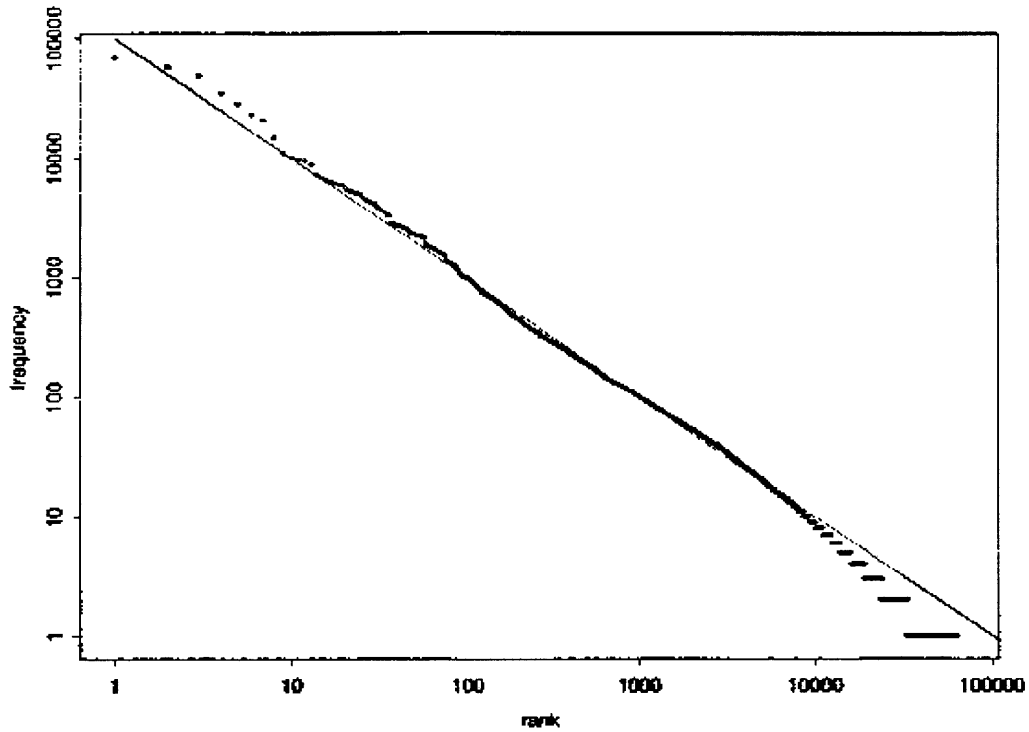
We can make the following observations from the Table 2.3:

- Nearly half of the words in the corpus appear only once and twice
- About 70% of the words in the corpus appear less than 5 times.

### Zipf's Law and Reality

A law of the form  $y = k * x^c$  is called a power law. Zipf's Law is a power law with  $c = -1$ . On the log-log plot, power laws give a straight line with slope  $c$ , or  $-1$  in the case of Zipf's Law.

Figure 2.2 is the comparison of Zipf's Law to the real data from Brown Corpus (Croft, 2001). Zipf's Law is quite accurate except for very high and very low ranks.

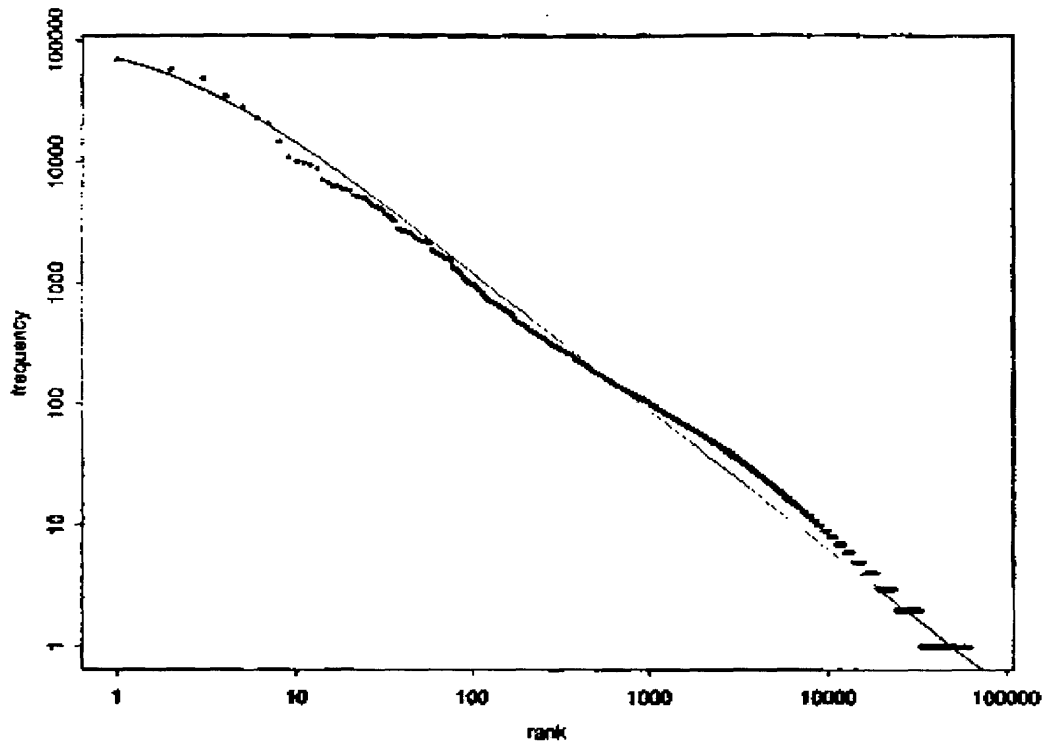


**Figure 2.2** Zipf on Brown Corpus (Croft, 2001)

A more general power law was introduced by Benoit Mandelbrot (1954). Mandelbrot's generalization of Zipf's Law is still very simple: the additional complexity lies only in the introduction of the two new adjustable constants, a number added to the rank ( $\rho$ ) and a number added to the power ( $\beta$ ), Mandelbrot's Law is shown in Equation 2.6.

$$p_r = A * \frac{1}{(r + \rho)^\beta} \quad (2.6)$$

Figure 2.3 shows the optimal fit of Mandelbrot's Law to Brown corpus with parameter of  $\rho = 100$  and  $\beta = 1.15$ .



**Figure 2.3** Mandelbrot’s function on Brown corpus (Croft, 2001)

### **Explanation of Zipf’s Law and other applications**

Zipf’s Law is an experimental law, not a theoretical one. The causes of Zipfian distributions in real life are a matter of some controversy. Zipf attributed it to the “principle of least effort”. In the case of English text, the nature of communication is such that it is more efficient to place emphasis on using shorter words. Hence the most frequent words tend to be short and appear often. The underlying theme is that efficiency, competition, or attention with regards to resources or information tends to result in Zipf’s Law distribution.

Many social and natural phenomena have been shown to follow Zipf’s Law, including:

- Populations of cities
- Income of companies
- Income of individuals

- Size of earthquakes
- Size of settlements

### **2.1.2 Impact of Zipf's Law on Information Retrieval**

Most large English text corpora follow Zipf's Law and thus have similar statistical characteristics. These statistics influence the effectiveness and efficiency of data structures used to index documents. They are the bases for determining words with significant resolving power - the ability to discriminate document contents. There are also implications from the statistics for infrequent words, which demonstrate significant indexing power – the ability of identifying documents that contain unique words. In the following, we look at the implications of Zipf's law on the whole spectra of words, from common words which appears often to rare words which seldom show up in a document.

#### **Stopwords**

A few words are very common. In the examination of the three year Wall Street Journal articles, we observed that the most two frequent words “the” and “of” account for 8% of total word occurrences. Top 50 words account for 40% of total word occurrences. Words that appear too frequent don't carry any real semantics of the document content. They are so called *stopwords* in the IR world. A stopword is a word that does not carry meaning in natural language and therefore can be ignored, such as ‘a’, ‘the’, ‘by’ etc. In document indexing, a stopword list is usually used to eliminate the high frequency words. This allows us to significantly reduce the space overhead of indices for natural language texts.

#### **Infrequent words or words with rare appearances**

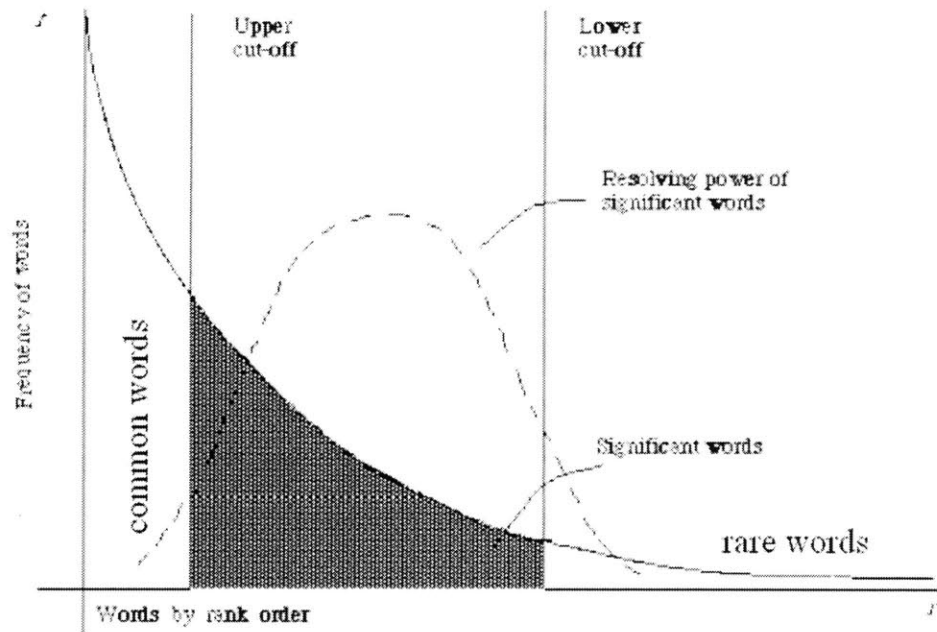
Many words appear very infrequently. In the study of the three years Wall Street Journal articles, nearly half of the words in the corpus appear only once or twice. About 70% of the words in the corpus appear less than 5 times. From the information retrieval point of view, it is very difficult to gather sufficient data on rarely occurring words for meaningful statistical analysis. (e.g. for correlation analysis for query expansion). Infrequent words, however, possess strong indexing power, the ability for words to resolve uniquely the documents they resides.

## Words in the middle

What can we say about words in the middle of the spectrum of word occurrences? Luhn (1958) stated that they are the words which have strong resolving power - the ability to discriminate document contents.

In 1958, Luhn published a paper titled “The automatic creation of literature abstract”. In the paper, Luhn states: *“It is here proposed that the frequency of word occurrence in an article furnishes a useful measurement of word significance. It is further proposed that the relative position within a sentence of words having given values of significance furnish a useful measurement for determining the significance of sentences. The significance factor of a sentence will therefore be based on a combination of these two measurements.”* He made an assumption that frequency data can be used to extract words and sentences to represent a document.

Luhn proposed two cut-offs to Zipf’s curve (Figure 2.4). The upper cut-off excludes all common words; the lower cut-off removes rare ones. Luhn states that both common and rare words don’t contribute significantly to the content of the document. Luhn went on further to claim that the resolving power of words to discriminate content (or semantics) reaches the peak at a rank order position half way between the two cut-offs, and decays to zero for rare and common words. Thus words in the middle are significant in document semantic analysis. In Chapter four, when we explain Latent Semantic Analysis, we will explore more of Luhn’s idea.



**Figure 1.4** Luhn cut-offs (Rijsbergen, 1979)

## 2.2. Information Theory and Entropy Measure of Keywords

In implementing the annotation persistence mechanism proposed in Chapter One, we need to answer several questions. One of them is about *keywords*. Assume we are given a document; we subsequently make annotations on arbitrary document locations. As we proceed to produce, as designed in Chapter one, each annotation's anchor representation, one of the location descriptors, the *keyword location descriptor*, needs to include a list of keywords this annotation's anchor text contains. How are we going to generate a list of *keywords* automatically based on the annotation's anchor text?

In this chapter, we delay presenting the full strategy to automatically generate the keyword list based on an annotation's anchor text to Chapter Four. However, we present the central ingredient of keyword selection --- the criteria we use to evaluate which word is statistically more significant than others.

In Chapter One, we reviewed a study conducted by Microsoft Research that developed a robust anchor mechanism using keywords. In their implementation, keyword selection is based on word frequencies in a document. The less frequent a word appears



in a document, the more important it is. The algorithm starts words that occur once, twice and so on.

Is word frequency the best criterion for determining *keywords*? Assume we have a large text document and two words appear with equal frequencies (or occurrences) in the document. One word, however, clusters to a few document locations; while the other is dispersed more evenly in the document. Which word is statistically more significant? We say word one. Word one points to a fewer document locations and it is less evenly distributed inside the document. When used as an index, word one can resolve document locations with more certainty, thus with stronger confidence - we say it has *greater indexing power*. The greater the word indexing power is, the statistically more significant the word becomes and more readily this word can be used to find unique parent documents.

Can we measure word's indexing power quantitatively? The answer is yes. The tool is entropy.

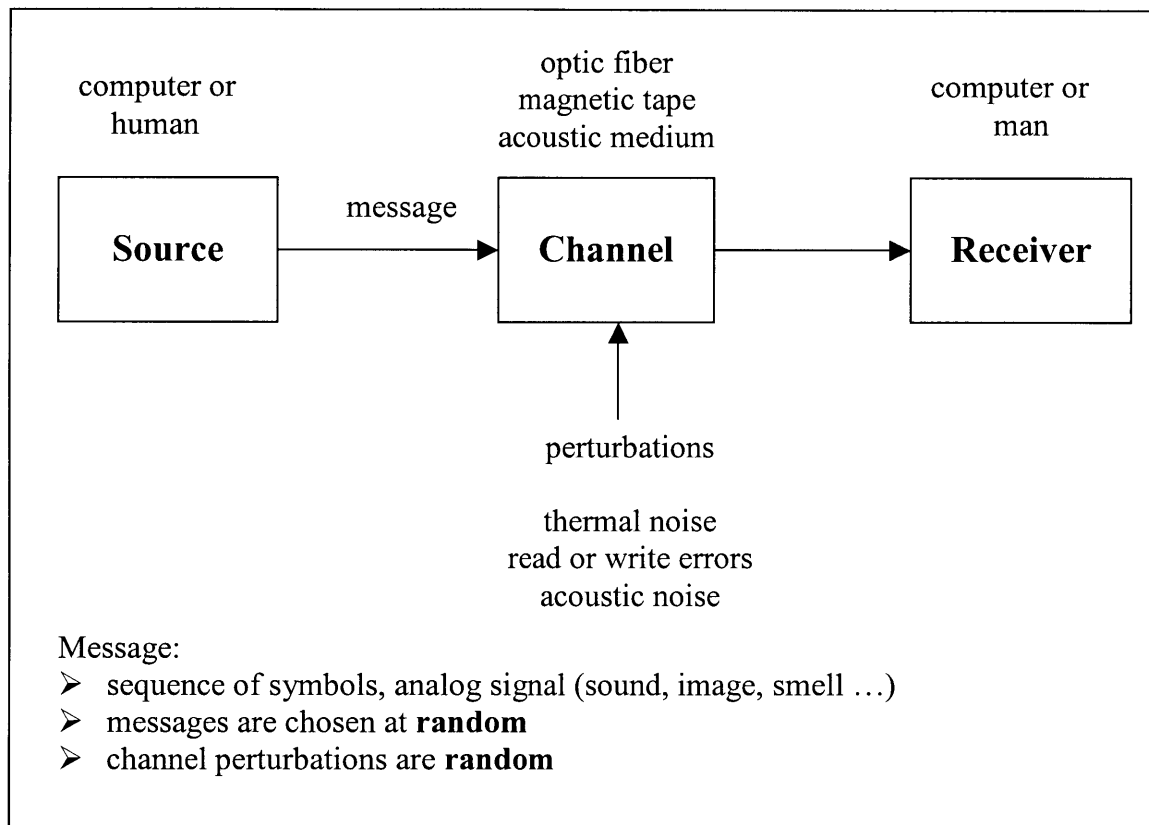
In the following, we first review briefly Information Theory and Shannon's Entropy. We then present the formulation of *Normalized Word Entropy*, which is used as our word significance indicator to evaluate the *keywords*. Lastly we report the evaluation of *Normalized Word Entropy* against sample words from the Wall Street Journal articles.

### **2.2.1 Information Theory and Shannon's Entropy**

#### **Information Theory**

Claude E. Shannon laid down the foundation of information theory in his landmark paper entitled *A mathematical theory of communication* (Shannon, 1948). In this paper, Shannon presented all the main theoretical ingredients of modern information theory. In particular, Shannon formulated and provided proofs of the two main coding theorems.

Shannon's theory of communication is based on the so-called Shannon paradigm (Figure 2.4).



**Figure 2.4** Shannon paradigm

A data source produces a message which is sent to a receiver through an imperfect communication channel. The possible source message can generally be modeled by a sequence of symbols, chosen in some way by the source which appears as unpredictable to the receiver. In other words, before the message has been sent, the receiver has some uncertainty about what will be the next message.

Most real life physical channels are imperfect due to the existence of some form of noise. This means that the message sent out will arrive in a corrupted version at the receiver (some received symbols are different from those emitted), and again the corruption is unpredictable for the receiver and for the source of the message.

The two main questions posed by Shannon in his early paper are as follows:

- Suppose the channel is perfect (no corruption), and suppose we have a probabilistic description (model) of the source, what is the maximum rate of communication (source symbols per channel usage), provided that we use an appropriate source code. The problem is presently termed as the source coding problem or as the reversible data compression problem. The answer to this question is given by Shannon's entropy of the source.
- Suppose now that the channel is noisy, what is then the maximum rate of communication without errors between any source and receiver using this channel? This is the so-called *channel coding problem* or *error-correction coding problem*. The answer here is the *capacity* of the channel, which is the upper bound of mutual information between input and output messages.

The two main results of information theory are thus the characterization of upper bounds in terms of data compression on the one hand, and errorless communication on the other.

A further result of practical importance is that (in most, but not all cases) source and channel coding problems can be decoupled. In other words, data compression algorithms can be designed independently from the type of data communication channel that will be used to transmit (or store) the data. Conversely, channel coding can be carried out irrespective of the type of data sources that will be used to transmit information over them. This result has led to the partition of coding theory into its two main subparts.

Source coding aims at removing redundancy in the source messages to make them appear shorter and purely random. On the other hand, channel coding aims at introducing redundancy into the message to make it possible to decode the message in spite of the uncertainty introduced by the channel noise.

### **Shannon's Entropy**

A key feature of Shannon information theory gives *information* a numeric measure based on probabilistic model. Solutions of many important problems of information storage and the transmission are then formulated in terms of this important measure about *information*. The numeric measure of *information* has a very concrete optional

interpretation: it roughly equals the minimum number of bits needed, on average, to encode the message in question.

Let  $X$  be a discrete random variable taking a finite number of possible values  $A_x = \{x_1, x_2, \dots, x_n\}$  with probabilities  $\{p_1, p_2, \dots, p_n\}$  respectively such that  $p_i \geq 0$ ,

$$\sum_{x \in A_x} P(x) = 1.$$

The entropy of  $X$  is defined by

$$H(X) \equiv \sum_{x \in A_x} P(x) \log \frac{1}{P(x)} \quad 2.7$$

with the convention for  $P(x) = 0$  that  $0 \times \log 1/0 \equiv 0$ , since  $\lim_{\theta \rightarrow 0^+} \theta \log 1/\theta = 0$ . The expression (2.7) is famous as *Shannon's entropy* or *measure of uncertainty*.

The concept of Shannon's Entropy play a central role of information theory and is sometimes referred as the *measure of information content* or the *measure of uncertainty*. The entropy of a random variable is defined in terms of its probability distribution and can be shown to be a good measure of randomness or uncertainty.

Shannon's entropy has following properties,

$$H(X) \geq 0 \text{ with equality if } p_i = 1 \text{ for one } i.$$

$H(X) \leq \log(|X|)$  with equality if  $p_i = 1/|X|$  for all  $i$ . ( $|X|$  denotes the number of elements in the set  $A_x$ .)

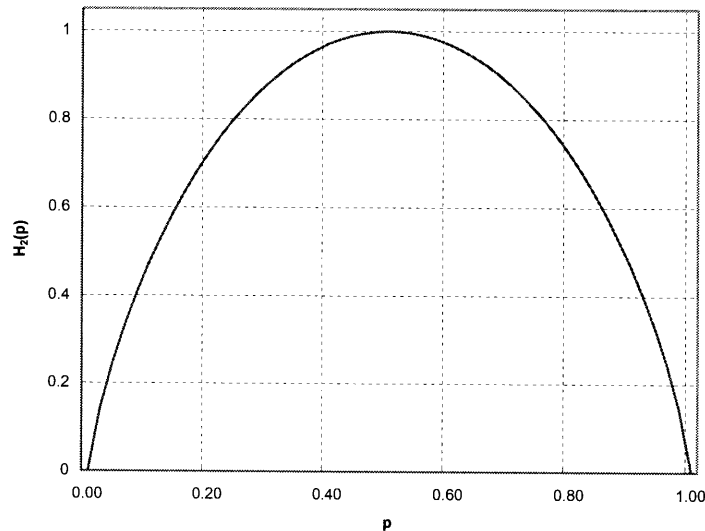
It should be noted that entropy is a function of the distribution of  $X$ . It does not depend on the actual values taken by  $X$ , but only on the probabilities.

A classic demonstration of entropy is the entropy measure of a binary source (e.g. a biased coin toss).  $H(X) = -p \log p - (1-p) \log(1-p) = H_2(p)$ , where  $p$  denotes the probability of either of the two values of  $X$ . Figure 2.5 shows the binary entropy function  $H_2(p)$  with respect to  $p$ .

Properties of  $H_2(p)$ :

- $H_2(p) = H_2(1-p)$
- $H_2(0) = H_2(1) = 0$

- $H_2(0.5) = 1$  (if two is used as the base for the logarithm)
- $H_2(p) \leq 1$



**Figure 2.5** Binary entropy function:  $H_2(p)$

How do we interpret entropy measure of a binary system, for example, if the system is a coin toss?

In a system of coin tosses, if the coin is a fair coin with head and tail appearing with equal probabilities ( $p = 0.5$ ), the entropy of the system (coin tosses) attains its greatest value and measures 1 if the base 2 is used for the log function. From the data compression point of view, for efficient encoding, each coin toss requires 1 bit of information to transmit. From the point of view of system uncertainty, the system of a fair coin toss is the most uncertain binary system, thus its entropy measures the highest.

In case of a biased coin toss, the entropy of the system is less than 1. For efficient encoding, a sequence of biased coin tosses contains less “information” than a sequence of unbiased tosses, so it require on average less than 1 bit to transmit.

For heavily biased coins, the entropy approaches zero, which means that system contains very little “information” and it is a very “certain” system. We almost know what the outcome of the each coin toss is, thus it requires on average near zero bits to transmit.

Shannon’s entropy is a measure of uncertainty of the message. It depends on the receiver’s prior knowledge as well as the message itself. The more uncertain the message is, the higher the Shannon’s entropy values. In the case of a random variable, the more uniformly distributed, the higher the entropy becomes.

### 2.2.2 Normalized Word Entropy

In word analysis, the more uniformly a word is distributed in document collections, the less semantic it carries (such as stopwords). The more skewed a word is distributed in a document collection, the more indexing power it has (such as rare occurring words). For the first case, the entropy measure is very high, while for the later, the entropy is close to zero.

Here we present the definition of *Normalized Word Entropy*:

We define the following from a text corpus  $T$ ,

$N$ : total number of documents;

$w_i$ : word  $i$ ;

$d_j$ : document  $j$ ;

$c_{i,j}$ : number of times  $w_i$  occurs in  $d_j$ ;

$t_i = \sum_j c_{i,j}$ : total number of times  $w_i$  occurs in the corpus  $T$ ;

We define *Normalized Word Entropy*  $\varepsilon_i$  for  $w_i$ ,

$$\varepsilon_i = -\frac{1}{\log N} \sum_{j=1}^N \frac{c_{i,j}}{t_i} \log \frac{c_{i,j}}{t_i} \quad 2.8$$

$\varepsilon_i$  has following properties,

- $0 \leq \varepsilon_i \leq 1$ ;
- $\varepsilon_i = 1$  if  $c_{i,j} = t_i/N$ , meaning  $w_i$  occurs equal times in all documents

➤  $\varepsilon_i = 0$  if exists  $j$ , where  $c_{i,j} = t_i$ , meaning  $w_i$  occurs only in one document.

A value of  $\varepsilon_i$  close to 1 indicates a near uniform distribution of  $w_i$  across all documents in the corpus. A value of  $\varepsilon_i$  close to 0 indicates  $w_i$  is present in very few documents.

The smaller  $\varepsilon_i$ , the stronger  $w_i$ 's indexing power is, and thus the more statistically significant  $w_i$  is.

Table 2.4 shows the normalized entropy measurements of sample words from the corpus of the three years Wall Street Journal articles. For each row in the table, the normalized word entropy is shown along side with the frequencies (or occurrences) of the word in the corpus.

Words	Normalized Entropy	Frequency
The	0.9321	2040063
Stock	0.8312	65644
Bank	0.7737	40920
Bush	0.6107	6533
Bailout	0.5163	774
Chile	0.4308	496
Cinema	0.3806	373
Aviator	0.2167	18
Bandit	0.1816	10
Splat	0.0504	11
Spurge	0.0450	13
Virago	0.0273	10
Spitball	0.0196	16
adveryorials	0.0173	19
Ballad	0.0000	18
Gunsmith	0.0000	1

**Table 2.4** Normalized word entropy for sample words in the corpus of three years of Wall Street Journal articles.

Table 2.4 helps us to draw the following conclusions on normalized work entropy.

- Stopwords have high entropy value.  $H(\text{stopwords}) \approx 1.0$ .
- Words occurring only once in the corpus have entropy value of zero.
- Generally, the less frequently the word appears in the corpus, the smaller the entropy value is.
- In words that appear equally frequently, entropy is larger for those that are distributed in more documents, smaller for those that are clustered in a few documents.
- When word entropy is smaller, the word has stronger indexing power.

We would also like to point out that word entropy is corpus related, word “stock” may have a different entropy value in the domain of financial corpus and corpus on computer science education.

## 2.3 Summary

In this chapter, we started by reviewing Zipf’s Law, which governs the statistics of word occurrences. We evaluated Zipf’s Law against a corpus of the three years of Wall Street Journal articles. Luhn separates the spectra of ranked words into three areas. Common words are those which appear very often in documents. Rare words seldom appear in documents. Luhn declares words in the middle ranks as significant words which demonstrate strong resolving power.

Common words are also called stopwords in IR. The most two frequent words can account for 10% of total word occurrences. The top 6 words usually compose 20% and top 50 words nearly 50% of total word occurrences. As common words don’t carry the semantics of the contents, a stopword list is usually used in document indexing to eliminate all common words. This helps to significantly reduce the space overhead of indices for natural language texts.

Rare words compose a large portion of the word vocabulary. In our evaluation of the corpus of three years of Wall Street Journal articles, nearly half of the words in the text corpus appears only once or twice. About 70% of the words in the corpus appear less than 5 times. We claim infrequent words have strong indexing power, implying they can be used as tokens to almost uniquely identify the documents from which the words are taken.



Luhn stated that words in the middle range of the spectrum of the ranked word list have strong resolving power; they are strong contributors to the content and semantics of the documents. Because of their significant semantic load and relatively rich occurrences in the document corpus, they are used exclusively to compare semantics between documents.

In the second half of the chapter, we presented entropy criteria to measure words' indexing power – the ability for words to resolve document locations with strong certainty. The greater the word indexing power is, the statistically more significant the word becomes, and more reliably the word can be used to find unique document locations.

We reviewed briefly Information Theory and Shannon's Entropy. We then present the formulation of *Normalized Word Entropy*, which is used as our word significance indicator to pick out the keywords which have strong indexing power.

Shannon studied the theoretical limits for data compression and transmission rates. The two main results of information theory are the characterization of upper bounds in terms of data compression on the one hand, and error-less communication on the other. In Shannon's theory, compression limits are given by Entropy and transmission limits are given by Channel Capacity.

Shannon's entropy measures the expectation of information content. It is a measure of uncertainty. The more uncertain the system is, the higher the entropy of the system. In the case of a random variable, the more uniformly distributed the variable is, the higher the entropy of the variable.

We presented the formulation of *Normalized Word Entropy*, which gives a  $[0 - 1]$  value for each word. *Normalized Word Entropy* measures the word distributions for each word in a text corpus. The more uniformly the word is distributed in the documents, the higher the *Normalized Word Entropy*, and closer the entropy value approaches to 1. The more skewed the word is distributed in the corpus, the lower the *Normalized Word Entropy* is. In the case if a word appears only once in a text corpus, the *Normalized Word Entropy* of this word is zero.

The smaller the *Normalized Word Entropy*, the greater the word indexing power. Thus words with small normalized word entropy are better keywords which can be used to resolve uniquely the parent document locations.

## Chapter 3

# Latent Semantic Analysis

---

In chapter one, we designed the annotation’s anchor to include three location descriptors to capture three important characteristics of the annotation’s anchor information:

- *Meta-structure information location descriptor*
- *Keyword location descriptor*
- *Semantic concept location descriptor*

When making decisions on transferring annotations between versions, the reattachment algorithm needs to compare the three location descriptors of the original annotation anchor with those of candidate annotation anchors. A heuristic evaluation mechanism is then used to weigh the comparison results to select the best match.

In chapter two, we presented a criterion to measure the “significance” of words. We made an assumption that keywords contained in keyword location descriptors should present strong indexing power – the ability for words to resolve document locations with strong certainty. The observation of human annotation practice on *keywords* also tells us that most of the keywords users annotate possess strong semantics and often appear “rarely” in documents. For example, a keyword readers annotate often in an object oriented programming (OOP) textbook is “polymorphism”, which represents a strong semantic concept in OOP and usually appears in just a few selected locations in the textbook.

Our criterion in measuring “keywords” is *Normalized Word Entropy*, which falls in the range of [0, 1] for each word. The smaller the *Normalized Word Entropy*, the greater the words indexing power.

In this chapter, we review and evaluate a technique supporting the formation and usage of the semantic concept location descriptor – Latent Semantic Analysis (LSA).

LSA will help us to answer questions such as, “How are semantic concepts extracted from texts?” and “How can we compare the semantic similarities between texts?” Before we delve into the detailed explanation of how semantic concept location descriptors are composed and used, we first review the theory of Latent Semantic Analysis.

In the first section of Chapter 3, we will present a brief introduction to LSA. We review the two salient problems that plague the lexical text matching methodologies in IR. We look at the assumptions LSA makes on the existence of implicit higher semantic structures in word-document associations. In section 2 of this chapter, we review the theoretical background of LSA and its methodologies. In section 3, we review the applications of LSA in information retrieval, information filtering, knowledge induction and representation, and text-based research. In section 4, we evaluate LSA against a text corpus of three years of Wall Street Journal articles and lay out the foundations for measuring and comparing an annotation’s anchor semantics.

### **3.1 Introduction**

Automatic document indexing combined with lexical word matching still are the predominant information retrieval methods for text documents, especially those on World Wide Web. Current full text search engines, no matter how complex, find their results based on exhaustive word search and lexical word matching. While the technique used to rank the results may be very sophisticated (Google is an example of innovation in choosing a system for ranking), the actual search is based entirely on word matches, with no higher-level understanding of the query or documents.

In the case of a search for a conceptual topic, lexical matching methods can be unreliable and inaccurate. The problem is two-fold. On the one hand, individual words indexed for the documents provide unreliable evidence about the conceptual topic or meaning of a document. On the other hand, the words that users use to search often are not the same as those words indexed from for the documents of interest. Deerwester et. al. (1988) attribute the deficiencies of the lexical word matching methods to two well-known language related problems, *synonymy* and *polysemy*.

*Synonymy*. There are many ways to express a concept. Words people use to describe semantic concepts usually depend on people’s educational background, their knowledge

of the subject area, linguistic habits or even personal preferences. Synonymy means that an object can be referred in many ways, i.e., people use different words to search for the same subject. A typical example is the words *car* and *automobile*; where both refer to nearly exactly the same semantic object. One is used more often in formal settings, however, than the other. Based on lexical text matching, a query containing only “car” will not be able to retrieve documents which contain “automobile” but not “car”. The prevalence of synonyms tends to decrease the recall performance of retrieval systems.

*Polysemy.* Polysemy is the problem of words having more than one specific meaning. An example of this is the word *jaguar* that could mean a well-known car type or an animal. A query term including “jaguar” will retrieve documents on both the car brand and the animals if lexical matching is used. Polysemy is one of the factors which lead to poor precision.

Methods have been proposed to improve the poor recall performance of the IR system due to synonymy. One of the proposals is to use automatic term expansion or construction of a thesaurus. Term expansion uses term matching, but augments a user’s original terms with related words, e.g., from a special thesaurus, in hopes of hitting more targets in the collection. For experienced searchers, this method provides more search terms, but pays a price in scatter. Terms with multiple meanings may hit spurious targets, leading to rapid degradation of precision (Jones, 1972).

LSA offers a dampening effect on synonymy, though the effect on polysemy is less pronounced.

LSA assumes there exist implicit higher semantic structures, known as latent semantic structures, in term-document associations. Terms tend to be similar if they appear in the same kind of documents, whether or not they actually occur within identical word contexts in those documents. Documents are semantically close if they have many similar words in common, and semantically distant if they have few words in common. This assumption correlates very well with how a human being, looking at content, might classify a document collection.

LSA examines the document collection as a whole; it looks at patterns of word distribution (especially word co-occurrence) across a set of documents. LSA starts with the formation of a *term-document matrix*, a matrix with all documents from the collection

listed along the rows, and all content words from the collection along the columns. Each cell of the matrix initially contains the frequency of the word (the row) appearing in the document (the column). Cell values are usually subject to modifications by global and local term weights.

The Term-document matrix is always sparse as each document usually contains only a tiny fraction of the content word vocabulary. The size of the matrix depends on the size of the corpus. Its sparsity is usually in the range of 0.1% to 0.5%.

The key step in LSA is decomposing this matrix using a technique called *singular value decomposition* (SVD). By performing SVD on a term-document matrix, the original matrix is decomposed into three matrices, a left orthogonal matrix; a center diagonal matrix; and a right orthogonal matrix. The dimension of the transformed space is reduced by selection of the highest singular triplets, where the majority of the variance of the original space lies. The reduced space reveals the latent semantic structure in the term-document associations. By using SVD, the major associative patterns of terms and documents are extracted from the document space, and the smaller patterns are ignored.

LSA works by projecting the usually large, multi-dimensional term-document space (measured in the thousands) down into a smaller number of orthogonal “semantic” dimensions (say 300). Intuitively, in doing so, words that are semantically similar will get clustered together, and will no longer be completely distinct.

In the process of dimension reduction, information is lost. Information loss sounds like a bad thing, but in reality it removes noise. By dimension reduction, more subtle and minor associative patterns of words and documents are removed, revealing major association patterns that are latent in the document collection. Similar things become more similar, while dissimilar things remain distinct. This reductive mapping is what gives LSA its seemingly intelligent behavior of being able to correlate semantically related terms and documents. Essentially, we are really exploiting a property of natural language, namely that words with similar meaning tend to occur together.

The computational advantage of LSA is that it is fully automatic and does not use language expertise.

## 3.2 Theoretical Background and Methodologies

### 3.2.1 Term-Document Matrix

To discover the major latent semantic structures in word-document associations, LSA first constructs a matrix of co-occurrences between words and documents, namely the term-document matrix (term is used here instead of word).

Before the construction of the term-document matrix, the document preprocessing is typically performed for all documents in the corpus.

#### Document Preprocessing

Document preprocessing is a procedure that normally includes several text operations in IR. Baeza-Yates (1999) listed the following common text operations under document preprocessing:

- Lexical analysis of the text with the objective of removing digits, hyphens, punctuation marks, and word capitalization.
- Elimination of stopwords with the objective of filtering out non-semantic carrying words.
- Stemming of words with the objective of removing affixes and retrieving documents containing syntactic variations of query terms.
- Selection of index terms with the objective of selecting proper words/stems to index documents.

In the following, we explain the need of each text operation for LSA as well as the procedures to perform each text operation adopted in this study.

#### *Lexical analysis of text*

LSA pays no attention to the order of words or the syntactic structure of passages, which makes it different from other language modeling methods, such as n-gram modeling or context free grammar. LSA takes a so-called “bag-of-words” paradigm, which disregards collocational information in word strings. When making a lexical analysis of text for each document, all information except the collection of words

composed purely of alphabetic characters (with no order) is removed, such as digits, punctuations, hyphens, quotation marks and word capitalizations.

### *Stopwords*

Many of the words in documents carry no semantic value. These include articles, conjunctions and other functional words (the, and, despite), as well as words that are too ubiquitous across the corpus to have any real meaning. They are called stopwords in IR.

In document indexing, they are normally eliminated. LSA is a methodology trying to extract semantic structures latent in the association of words and documents. Non-semantic bearing words, such as stopwords don't contribute to this analysis. The usual way of dealing with stopwords in LSA, is to eliminate them entirely from the term list.

Creating a stopword list is not a trivial task, however. It is something of an art, since the choice of stopwords depends very much on the nature of the data collection and the frequency threshold used in deciding what constitute a "frequently used" word.

To avoid making clear a decision on what words should be stopwords, a different strategy can be used in which all words except very low frequency words are included in the construction of term list. After construction of the term-document matrix, a global weight is then calculated for each word, which in turn is then applied to all the cell values in each row of the term-document matrix. The global weight is set near zero for stopwords, thus the impact of stopwords on LSA is minimized. We will elaborate more on this approach when we discuss the term-weighting strategies in the later part of this section.

### *Stemming*

A stem is the portion of a word that is left after the removal of its affixes. Stemming is the process of changing all variants of words to their respective stems or to a standard form like the infinitive for verbs. Stemming is thought to be useful for improving IR performance because it reduces the variants of the same root word to a common root and it reduces the number of distinct index terms. The argument supporting stemming seems sensible, though there is controversy in the literature about the benefits of stemming for retrieval performance (Baeza-Yates, 1999).



No consensus has been reached in the IR community about the effect of stemming on IR. In fact, different studies lead to conflicting conclusions. Frakes (1992) compared eight distinct studies on the potential benefits of stemming. In his study, he favored stemming, but the results of the eight experimental studies he investigated don't provide satisfactory consensus to support stemming. As a result of the uncertainty over the benefit of stemming, many web search engines don't use any stemming algorithm whatsoever.

In a sense, stemming is done to capture likely synonyms. Since LSA deals with clustering synonyms to some extent, the additional value of stemming is an open question (Deerwester, 1988). If words with the same stem are used in similar documents, they are clustered closer to each other; otherwise they are clustered apart. For example, in analyzing an encyclopedia, *doctor* is likely to occur in the same articles as *doctors* but less likely to occur with *doctoral*. In this study, we don't use stemming.

### *Selection of (indexing) terms*

To construct the term-document matrix, we need to make decisions on selecting terms, though the terms selected are not for indexing purpose (we don't index terms in LSA). Selecting terms in LSA follows same principles as that of selecting terms for indexing.

Earlier we present Luhn's cut-offs to Zipf's curve (Figure 2.4), in which Luhn stated that words in the middle of the spectra of word occurrences have strong resolving power - the ability to discriminate document contents. The upper cut-off excludes all common words, while the lower cut-off takes off rare ones. Luhn states that both common and rare words don't contribute significantly to the content of the document. Luhn went on further to claim that the resolving power of words to discriminate content (or semantics) reaches a peak at a rank order position half way between the two cut-offs, and decays to zero for rare and common words. Thus words in the middle are significant in document semantic analysis.

Terms that appearing too often (stopwords) should be eliminated. In our implementation, we don't eliminate them, but rather applying global weights which are close to zero to them to minimize their influences on the analysis.

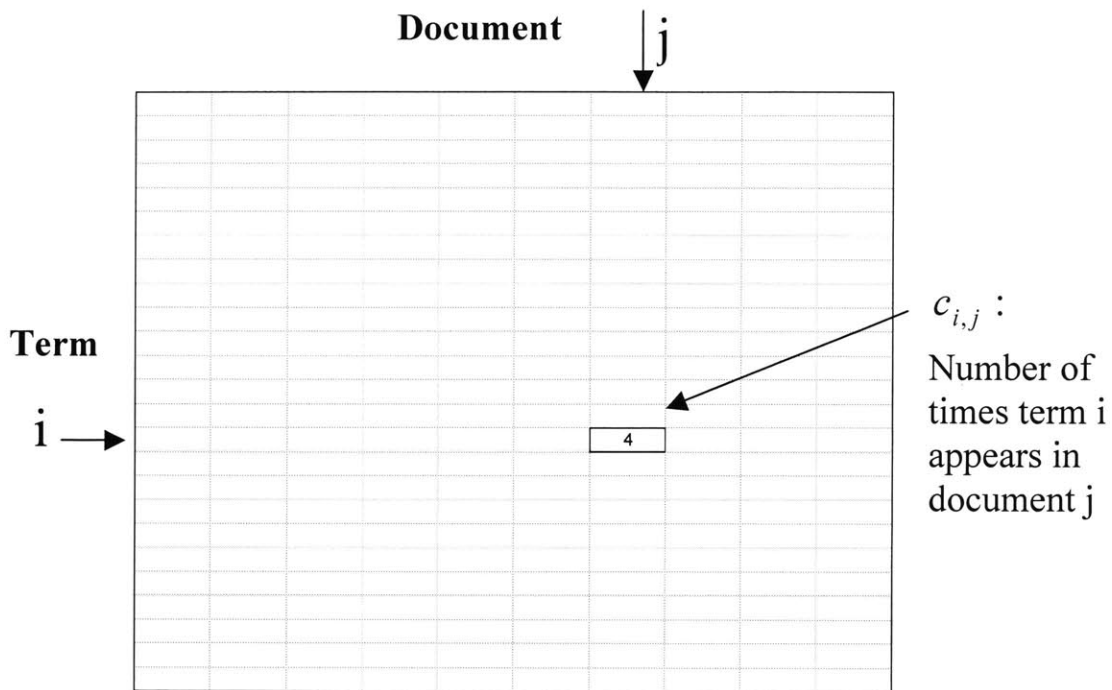
Terms that appear rarely in the corpus have little association power. They are not included in the term list when constructing it. The term list in this study is constructed by

selecting top ranking words from a rank-sorted word list; thus low ranked (e.g. those that appear only once or twice) words are essentially eliminated in the term list.

### **Construction of Term-Document Matrix**

Let  $V$ ,  $|V| = M'$ , be the underlying vocabulary and  $T$  a training text corpus, comprising  $N$  articles (documents). Typical order of  $M'$  and  $N$  is in the orders of 10,000 and 100,000, respectively.  $T$  may comprise a hundred million words or so. LSA first takes the top  $M$  ranked terms from  $V$ , excluding terms that appear rarely in the corpus (e.g. total occurrences less than three times). Zipf's Law tells us normally  $M' < \frac{1}{2}M$ .

We then construct a matrix of term-document co-occurrence,  $A$ . On the rows are elements of the truncated term list of total vocabulary with  $M$  unique terms. On the columns are  $N$  documents. The value of each cell is the occurrences of term  $w_i$  in document  $d_j$ .



**Figure 3.1** term-document matrix

### Term Weighting

In Chapter 1, when reviewing the Vector Model in IR, we pointed out the desire to apply term weights to index terms. The same logic applies to cell values of the term-document matrix. Dumais (1987) suggested applying both local and global weightings to increase and decrease the importance of terms within or among documents. Dumais (1987) further suggested using word entropy as global weights and normalizing document length as local weights.

Using the same terminology we used to decide word entropy in Chapter 2, we have the following definitions from an English text corpus  $T$ ,

- $N$  : total number of documents;
- $w_i$  : word  $i$ ;
- $d_j$  : document  $j$ ;
- $c_{i,j}$  : number of times  $w_i$  occurs in  $d_j$ ;

$t_i = \sum_j c_{i,j}$ : total number of times  $w_i$  occurs in the corpus  $T$ ;

$n_j$ : total number of words presented in  $d_j$ ;

the *Normalized Word Entropy*  $\varepsilon_i$  for  $w_i$  is then defined as,

$$\varepsilon_i = -\frac{1}{\log N} \sum_{j=1}^N \frac{c_{i,j}}{t_i} \log \frac{c_{i,j}}{t_i} \quad 3.1$$

Dumais suggested applying the following weight to cell  $(i, j)$ ,

$$(1 - \varepsilon_i) \frac{c_{i,j}}{n_j} \quad 3.2$$

The global weighting implied by  $1 - \varepsilon_i$  reflects the fact that words appearing in the corpus don't carry the same semantic weight. For stopwords, the global weighting  $1 - \varepsilon_i$  approaches zero, which effectively removes the influence of stopwords.

### 3.2.2 Singular Value Decomposition

#### Theoretical Background

Singular value decomposition is closely related to a number of mathematical and statistical techniques in a wide variety of other fields, including eigenvalue analysis, spectral analysis, and factor analysis.

Given an  $(m \times n)$  matrix  $A$ , without loss of generality, assume  $m \geq n$  and  $\text{rank}(A) = r$ , the SVD of  $A$ , denoted as  $\text{SVD}(A)$ , is defined as

$$A = USV^T \quad 3.3$$

where  $U^T U = V^T V = I_n$  and  $S = \text{diag}(s_1, \dots, s_n)$ ,  $s_i > 0$  for  $1 \leq i \leq r$ ,  $s_i = 0$  for  $i \geq r + 1$ .

It can be easily shown that orthogonal matrices  $U$  and  $V$  are the eigenvectors of  $AA^T$  and  $A^T A$  respectively.

$$AA^T = US^2U^T$$

and

$$A^T A = VS^2V^T$$

$U$  and  $V$  are called left and right singular vectors. Singular values of  $A$  are the non-zero diagonal elements of  $S$ , which are the nonnegative square roots of the eigenvalues of  $AA^T$ .

Singular value decomposition (SVD) is unique up to certain row, column and sign permutations. If the diagonal elements of  $S$  are constructed to be all positive and ordered in decreasing magnitude with dimension  $(r \times r)$ , then  $U$  has a dimension of  $(m \times r)$ ,  $V$  has a dimension of  $(n \times r)$  and their values are unique. The set  $\{u_i, s_i, v_i\}$  is called the  $i$ -th singular triplet.

The following two theorems demonstrate how the SVD can reveal important information about the structure of a matrix. (Berry, 1995).

*THEOREM 3.1.* Let the SVD of  $A$  be given by 3.3 and

$$s_1 \geq s_2 \cdots \geq s_r > s_{r+1} = \cdots = s_n = 0$$

and let  $R(A)$  and  $N(A)$  denote the range and null space of  $A$ , respectively. Then,

1. rank property:  $rank(A) = r$ ,  $N(A) \equiv span\{v_{r+1}, \dots, v_n\}$ , and

$$R(A) \equiv span\{u_1, \dots, u_r\}, \text{ where } U = [u_1 u_2 \cdots u_m] \text{ and } V = [v_1 v_2 \cdots v_n].$$

2. dyadic decomposition:  $A = \sum_{i=1}^r u_i * s_i * v_i^T$ .
3. norms:  $\|A\|_F^2 = s_1^2 + \cdots + s_r^2$ , and  $\|A\|_2^2 = s_1^2$ .

The rank property, the most valuable aspect of the SVD, allows us to use the singular values of  $A$  as quantitative measures of the qualitative notion of rank. The dyadic decomposition, which is the rationale for data reduction or compression in many

applications, provides a canonical description of a matrix as a sum of  $r$  rank-one matrices of decreasing importance, as measured by the singular values.

*THEOREM 3.2.* (Eckart and Young). Let the SVD of  $A$  be given by 3.3 with  $r = \text{rank}(A) \leq p = \min(m, n)$  and define

$$A_k = \sum u_i \cdot s_i \cdot v_i^T, \quad 3.4$$

then

$$\min_{\text{rank}(B)=k} \|A - B\|_F^2 = \|A - A_k\|_F^2 = s_{k+1}^2 + \dots + s_p^2 \quad 3.5$$

3.5 implies that matrix  $A_k$ , which is constructed from the  $k$ -largest singular triplets of  $A$ , is the closest in the least square sense to  $A$ . Equation 3.5 is the basis for concepts such as data reduction and image compression.

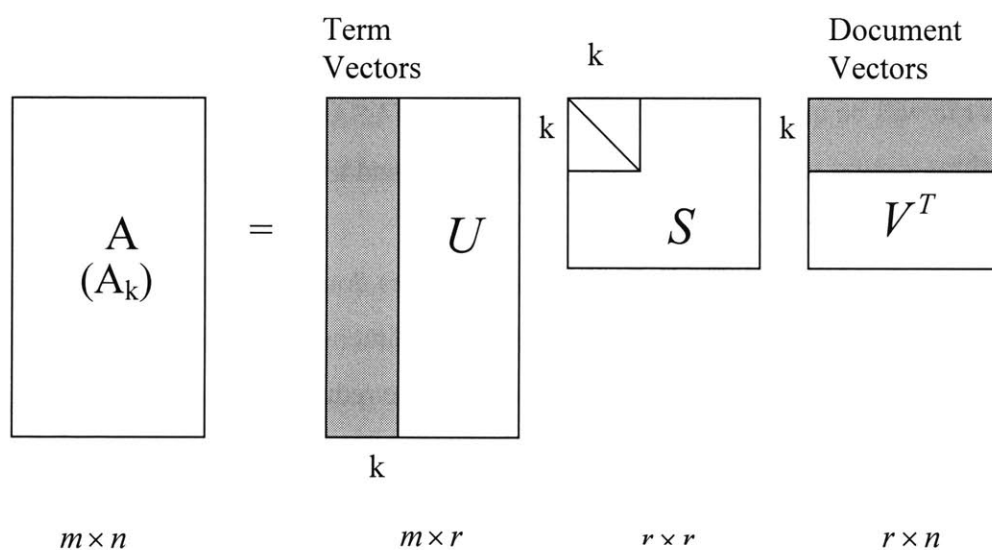
### Dimension Reduction

The SVD reveals important latent semantic structure by decomposing the term-document matrix into a left orthogonal matrix  $U$ , a right orthogonal matrix  $V$ , and a diagonal matrix  $S$ . These matrices reflect a breakdown of the original relationships into linearly independent vectors or factor values.

An important step in LSA is the dimension reduction after SVD. Instead of using the full representation of  $U$ ,  $V$  and  $S$  with  $r$  unique triplets, only the first  $k$  largest singular triplets are maintained to approximate the term-document relationship ( $k \ll r$ ). As we stated earlier, mathematically,  $A_k$  is the closest in the least square error sense rank- $k$  matrix to  $A$ .

Figure 3.2 is a geometric representation of the SVD model.  $U$  and  $V$  are considered the term and document vectors, respectively, and  $S$  represents the singular values. The shaded area in  $U$  and  $V$  and the diagonal line in  $S$  represent the constituents of  $A_k$  in Equation 3.4.

Truncating  $A$  to  $A_k$  is an important procedure, in which the most important latent semantic structure is captured; yet at the same time, the noise and variability of word usage that plague word-based retrieval methods is removed. The number of dimensions  $k$  is usually much smaller than the number of unique words  $M$ . Words are projected into  $k$  dimensional orthogonal space with similar and closely related words being clustered together. Words that occur in similar documents, for example, will be near each other in the  $k$ -dimensional space even if they never co-occur in the same document. This feature further implies that some documents, which may not share any words with a user's query, may still lie near it in  $k$ -space, and thus be retrieved as being relevant.



**Figure 3.2** Geometric representation of the matrix  $A$  and  $A_k$

In vector space representation, after dimension reduction, each word and document is projected into the truncated high dimension space as a vector. Intuitively, in the high dimensional space, each orthogonal dimension may be thought of as an artificial concept; it represents one concept extracted from many different words and documents. As a whole, the space comprises the whole set of semantic concepts embodied the corpus with minor semantics ignored. Each word and document, after the projection into this space, is then characterized by a vector of weights indicating its strength of association with each of these underlying concepts. That is, the “meaning” of a particular term, query, or

document can be expressed by  $k$  factor values, or equivalently, by the location of its vector in the  $k$ -space.

Consider the words *car*, *automobile*, *vehicle*, *traffic*, and *tile*. The word *car*, *automobile*, and *vehicle* are synonyms, *traffic* is a related concept, and *tile* is not related. In most retrieval systems, the query *automobile* is no more likely to retrieve documents about *car* than documents about *tile*, if the precise term *automobile* was not used in the documents. It would be preferable if a query about *automobile* also retrieves articles about *cars* and *vehicles*, or even articles about *traffic* to a lesser extent. The derived  $k$ -dimensional feature space can represent these useful term interrelationships. Roughly speaking, the words *car*, *automobile* and *vehicle* will occur with many of the same words (e.g. motor, Toyota, sedan, engine, truck, automaker, model, etc.), and they will have similar representations in  $k$ -space. The context for *traffic* will overlap to a lesser extent, and those for *tile* will be quite dissimilar. The main idea of LSA is to explicitly model the interrelationships among terms (using the truncated LSA) and to exploit this for information retrieval.

It is important to note that reduced dimensional matrices don't reconstruct the original term-document matrix perfectly because the reduction of dimension eliminates noise and the unreliability of word usages. It is also important that the reduction is not so big that the SVD loses major semantic components. The choice of  $k$  should be large enough to fit all the real structure in the data, but small enough so that we don't also include sampling errors or unimportant details.

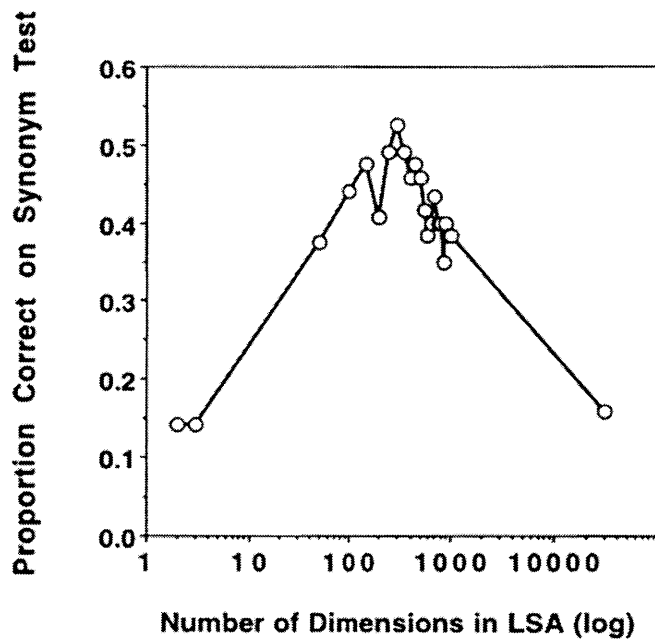
LSA is a Euclidean model which is a linear approximation model. In reality, conceptual relations among words and documents certainly involve more complex structures, including, for example, local hierarchies, and non-linear interactions between meanings. More complex relations can often be approximated better by increasing the number of dimensions.

Since LSA is an information retrieval model, the ability to reproduce the greatest amount of detail in the semantic concept space is not the measure of success, but rather the ability to give the best retrieval effectiveness.

In a study by Landauer et. al. (1998), the ability of LSA to capture synonyms was tested against a standard vocabulary test. LSA was first trained by running SVD analysis



on a large corpus of representative English, and then an eighty-item synonym test was taken from retired versions of the Educational Testing Service (ETS) Test of English as a Foreign Language (TOEFL). To assess the role of dimension reduction, dimensions ranging from 2 to 1032 were used to represent the semantic space. Figure 3.3 plots the success rate of synonyms versus the numbers of dimensions used in LSA. In log-linear coordinates, the TOFEL test results showed a very sharp and highly significant peak. LSA got 52.7% correct with 3000 and 325 dimensions, 13.5% correct with just two or three dimensions and 15.8% correct rate with no dimension reduction at all.



**Figure 3.3** The effect of number of dimensions on performance in an LSA corpus-based representation of meaning in a synonym test (from Landauer, 1998)

A central theme of LSA is that term-term inter-relationships can be automatically modeled and used to improve retrieval. LSA examines the similarity of the “contexts” in which words appear, and creates a reduced-dimension feature-space representation in which words that occur in similar contexts are near each other. That is, the method first

creates a representation that captures the similarity of usage (meaning) of terms and then uses this representation for retrieval.

Figure 3.4 illustrates the effect of LSA on term representations using a geometric interpretation. Traditional vector methods represent documents as linear combinations of orthogonal terms, as shown in the left half of the figure, so that the angle between two documents depends on the frequency with which the same terms occur in both, without regard to any correlations among the terms. Here, Doc 3 contains Term 2, Doc 1 contains Term 1, and Doc 2 contains both terms. In contrast, LSA represents terms as continuous values on each of the  $k$  orthogonal semantic dimensions. As depicted in the right half of Figure 3.4, since the number of factors or dimensions is much smaller than the number of unique terms, terms will not be independent. When two terms are used in similar contexts (documents), they will have similar vectors in the reduced-dimension LSA representation. LSA partially overcomes some of the deficiencies of assuming independence of words, and provides a way of dealing with synonymy automatically without the need for a manually constructed thesaurus.

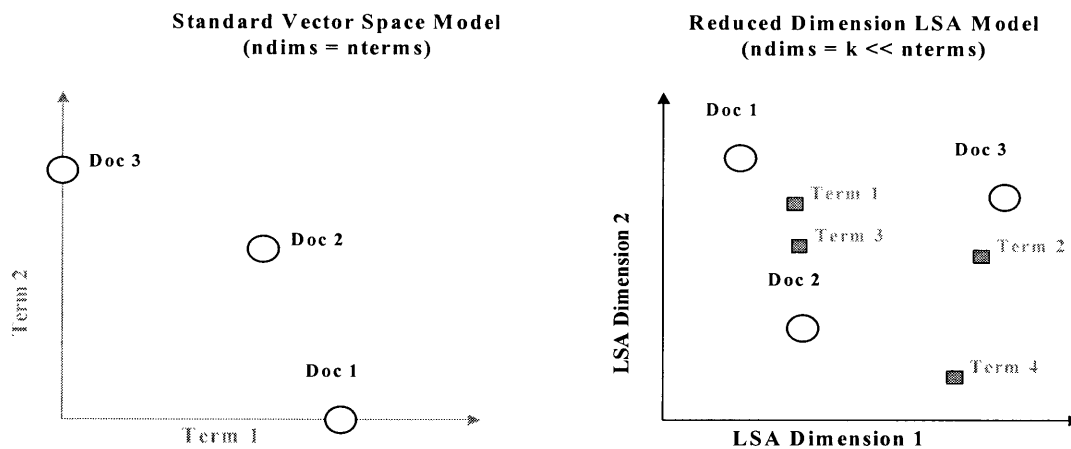


Figure 3.4 Term representations in the standard vs. reduced dimension LSA model

### 3.2.3 Words and Documents Representations and Comparisons

#### Words Representations and Comparisons

LSA clusters words with semantic meaning closer to each other in a reduced high dimensional space. An important question is how to measure the similarity between one word and another. After the SVD dimension reduction of the term-document matrix, each word  $w_i$  (corresponding to a row in the term-document matrix) is now represented by a  $(1 \times r)$  vector in the reduced high dimensional space,  $[U_k S_k]_i$ .  $U_k$  is called term vectors (Figure 3.2)

Adopting the most popular similarity measures from Vector models in IR, the cosine of the angle between two vectors  $w_i$  and  $w_j$  can be calculated as

$$sim(w_i, w_j) = \frac{[U_k S_k]_i [U_k S_k]_j^T}{|[U_k S_k]_i| \times |[U_k S_k]_j|} \quad 3.6$$

The smaller the angle between word pairs  $w_i$  and  $w_j$  in the reduced  $k$  dimensional space computed by SVD, the more similar they are semantically.

When making semantic comparisons between words, the words need to exist in the term list of the term-document matrix.

#### Document Representations and Comparisons

Just like the comparisons between words, comparing two documents from the training corpus measures the cosine of the angles between two column vectors of the matrix  $V_k$ .  $V_k$  is called document vectors (Figure 3.2). Rows of  $V_k S_k$  are the coordinates for documents from the training corpus.

$$sim(d_i, d_j) = \frac{[V_k S_k]_i [V_k S_k]_j^T}{|[V_k S_k]_i| \times |[V_k S_k]_j|} \quad 3.7$$

The angle between documents  $d_i$  and  $d_j$  is measured by Equation 3.7. Just as in the comparisons between two words, the smaller the angle between document pairs  $d_i$  and  $d_j$ , the more similar they are semantically.

### New Query Representation

When retrieving information, the user's query (usually a document) must be represented as a vector in  $k$ -dimensional space and compared to other documents. The previous results show how to compare the semantic similarities between words and documents from the training corpus. It is also very important that documents that don't appear in the training corpus can be compared to those that do. A query (like a document) is a set of words. After the same application of document processing when generating term-document matrix, the query vector can be represented as,

$$\hat{q} = q^T U_k S_k^{-1} \quad 3.8$$

$\hat{q}$  is obtained by first projecting pre-processed query vector to the orthogonal semantic space  $U_k$  (which gives  $q^T U_k$ ), then applying column reweighting by  $S_k$ . The query vector is the weighted sum of its constituent term vectors. The query vector can then be compared to all existing document vectors, and the similarity ranking can be made by comparing the cosine of the vector angles.

#### 3.2.4 Singular Value Decomposition of Sparse Matrices

Computationally, the singular value decomposition of sparse matrices is the most costly operation in LSA. In this section, we review the methods that are best suited for sparse matrix SVD and their respective computational cost.

The term-document matrix is a sparse matrix. The classical methods for determining the SVD of dense matrices (the Golub-Kahan-Reinsch method and Jacobi-like SVD methods) are not optimal for large sparse matrix SVD. Since these methods apply orthogonal transformations (Householder or Givens) directly to the sparse matrix. As a result, they incur excessive fill-in and thereby require tremendous amounts of memory. In

addition, these methods compute all the singular triplets, which is computationally wasteful when our interest lies in getting the few largest singular triplets.

Before we review briefly the methods of sparse matrix SVD, we first look at the two canonical symmetric equivalent problems to an asymmetric matrix SVD. The term-document matrix is asymmetric. Its canonical symmetric equivalence is usually sought to compute the sparse SVD.

### Equivalent Eigenvalue Problems

Assume a sparse matrix  $A$  with dimensions of  $m \times n$  ( $m \gg n$ ) has rank of  $r$ . A symmetric matrix  $B$  can be defined as

$$B = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \quad 3.9$$

It can be shown that the eigenvalues of  $B$  are the  $2r$  pairs,  $\pm s_i$ , where  $s_i$  is a singular value of  $A$ . The following Lemma demonstrates how to derive the SVD of  $A$  from the eigen pairs of the matrix  $B$ .

LEMMA 3.1. Let  $A$  be an  $m \times n$  ( $m \gg n$ ) matrix and  $B$  defined by Equation 3.9.

1. For any positive eigenvalues,  $s_i$  of  $B$ , let  $(u_i, v_i)^T$  denote a corresponding eigen vector of norm  $\sqrt{2}$ .  $s_i$  is a singular value of  $A$  and  $u_i, v_i$  are respectively, left and right singular vectors of  $A$  corresponding to  $s_i$ .
2. For  $s_i = 0$ , if  $B$  has corresponding orthogonal eigen vectors  $(u_j, v_j)^T$  with  $v_j \neq 0$  and  $u_j \neq 0$  for  $j = 1, \dots, t$  for some  $t \geq 1$ , then zero is a singular value of the matrix  $A$ , and the corresponding left and right singular vectors can be obtained by orthogonalizing these  $u_j$  and  $v_j$ , respectively. Otherwise,  $A$  has full rank ( $rank(A) = n$ ).

An alternative solution is to compute the eigen pairs of either the  $m \times m$  matrix  $AA^T$  or the  $n \times n$  matrix  $A^T A$ . The following lemma indicates the relationships between these symmetric eigenvalue problems and that of their asymmetric form.

LEMMA 3.2. Let  $A$  be a  $m \times n$  ( $m \gg n$ ) matrix with  $rank(A) = r$ .

1. If  $V = \{v_1, v_2, \dots, v_r\}$  are linearly independent  $n \times 1$  eigenvectors of  $A^T A$  so that  $V^T (A^T A)V = diag(s_1^2, s_2^2, \dots, s_r^2)$ , then  $s_i$  is the  $i$ -th nonzero singular value of  $A$  corresponding to the right singular vector  $v_i$ . The corresponding left singular vector,  $u_i$ , is then derived by  $u_i = \frac{1}{s_i} Av_i$ .
2. If  $U = \{u_1, u_2, \dots, u_r\}$  are linearly independent  $m \times 1$  eigenvectors of  $AA^T$  so that  $U^T (AA^T)U = diag(s_1^2, s_2^2, \dots, s_r^2)$ , then  $s_i$  is the  $i$ -th nonzero singular value of  $A$  corresponding to the left singular vector  $u_i$ . The corresponding right singular vector,  $v_i$ , is then derived by  $v_i = \frac{1}{s_i} A^T u_i$ .

### **Sparse Matrix SVD Methods**

Berry (1992) presented four methods for solving equivalent sparse symmetric eigenvalue problems, including two Lanczos-based methods: Single-Vector Lanczos (LASVD) and Block Lanczos (BLSVD); and two subspace methods: Subspace Iteration (SISVD) and Trace Minimization (TRSVD). All four methods are mathematically quite complicated and have been extensively described by Berry in his paper.

In his paper (Berry 1992), Berry also presented fundamental comparisons of all four methods in model complexity, memory requirements, parallelism, and parameter selections. With regard to speed of computation (in CPU time), Berry indicated that the Single-Vector Lanczos method (LASVD) is the fastest method for approximating several of the largest singular triplets with low or moderate accuracy.

### **Computational Cost of SVD**

The primary cost of all the algorithms proposed by Berry (1992) lies in the total number of sparse matrix-vector manipulations required. If we denote  $\Delta_A$  as the density of  $A$ , which is defined as the total number of nonzero cells in  $A$  divided by the total number of cells ( $m \times n$ ), then the total cost in floating point operations per iteration is given by (Berry, 1992),

$$N_{SVD} = R[2(1 + \Delta_A n)m + 2(1 + \Delta_A m)n]$$

In a typical scenario,  $\Delta_A$  is in the range of [0.25%, 0.5%] (Dumais, 1997), and the value of  $R$  is between 100-300. this expression can, therefore, be approximated by

$$N_{SVD} \approx (4R\Delta_A)mn \approx mn$$

For the values of  $m$  and  $n$  typically in the order of 10,000 and 100,000,  $N_{SVD}$  measures up to a few billion floating-point operations (flops) per iteration. On a desktop machine (such as a 800-MHz PC, rated at approximately 120 Mflops), this translates into (up to) a few minutes of CPU time. As convergence is typically achieved after 100 or so iterations, the entire decomposition is usually completed within a matter of hours.

### 3.3 Applications

LSA was first developed for information retrieval purposes. Standard evaluations have proved its effectiveness over other retrieval methods (Funas et. al. 1998). Because LSA is a completely automatic method, it has been applied to a wide range of problems.

In LSA, queries can be either terms, documents, or combinations of the two (as in relevance feedback). The returned objects can also be terms and documents. Returning nearby terms is useful for some applications like online thesauri, or for suggesting index terms for documents.

In LSA, the term-document matrix is used to retrieve terms and documents. The same concept can be applied to any descriptor-object matrix. We typically use only single terms to describe a document, but phrases or n-grams could also be included as rows in the matrix. Similarly, an entire document is usually the text object of interest, but smaller,

more topically coherent units of text (e.g. paragraphs, sections) could be represented as well.

In the following, we first review the application of LSA in information retrieval and information filtering. In information retrieval application, we emphasize the evaluation of LSA over standard test collections developed in the information retrieval community to showcase the effectiveness of LSA as a novel information retrieval method. We then review a few innovative applications of LSA to showcase its broad impact beyond information retrieval.

### **Information Retrieval**

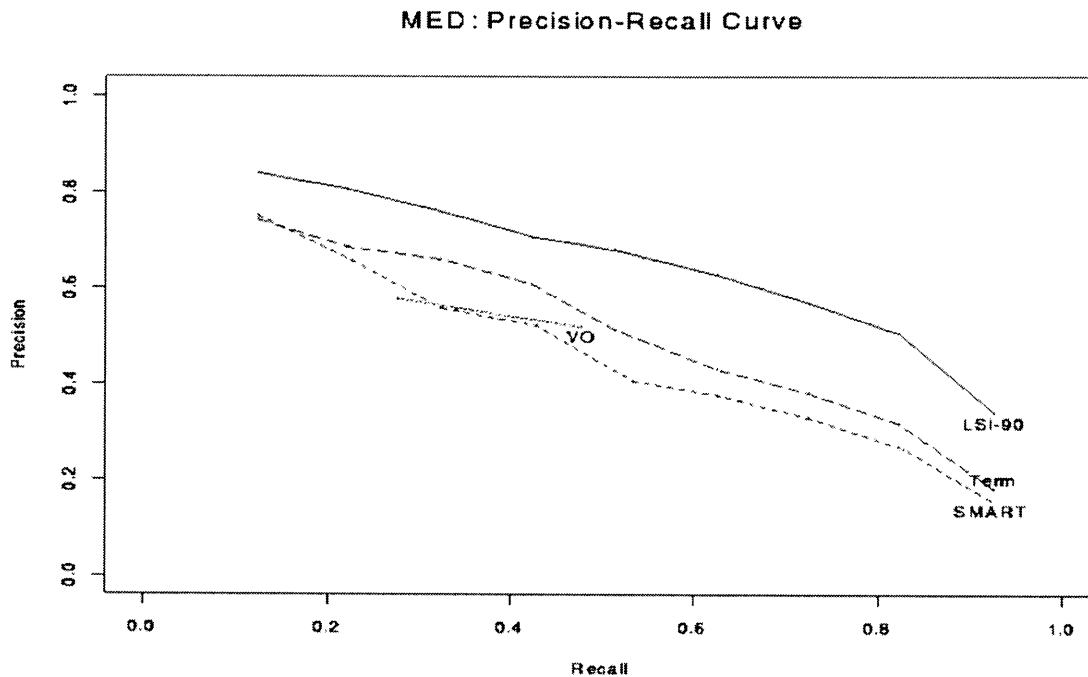
Information retrieval applications are characterized by relatively stable databases, but rapidly changing ad hoc user queries. A number of test collections developed by information retrieval communities are usually used to evaluate the effectiveness of the retrieval system. In these test collections, a set of user queries is calibrated, and their relevance judgments (i.e. for each query, every document in the collection has been judged as relevant or not to the query) are available. Thus, the standard evaluation criteria, *precision* and *recall*, can be measured in these test collections for each retrieval system. Average *precision* across several levels of *recall* can then be used as a summary measure of performance.

LSA has been evaluated against several information science test collections. The average *precision* using LSA ranged from comparable to 30% better than that obtained using standard keyword vector methods (Dumais, 1991; Furnas 1988). The LSA method performs best relative to standard vector methods when the queries and relevant documents do not share many words, and at high levels of *recall*.

Figure 3.5 shows the *precision-recall* curves for TERM matching, a 90-factor ( $k = 90$ ) LSA, SMART, and Voorhees systems on the MED dataset. MED is a commonly studied collection of medical abstracts, which consists of 1033 documents and 30 calibrated queries. TERM represents a straight forward term matching method. SMART is a standard keyword vector method (Salton 1968). The Voorhees data were obtained directly from her paper in which she used a vector retrieval system with



extended Boolean queries (Voorhees, 1985). SMART and Voorhees methods are representative of state of the art information retrieval systems.



**Figure 3.5** Precision-recall curve for TERM matching, a 90-factor LSA, SMART, and Voorhee systems on MED dataset (Dumais, 1997)

As we pointed out earlier, one of the common and usually effective methods for improving retrieval performance in vector methods is to transform the raw frequency of occurrence of a term in a document by some function (see Chapter 1, section 1.4.2). Such transformations normally have two components. Each term is assigned a global weighting, indicating its overall importance in the collection as an index term. The same global weighting is applied to an entire row (term) of the term-document matrix. It is also possible to transform the term's frequency in the document; such a transformation is called a local weighing, and is applied to each cell in the matrix.

Harman (1986) studied the IRX (Information Retrieval Experiment) at the National Library of Medicine using a standard vector retrieval method. Harman found that both *Idf* and Entropy weighting produced large performance advantages. Harman found the following advantages over her baseline term overlap measures: *Idf* 20%; Entropy 24%, LogEntropy 35%; LogEntropy/normalized-document-length 44%.

Dumais studied the effect of term weighting on the performance of LSA (Dumais, 1991). Dumais found Idf and Entropy global term weighting improved performance by an average of 30%, and improvements with the combination of a local Log and global Entropy weighting were 40%.

### **Information Filtering**

Information filtering is a problem closely related to information retrieval. Information filtering is characterized by relatively stable information needs, but a rapidly changing data. Information filtering is also known as information routing; selective dissemination of information; electronic clipping services; and personalized information delivery.

Applying LSA to information filtering is straightforward. Initial samples of documents are analyzed using LSA/SVD tools to build a reduced-dimension semantic space. A user's interest is then represented as a vector in this space. Each new incoming document is projected into this space and then compared to the vector representing user's interests. Similar documents are routed to the user. Relevance feedback is used to improve the representation of user's interest over time.

Foltz (1991) studied the effectiveness of using LSA for information filtering. In their study, a semantic LSA space is built for a set of articles that have previously been judged by a user to be interesting or not. To determine if a new article is relevant, it is projected into ("folded in", as used in LSA literatures) the semantic spaces on the basis of contained terms. If the article appears close to other interesting articles in the space, then it is considered likely to be interesting to the user. Otherwise, if the article appears closer to other non-interesting articles, it is considered not interesting to the user.

The study indicated LSA improved prediction performance over keyword matching an average of 13% and showed a 26% improvement in precision over presenting articles in the order received. The results indicate that user's preferences for articles tend to cluster based on the semantic similarities between articles.

Foltz and Dumais (1992) studies different methods to predict which technical memos best match people's technical interests. Two methods are used to describe technical interests, one based on sets of keywords that the testers provided, and the other using feedback about previous memos they found relevant. Two information retrieval methods

were tested to make the predictions, one using standard keyword matching, and the other using LSA. All four methods effectively selected relevant memos. In this study, the best method for filtering was LSA with feedback about previous relevant memos.

### **Cross Language Retrieval**

It is important to note that LSA makes no assumptions about English syntax or semantics. Words are tokenized by delimitation of spaces and punctuation. It is also important to note that in LSA applications, no stemming has been used to collapse words with the same morphology. If the words with the same stem are used in similar documents, they will be represented as vectors closer to each other in the reduced dimensional semantic space; if they don't appear in similar documents, they will not be judged to lie semantically close. These characteristics of LSA methodology imply LSA is applicable to any language. It can also be applied to cross-language retrieval if there exists a common LSA space in which words in different languages with similar semantic meanings are close to each other.

Landauer and Litterman (1990) described one method for creating such a LSA space. In their study, an initial sample of documents is translated by humans, or perhaps, by machine to create a set of dual-language training documents. These same documents, but in different languages, are then merged into a single document. The LSI method ignores word order, therefore, treats the merged document as a bag of freely intermingled words in multi-languages. The collection of documents with each document including multiple versions in different languages (French and English in their experiment) is used to form the term-document matrix. The same words, but in different languages, are treated as independent terms. The resulting reduced dimension semantic space is then a space which contains vectors of both English and French words and combined documents. Words that are consistently paired in translation will be given identical representations in LSA space, whereas words that are frequently associated with one another will be given similar representations.

The next step in their method is to add (or "fold in") documents in just English or French. The same process is adopted for representing the English or French documents, in which each document is a weighted vector sum of its constituent terms. The result of

this process is that each document in the database, whether it is in French or English, has a language-independent representation in terms of numerical vectors. Users can now pose queries in either English or French and get back the most similar documents regardless of language.

Experimental studies showed that the completely automatic multilingual space was more effective than single language space. The retrieval of French documents in response to English queries (and vice versa) was as effective as first translating the queries into French and searching a French only database.

An extension to three languages (English, French, and German) of cross-language information retrieval is performed by Rehder et. al. (1997) with larger document collections and much noisier training data.

### **People Matching**

LSA has also been used to match people instead of documents. An expert locating system, known as Bellcore Advisor, was developed by Streeter and Lochbaum (1988) to find local experts relevant to users' queries. A user's query or information was matched to the nearest documents and project descriptions, and the author's organization was returned as the most relevant group or organization.

In another applications (Dumais and Nielsen, 1992), LSA is used to automate the assignments of submitted conference papers to reviewers. In this application, a semantic space is first built by analyzing the available textual databases, which contain the relevant domain knowledge. Submitted papers, which are represented by their titles and abstracts, are projected into the semantic space as vectors. Abstracts from the past work of the reviewers are used to represent the reviewers' expertise. Reviewers (twenty five in this study) are automatically represented as points in the  $k$ -dimensional LSA space. Hundreds of submitted papers were then matched to the closest reviewers. These LSA similarities along with additional constraints to ensure that each paper was reviewed  $p$  times and that each reviewer received no more than  $q$  papers to review were used to assign papers to reviewers for a major human computer interaction conference. Study indicated that these completely automatic assignments (which took roughly twenty minutes) were as good as those of human experts.

## Noisy Input

Because LSA does not depend on lexical word matching, it is especially useful when the input text is *noisy*, as in OCR (optical character recognition), open input, or documents with spelling errors. If there are scanning errors and a word (*polymorphism*) is misspelled (as *polymophism*), many of the other words in the document are spelled correctly. If these correctly spelled context words also occur in documents that contain a correctly spelled version of *polymorphism*, then *polymophism* probably will be near *polymorphism* in the k-dimensional space.

Nielsen et. al. (1994) used LSA to index a small collection of abstracts input by a commercially available pen machine in its standard recognizer mode. Even though the error rates were 8.8% at the word level, information retrieval performance using LSA was not disrupted (compared with the same uncorrupted text). Kukich used LSA for a related problem, spelling correction. In this application, the rows were unigrams and bigrams and the columns were correctly spelled words. An input word (correctly or incorrectly spelled) was broken down into its bigrams and trigrams, the query vector was located at the weighted vector sum of these elements, and the nearest word in LSA space was returned as the suggested correct spelling.

## 3.4 LSA - Evaluation

In this section, we present our evaluation of LSA against a text corpus of three years of Wall Street Journal articles. The text corpus in this study is not a calibrated text collection (in the sense that no relevance judgments available), thus it is impossible to measure the precision and recall performance of the LSA retrieval system. The intention of the evaluation in this section, though, is not to compare the effectiveness of LSA with other retrieval models (we believe LSA is effective based on the study reviewed in section 3.3), but rather to evaluate:

1. the tools developed in this study, which are used for different purposes, in different stages of latent semantic analysis.
2. computation cost of LSA, including SVD, and the word/document retrieval cost.
3. sample retrieval results and overall effectiveness of the LSA system.

### **3.4.1 Software Tools**

#### **CMU-Cambridge Statistical Language Modeling Toolkit**

CMU-Cambridge statistical language modeling toolkit is a set of Unix software tools designed to facilitate language modeling work in the research community. It is a popular toolkit widely used in academic, government, and industrial laboratories all over the world (Clarkson and Rosenfeld). It includes tools to process general text data, such as generating word frequency lists and vocabularies; word bigram and trigram counts; bigram and trigram related statistics; various backoff bigram and trigram language models etc. It also includes tools to use the resultant language models to compute such quantities as perplexity; out-of-vocabulary (OOV) rate; distribution of backoff cases etc.

In this study, only the first two tools are used, namely the tools to generate the word frequency list and tools to generate the vocabulary.

#### **SVDPACKC**

SVDPACKC (Berry et. al. 1996) is a software package written in ANSI C, which includes the implementations of the four algorithms proposed by Berry (1992) to compute the singular valued decomposition (SVD) of large sparse matrices. Normally, only the largest 100 to 300 singular triplets (singular values and corresponding left and right singular vectors) are determined in these algorithms.

Each algorithm (Single-Vector Lanczos; Block Lanczos; Subspace Iteration; and Trace Minimization) is applied to both canonical symmetric matrices. Thus the package includes eight stand alone C programs to calculate the sparse matrix SVD.

The following table presents the naming convention for all the standalone C programs.

MMTE[.c]		
Field	Description	Possible Entries
MM	Method (Algorithm)	bl ≡ Block Lanczos la ≡ Single Vector Lanczos si ≡ Subspace Iteration tm ≡ Trace Minimization
T	File Type	d ≡ Document File p ≡ Input Parameters File o ≡ Output File s ≡ Source File
E	Eigensystem Or Output Channel from SVDPACK (Fortran-77)	1 ≡ Cyclic Matrix $B$ defined by (3.9) 2 ≡ $A^T A$ Matrix (for comparison purposes) 2,3,8,9 ≡ Output Channel

**Table 3.1** SVDPACKC program naming convention

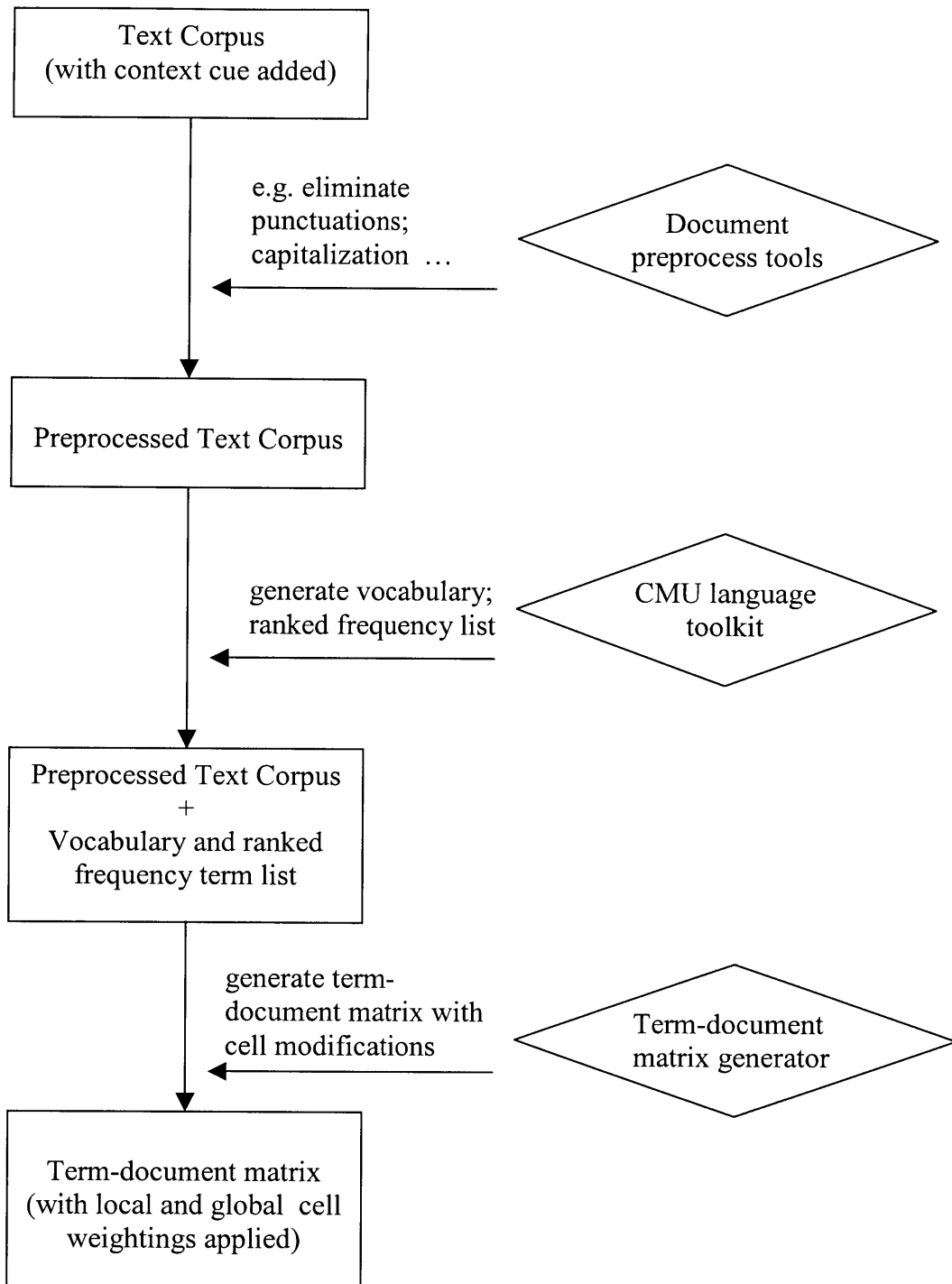
For example, las2 is the Single Vector Lanczos algorithm with the symmetric matrix of  $A^T A$ .

As indicated by Berry (1992), the Single-Vector Lanczos method is the fastest method for approximating several of the largest singular triplets for low or moderate accuracy. Comparing the effect of the choice of symmetric matrix ( $E = 1$  or  $2$ ), when ill-conditioning is not likely, the use of las2 is recommended because of smaller eigensystems and less memory requirements. Otherwise las1 should be used which is a root-free method and approximates  $+/-$  pairs of each singular value of  $A$ .

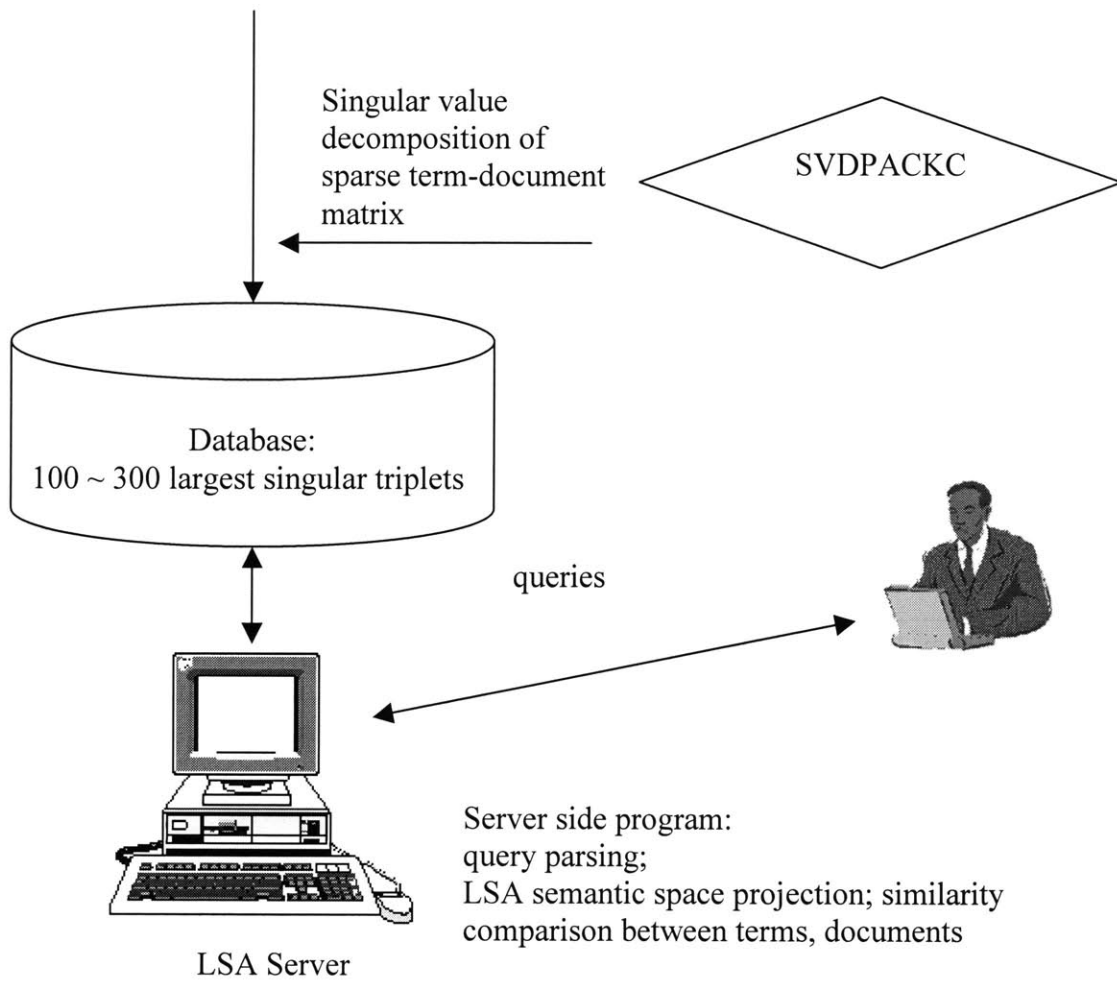
In this study, we use las1 as our choice of SVD algorithm for the term-document matrix.

### 3.4.2 LSA Flow Chart

Figure 3.6 is a flow chart showing the intermediate functions and software in each step towards building a LSA database retrieval system.







**Figure 3.6** LSA flow chart

### 3.4.3 The Corpus

We use a text corpus of three years of Wall Street Journal articles (1988-1990) in this section to evaluate the effectiveness of LSA (It is the same text corpus we used in Chapter two to evaluate work entropy). The text corpus we use in this evaluation have the following statistics.

<b>Number of documents</b>	150,981
<b>Average document length (in words)</b>	245
<b>Total number of word occurrences</b>	36,920,947
<b>Number of unique words (vocabulary)</b>	164,799
<b>Number of unique words that appear more than once</b>	107,579
<b>Number of unique words that appear more than twice</b>	86426
<b>Number of unique words that appear more than five times</b>	60316
<b>Number of unique words that appear more than seven times</b>	52403

**Table 3.3** Statistics of the three years of Wall Street Journal article collections.

The term list we use to generate the term-document matrix is the top ranked **50,000** words, which essentially eliminate all the vocabulary of less than seven total occurrences in the text corpus.

The document list we use to generate the term-document matrix is the first 150,000 documents. Thus the term-document matrix is a 50,000 by 150,000 matrix. The number of non-zero cells is 17,779,714, which makes the sparsity of the term-document matrix 0.24%.

### 3.4.4 Computation Cost

As reviewed earlier, SVD of the sparse term-document matrix is the major component of total computational time. The subsequent term/document retrieval cost is significantly lower. Table 3.4 is the cost estimate of the three major LSA components

Components	Computation time	BigO estimation
SVD	$4R\Delta_A mn$	$O(m \times n)$
Term retrieval	$lRm$	$O(m)$
Document retrieval	$lRn$	$O(n)$

Where:

$R$ :	number of singular values and vectors in SVD	$\approx 100 \sim 300$
$\Delta_A$ :	sparsity of the term-document matrix	$\approx 0.25\% \sim 0.5\%$
$l$ :	number of similar terms/documents retrieved for each query	$\approx 20$
$m$ :	number of terms	in the order of 10,000
$n$ :	number of documents	in the order of 100,000

**Table 3.4** Computation cost of LSA

The following table summarizes the SVD statistics after a successful run using las2 (Single Vector Lanczos method on  $A^T A$ ) against the 50,000 X 150,000 term-document matrix on a Pentium 4 PC with 2.80 GHz CPU and 2.0GB RAM. Table 3.5 indicates that the total of 321 eigenpairs are derived in a matter of an hour on a very powerful PC. All other algorithms, BLSVD, SISVD, TRSVD, take two up to five time more CPU than LASVD when  $A^T A$  is solved. When the  $B$  matrix (Equation 3.9) is used for SVD analysis, the memory requirements are much bigger (from 3 to 5 times) compared to those of the  $A^T A$  matrix. In this study, we did not test the alternative method.

Number of Terms (rows)	50,000
Number of Documents (cols)	150,000
Order of matrix $A^T A$	50,000 X 50,000
Max. No. of Lanczos steps	800
Max. No. of eigenpairs	800
Allocated memory	1.82MB
CPU time	50 minutes
Number of eigenpairs solved within accuracy convergence requirements	321

**Table 3.5** Statistics of the 50,000 by 150,000 term-document matrix SVD

### 3.4.5 Effectiveness of Retrieval

In this section, we present a few sample retrieval results of both terms and documents in tabular forms. From the observation of these results, we can get a much deeper understanding of the power of LSA in retrieving terms and documents that are semantically close.

#### Word Retrieval

In word retrieval, we evaluate the power of LSA by investigating the retrieval results of several words that are semantically very distant. In this way, we can clearly see the clustering power of LSA – the power that can bring similar words together, but separate semantically different words far apart.

In this section, we pick words in three different semantic concept spaces, namely financial, medical and political concept.

In the following, we present the retrieval results of the words. For each query word, we list the top 15 words that are semantically most close from the vocabulary list (50,000 in this case) with the semantic angles between them listed at the side.

Before we show the retrieval results of nouns, Table 3.6 and Table 3.7 are the LSA retrievals of two adjectives, *rise* and *increase*. In table 3.6, the query term is *rise*. The retrieval list includes both synonyms and antonyms of *rise* as well as words semantically close to *rise*. It also includes *risen* as the 5<sup>th</sup> most semantically close word. Earlier in section 3.2.1, we made a case for not using stemming in LSA as we believe LSA provides the capability of clustering words with same stems if they appear commonly in same contexts. LSA clearly clusters *rise* and *risen* very close to each other in Table 3.6. Table 3.7 also indicates LSA clusters *increase* and *increases* together.

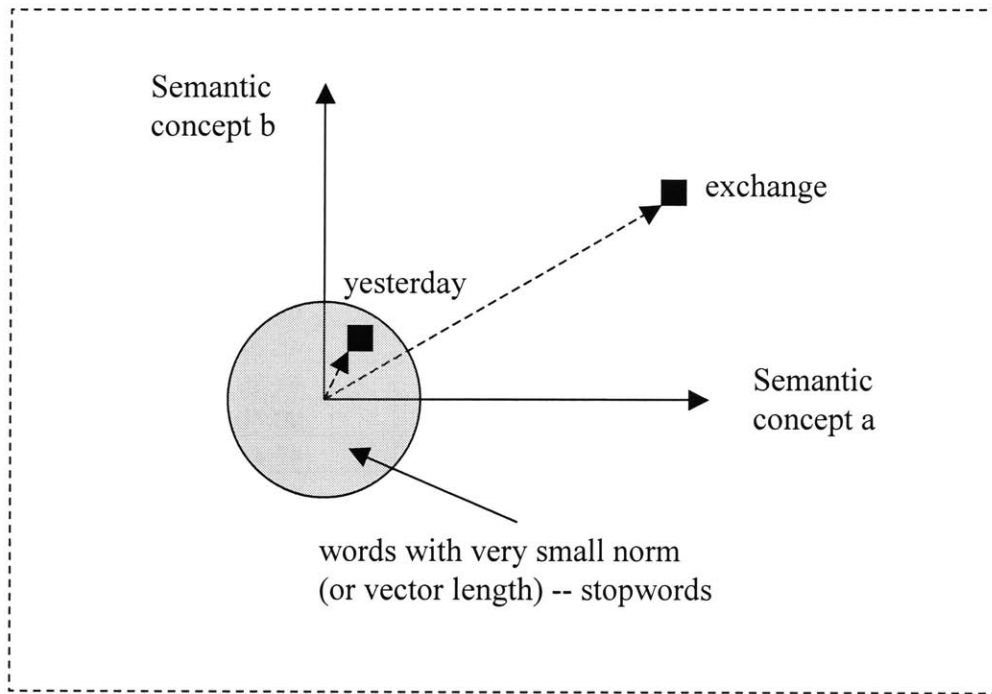
query word	Word	Semantic angle <sup>(o)</sup>
<b>rise</b>	decline	21.15
	drop	27.08
	surge	30.56
	accelerated	30.96
	<b>risen</b>	31.03
	slowed	31.89
	pace	33.15
	growth	33.57
	Inflation	34.34
	overall	34.90
	fueled	35.19
	comparison	35.21
	gross	35.69
	jump	36.01
<b>increase</b>	36.22	

**Table 3.6** LSA query result of *rise*

query word	Word	Semantic angle <sup>(o)</sup>
<b>increase</b>	decrease	31.36
	<b>rise</b>	36.22
	reduced	38.53
	boost	38.86
	drop	39.13
	growth	39.23
	overall	39.90
	decline	40.57
	<b>increases</b>	40.62
	reflect	40.75
	excluding	40.82
	offset	42.48
	yearly	43.56
	forecast	44.51
	gross	44.62

**Table 3.7** LSA query result of *increase*

Table 3.8 and Table 3.9 show the retrieval results of two words from financial area, *stock* and *exchange*. Most retrieved words are closely related to the query words in both retrievals. However it also contains words which are not related. For example in Table 3.9, *yesterday* is retrieved as highly related word to *exchange*. The normalized entropy of *yesterday* measures 0.812, which is very high, indicating it is a frequent and common word (thus possibly a stopword in this document corpus). Remember we apply  $1 -$  normalized word entropy as the global word weight, thus *yesterday* will appear in the LSA reduced space as a vector with very small norm (vector length).



**Figure 3.7** LSA measures of stopwords with others

Figure 3.7 indicates the small zone close to the origin as the zone where stopwords will gather in the reduced LSA space. In general, even if a stopword is close to a low entropy word in a corpus, the relation has little significance to us because stopwords carry so little semantic weight.

Since we use angles between vectors to measure the semantic similarities, it is likely that a stopword is “close” to a query word if their vectors have very small angles between them.

In table 3.8 and 3.9, there exist a few possessive words (i.e. words ending with 's). They are negligence in document preprocessing in which case all punctuations should be removed, including “'”. Since they are left unprocessed, the possessive forms are treated as independent words from their root words with. None-the-less, LSA result shows that if these words appear in common context with the query words, they are labeled as semantically close words. In Table 3.9, *exchange's* is retrieved as semantically close word as *exchange*.

query word	Word	Semantic angle <sup>(o)</sup>
<b>stock</b>	plummeting	38.09
	delisted	42.24
	<b>exchange</b>	43.35
	<b>empire's</b>	44.92
	composite	44.94
	mercury's	45.50
	options	46.32
	oneida	46.39
	basket	46.70
	trading	47.44
	<b>where's</b>	48.11
	halted	48.35
	penny	48.71
	splits	48.81
	unremarkable	49.12

Table 3.8 LSA query result of *stock*

query word	Word	Semantic angle <sup>(o)</sup>
<b>exchange</b>	composite	41.71
	stock	43.35
	delisted	43.90
	<b>mercury's</b>	44.21
	trading	44.86
	disciplined	48.86
	amex	49.21
	<b>empire's</b>	49.52
	closed	50.48
	unremarkable	51.16
	<b>yesterday</b>	51.81
	colbert	51.85
	unchanged	51.89
	<b>exchange's</b>	52.28
	options	52.46

Table 3.9 LSA query result of *exchange*



Table 3.10 and Table 3.11 are words from medical field, *drug* and *prozac*. The observation indicates that they are clustered much more close by than words from the financial area (by comparing semantic angles).

query word	Word	Semantic angle <sup>(o)</sup>
<b>Drug</b>	depressant	11.32
	prozac	11.55
	psychotic	11.70
	unemployable	11.79
	deters	12.04
	ulcer	12.10
	investigational	12.65
	relieving	12.81
	hypertension	13.20
	gastrointestinal	14.01
	lymphadenopathy	14.07
	mylan	14.21
	prolongs	14.46
	acceptability	14.52
	addicts	14.63

**Table 3.10** LSA query result of *drug*

query word	Word	Semantic angle <sup>(o)</sup>
<b>prozac</b>	depressant	6.63
	drug	11.55
	ulcer	14.40
	unemployable	16.07
	hypertension	16.17
	acceptability	17.01
	gastrointestinal	17.19
	relieving	17.23
	deters	17.35
	mylan	17.59
	baldness	17.69
	investigational	17.86
	psychotic	18.05
	lymphadenopathy	18.25
	lovastatin	18.82

**Table 3.11** LSA query result of *prozac*

When we made the case of no stemming, we used an example of *doctor*, *doctors* and *doctoral*. We hypothesized that the first two words, *doctor* and *doctors* are mostly likely to appear in semantically close context, but the word *doctoral* would mostly be used in different context. Table 3.12 to 3.13 show us just that. LSA retrievals of query words *doctor* and *doctors* are mostly related to the medical world, but the retrieval of *doctoral* indicates that it is mostly used in the context of the education field, e.g. this is a *doctoral* dissertation.

query word	Word	Semantic angle <sup>(o)</sup>
<b>doctor</b>	patient	35.39
	dani	36.52
	bolognesi	37.39
	curable	39.42
	relman	39.83
	zagury	40.43
	disease	40.85
	pizzo	40.87
	physician	40.94
	infection	40.94
	acetylcholine	41.14
	doctors	41.45
	therapy	41.69
	alzheimer's	41.81
	sentries	41.83

Table 3.12 LSA query result of *doctor*

query word	Word	Semantic angle <sup>(o)</sup>
<b>doctors</b>	patient	21.01
	physician	25.88
	physician's	25.99
	illnesses	27.52
	doctors'	28.89
	relman	28.97
	infants	30.00
	expectancy	30.99
	obstetrics	31.70
	tuberculosis	32.35
	surgeries	32.48
	clinics	32.93
	disease	33.27
	inpatient	33.61
	therapy	33.75

Table 3.13 LSA query result of *doctors*

query word	Word	Semantic angle <sup>(o)</sup>
<b>doctoral</b>	author	50.80
	graduates	51.66
	dimes	51.90
	lecturer	52.18
	tenured	52.33
	bednorz	53.01
	accomplishment	53.29
	housewife	53.43
	parodied	53.48
	suffocating	53.49
	black	53.56
	erudite	53.64
	humanities	53.64
	polio	53.65
	chaucer	53.67

**Table 3.14** LSA query result of *doctoral*

Table 3.15 and Table 3.16 are words from political field, *bush* and *administration*. We can clearly see the clustering power of LSA.

query word	Word	Semantic angle <sup>(o)</sup>
<b>bush</b>	nonlethal	25.47
	administration's	27.69
	Veto	29.21
	aides	29.28
	confrontations	29.67
	byrd	31.63
	house	33.06
	fitzwater	33.14
	marlin	34.15
	soars	34.46
	democrats	34.76
	initiative	35.02
	senate	35.26
	lawmakers	36.00
	dole	36.45

**Table 3.15** LSA query result of *bush*

query word	Word	Semantic angle <sup>(o)</sup>
<b>administration</b>	lawmakers'	24.25
	administration's	29.57
	interdiction	30.56
	narcotics	32.64
	anti	32.67
	institutionalize	34.98
	offensive	35.86
	antidrug	35.90
	kingpins	35.98
	initiative	36.24
	recast	36.58
	contemplates	36.60
	aid	36.81
	ceaseless	37.05
	trafficking	37.30

**Table 3.16** LSA query result of *administration*

### Document Retrieval

The document retrieval results clearly demonstrate the power of LSA to retrieve semantically similar documents to user queries. We list some sample results in the appendix.

From the study of the retrieval results, we can make following observations:

- The top ranked retrieved documents are semantically very similar to the query document.
- When retrieved results are ranked by semantic angles between query document and retrieved documents, we see clearly patterns of the direct correlation between semantic closeness and ranked order. The higher the ranked order the retrieved document is, the semantic much closer it is to the query document.
- Top ranked documents may not contain common words in the query documents, thus they are being retrieved as relevant documents.

### 3.5 Summary

In Chapter Four, we reviewed and evaluated Latent Semantic Analysis. Two well-known language phenomena which plague the information retrieval performance of lexical word matching are synonymy and polysemy. Synonymy means that an object can be referred in many ways, i.e. people use different words to represent the same semantic subject. Polysemy is the problem of word having more than one specific meaning. LSA offers a dampening effect on synonyms, though the effect on polysemy is less pronounced.

LSA assumes there exist implicit higher semantic structure in term-document associations. Terms tend to be similar if they appear in the same kind of documents, whether or not they actually occur within identical word contexts in those documents. Documents are semantically close if they have many similar words in common, and semantically distant if they have few words in common.

LSA starts with the construction of term-document matrix, with each cell represents the co-occurrences between words and documents. In this section, we reviewed the several text operations which are normally performed in IR, such as lexical analysis of text, treating stopwords, stemming and selection of terms for term-document matrix. We presented the global and local weighting strategies for each cell of term-document matrix.

The central component of LSA is the singular value decomposition of the term-document matrix. The SVD reveals important latent semantic structure by decomposing the term-document matrix into three sets of matrices, a left orthogonal matrix, a right orthogonal matrix and a diagonal matrix. An important step in LSA is the dimension reduction after SVD. By dimension reduction, LSA is able to extract the major latent semantic structure, at the same time, eliminates noise and unreliability of word usages. Term-document matrix is a large sparse matrix. In this chapter, we discussed the numerical solutions to decompose a large sparse matrix.

LSA has been an active research subject. In this chapter, we reviewed the many applications LSA has been applied. Since LSA is a completely automatic method, it has been applied to a wide range of problems.

In the last part of the chapter, we evaluated LSA against a text corpus of three years of Wall Street Journal articles. The text corpus in this study is not a calibrated text

collection, thus it is impossible to measure the precision and recall performance of the LSA retrieval system. Still, the evaluation against the text corpus allows us to evaluate three things, 1) the tools developed in this study; 2) the computation cost of LSA; 3) the sample retrieval results and overall effectiveness of the LSA system.

The retrieval results by LSA present us an opportunity to see its power of retrieving words and terms that are semantically close. In word retrieval, we picked words in three different semantic concept spaces, namely financial, medical and political. The results clearly showed the ability of LSA to cluster words based on their semantic meanings. In document retrieval, the results showed LSA is able to pick up documents which are semantic similar, though they might not share many common words.

## Chapter 4

# Design and Evaluation of Robust Annotation

## Persistence Scheme

---

In Chapter One, after we made extensive review on the state-of-the art research on digital annotation, we arrived at several important conclusions,

- Building a robust annotation persistence scheme over dynamic documents is a crucial and indispensable component in designing an annotation system.
- To meet the requirements of such a robust annotation persistence scheme, annotation anchor information can be extracted from the study of annotation text only (i.e. annotation anchor text and surrounding context)
- Annotation persistence over dynamic documents is a specialized information retrieval problem.

These conclusions lead us to design the annotation's anchor formulation by exploiting all potential information embodied in the annotation's anchor text and surrounding context. In Chapter one, we designed the annotation's anchor to include three location descriptors to capture three important aspects of the annotation's anchor information,

- *Meta-structure information location descriptor*
- *Keyword location descriptor*
- *Semantic concept location descriptor*

Meta-structure information represents the metadata context in which the annotation's anchor resides, such as *Chapter 1 Introduction/Section 1.3 Concurrency*. This information can be readily extracted when document texts are parsed. We differentiate the tree-like information from XPath, since we make no attempt to convert the document format to XML format. If the document is in XML format, then the meta-structure information will normally be its XPath locator. The depth of the tree-like metadata



context information is restricted to the high level structures manifested in the document text itself. We make no attempt to further de-segment the text into sub-contexts.

Matching of the meta-structure information is part of the important criteria to evaluate the relevance of the candidate annotation anchor with the original annotation anchor. Our observation of text modification tells us that when the meta-structure of the document is maintained, text within the meta-structure segments in the original version usually stays (though with or without modifications) in the revised version; when the meta-structure of a document is modified, text usually follows with its meta-structure segment in the newer version. Pre-match of the meta-structure information among the original and revised version of a document can be easily carried out by humans. Once this matching is performed, the meta-structure information associated with each text segment can be compared. Mismatch of the meta-structure information reduces the confidence of annotation persistence between the candidate anchor and the original anchor.

The second location descriptor contains a list of “keywords”, words which possess strong indexing powers. As indicated in Chapter 2, when words appear rarely in a version of a document, the re-appearance of those words in the newer version of the document presents an artifact indicating that the two paragraphs containing those words are very likely related. In Chapter Two, we used normalized word entropy to measure the indexing power of words. The property of entropy implies that the more uniformly distributed a word is, the higher its entropy value is. Rare appearing words usually have small entropy values.

The third location descriptor contains the semantic concept of the annotation text, which is the determining factor in our annotation persistence scheme design. Annotation persistence is only meaningful for the original anchor and the candidate anchor when they contain semantically similar content/context. The power of annotations stems from being semantic context-based, and they are worded with the semantic context assumed. When the semantics of the context change, annotations lose their applicability. In Chapter Three, we investigated and evaluated an information retrieval model which is capable of extracting text semantics and retrieving similar texts based on semantic comparisons.

As indicated in Chapter One, a robust annotation persistence scheme needs to include two elements, anchor representation, which describes an annotation anchor location

within a document, and a reattachment algorithm, which attempts to reposition the annotation anchor within a possible mutated target. In this chapter, before we present the reattachment algorithm, we will finalize our definition of the annotation anchor representation. We will elaborate further on the parsing of the meta-structure information associated with each text segment. We then provide the reasoning behind the selection of the threshold of choosing “keywords”. For the text semantics, we quantify the semantic closeness of document texts into a continuous numeric number ranging from 0 to 1, with 1 meaning two document texts are identical and 0 means they are completely different.

One of the most important steps in designing the reattachment algorithm is to define an index which can be used to measure reattachment confidence among annotation anchors. We call this index the *reattachment confidence index*, which is a quantitative measure of how close a candidate anchor is to the original anchor. In this chapter, we present the definition of the reattachment confidence index. A small training test is performed to calibrate the confidence index to quantify the proportions of contributions from the three location descriptors. Based on the score of the reattachment confidence index, the reattachment algorithm will make evaluate the reattachment of an annotation, whether it is reattached with high confidence, medium confidence, or left orphaned.

In the second half of this chapter, a full-scale evaluation of the annotation persistence scheme is performed. By examining a pre-edition and post-edition of an introductory computer textbook, we first identify the differences between the versions and then we mark the texts in the pre-edition which are subsequently modified in the post-edition as one of the followings:

- lightly modified; minor editing, rewording, but otherwise should be treated as the same as the pre-edition version
- moderately modified; a large part of the text is modified and reworded, the semantics of the post-edition version overlaps those in the original pre-edition version, but there are some additional concepts/terms which are not present in the pre-edition text;
- heavily modified; totally deleted from pre-edition or heavily rewritten with little semantic overlapping with post-edition version

Ideally, the application of the annotation persistence scheme should present the following results; if the annotation is made on a lightly modified pre-edition text, the persistence scheme should make the reattachment to the post-edition text with a high success rate and with high confidence index values; if the annotation is made on moderately modified pre-edition text, the persistence scheme will only be able to identify post-edition text with medium confidence index value, and the scheme will present users alternative options for possible reattachment solutions; for heavily modified annotation text, the scheme should orphan the annotation with high confidence rather than re-positioning them with low confidence in order to minimize the false positives in IR terms.

## **4.1 Design of Robust Annotation Persistence Scheme**

### **4.1.1 Annotation Anchor Formulation**

In Chapter One, we design the annotation's anchor to include three location descriptors to capture three important aspects of the annotation's anchor information,

- *Meta-structure information location descriptor*
- *Keyword location descriptor*
- *Semantic concept location descriptor*

In this chapter, we elaborate in detail how each one is represented and, in particular, answer the following questions,

- How is meta-structure information parsed and compared?
- What is the threshold of determining a word to be a “keyword” and how are “keywords” compared among annotation anchors?
- Can we quantify the closeness between document texts in a continuous spectrum ranging from 0 to 1?

### **Parsing and Comparison of Meta-Structure Information**

Each document naturally presents some form of content structures. In a technical book, this structure can easily be obtained from studying the table of contents. For example, we obtained a pre-edition and post-edition copy of an introductory computer textbook from

MIT press. Both book versions contain a table of contents section. In the pre-edition of the book, it contains total thirteen chapters and four additional appendices. Each chapter is further divided into sections and subsections. For example in Chapter Four, “Declarative Concurrency”, there are eleven sections. Within each section, there are a several subsections. A typical path from the root to a leaf of the content structure looks like this,

- *Chapter 4 Declarative Concurrency*

- *4.3 Streams*

- *4.3.4 Stream Objects*

Under section *4.3.4 Stream Objects*, we pick one paragraph of text:

*We call a stream object an object because it has an internal state that is accessed in a controlled way (by messages on streams). Throughout the book, we will use the term “object” for several such entities, including port objects, passive objects, and active objects. These entities differ in how the internal state is stored and how the controlled access is defined. The stream object is the first and simplest of these entities.*

If the document is in XML format, the meta-structure information is the XPath of the above paragraph. No additional processing of the text is needed. XML is clearly the easiest format to derive meta-structure information. However, since we don’t make presumption about the text format, we don’t need to convert to XML in order to extract meta-structure information. Depending on the characteristics of the available document format, any proper text parsing strategies can be used.

The textbook we obtained is in pdf format. We converted them to ASCII text with lines returns clearly marked. Here is the strategy we build to parse the meta-structure information for particular paragraph of content.

We build the parser by first parsing the table of contents of the textbook to build a table-of-contents tree. The parser then scans sequentially from the beginning of the book to, say, the above paragraph. In our case, each node representing content structure

information is written as a separate single line in the textbook (e.g. “*Chapter 4 Declarative Concurrency*” appears in the textbook as a single line first, then after some contents, “*4.3 Streams*” also appears as a single line, as does “*4.3.4 Stream Objects*”). When a parser reads each line, it checks if this line of text represents a line of textbook content structure by comparing it to the table-of-contents tree we build earlier. If yes, it will record and update the current content structure information. Each paragraph of document text thereafter is then associated with the current content structure information until a new line of content structure information appears.

When a user makes an annotation using our test annotation editor (say on above paragraph), the content structure information (i.e. *Chapter 4 Declarative Concurrency/4.3 Streams/4.3.4 Stream Objects*) is saved into the database along with annotation text and annotation contents.

In our design of the annotation persistence scheme, a one-time pre-match of the content structure information between two different versions of the document is performed if there are differences. This involves human understanding of the document structure and judgment of the document structure changes. For example, in the pre-edition of the textbook, the entire section *2.5 Memory management* under *Chapter 2, Declarative Computation Model* is merged into section *2.4 Kernel language semantics* in the post-edition. Section *2.5.1 Last call optimization* in the pre-edition of the textbook becomes *2.4.6 Last call optimization* in the post-edition of the textbook. *2.5.2 Memory life cycle; 2.5.3 Garbage collection; 2.5.4 Garbage collection is not magic; and 2.5.5 The Mozart garbage collector* in the pre-edition all are merged into a new section in post-edition, *2.4.7 Active memory and memory management*.

Pre-edition	Post-edition
2.5.1 Last call optimization	2.4.6 Last call optimization
2.5.2 Memory life cycle	2.4.7 Active memory and memory management
2.5.3 Garbage collection	
2.5.4 Garbage collection is not magic	
2.5.5 The Mozart garbage collector	

**Table 4.1** Meta-structure matching table to pre-match the meta-structure information among versions

Table 4.1 shows the pre-matching of meta-structure information among the different versions. When we compare the meta-structure information associated with original and candidate annotation anchors, if they are different, the initial meta-structure matching table is examined to see if a match can be found. If so, the meta-structure information is considered equivalent. Otherwise, the meta-structure information of the original and the candidate anchor are considered different.

To come up with a quantitative measure of the comparisons of meta-structure information between annotation anchors, we define a Boolean index, **meta-structure information index (*mi*)**.

If meta-structure information matches between annotation’s original anchor and candidate anchor,

$$mi = 1$$

If meta-structure information does not match between annotation’s original anchor and candidate anchor,

$$mi = 0$$

### **Threshold of Being “Keyword” and Definition of *Keyword Index***

Like selecting stopwords in IR, determining the threshold that determines “keywords” is not an easy task, since the relative frequency and entropy of words change continuously from low to high. If we use normalized word entropy to decide if the word is a keyword, we need to decide the cut-off point, i.e. words whose entropy value is smaller than the

cut-off point, are considered to be keywords, and those whose entropy value is greater are not.

Making a decision of entropy cut-off point depends on the length of the document or the size of the corpus. Assume we have a single document, if the document is very short, we may chunk document into a limited number of semantically self-containing segments. Being a keyword in our context implies it has strong indexing power, the power of referencing the segment of text uniquely. If the number of document segments in the collection is small, the frequency of the occurrences of keyword must be very small to guarantee its referencing power, which suggest the entropy cut-off number for choosing “keywords” must be very close to 0. (In terms of frequency of occurrences, it means roughly only 1 or 2 occurrences of words in the entire document). With the growth of document length, the cut-off number naturally shifts away from 0, which allows more frequently appearing words to be keywords.

As we indicated in Chapter Two, Luhn proposed two cut-offs to Zipf’s curve (Figure 2.4). The upper cut-off excludes all common words; while the lower cut-off takes off rare ones. Luhn states that both common and rare words don’t contribute significantly to the content of the document. Luhn went on further to claim that the resolving power of words to discriminate content (or semantics) reaches the peak at a rank order position half way between the two cut-offs, and decays to zero for rare and common words. Thus words in the middle are significant in document semantic analysis.

From the perspective of Latent Semantic Analysis, when the document collections are large, the choice of lower cut-off (which removes rare occurring words) generally needs to balance between the memory limitation of the computation machine and the contribution of semantics from rare occurring words. We know based on Zipf’s law, rare occurring words compose a large portion of documents total vocabulary. For example, words occurring only once compose nearly half of the vocabulary, words occurring twice compose nearly  $1/6^{\text{th}}$  of total word vocabulary and so on. Including rare occurring words in the LSA terms will significantly increase the size of the term-document matrix, thus requiring a much larger computation memory. At the same time, the contribution of semantics from rare occurring words decreases with the rarity of word occurrences.

Clearly, the choice of entropy cut-off point should also be one of parameters in the reattachment algorithm, which need to be calibrated. The selection of the cut-off point will have an impact on the performance of the reattachment scheme.

In Chapter Three, we evaluated normalized word entropy for a very large corpus. Table 2.4 shows the entropy measures of words along with their frequency of occurrences in the corpus. We made some limited testing and we consider a cut-off value of 0.2~0.25 is a reasonable good threshold. A value of 0.2 ~ 0.25 of normalized word entropy correspond to 20-30 word frequency of occurrences in the whole corpus (in this case, corpus is very large).

In this chapter, we study and evaluate our model of annotation reattachment. We have selected a pre-edition and post-edition textbook as our target. The textbook is about 900 pages long with 24500 total word occurrences and 6800 unique words. This is a relatively small text corpus, but it is a reasonably large book. The evaluation of entropy reveals that words with normalized entropy value of 0.2 ~ 0.25 corresponds roughly to a frequency of 6-7 occurrences in the document.

In this study, we use 0.25 as our entropy cut-off number for selecting keywords.

To compare keywords between annotation's original anchor and candidate anchor, we define a **Keyword Index (*ki*)**. If we define  $n_k$  as the number of keywords that exist in annotation's original anchor,  $n'_k$  as the number of keywords that exist in the annotation's original anchor as well as in annotation's candidate anchor (thus,  $n'_k \leq n_k$ ), the keyword index is defined as:

$$ki = \frac{n'_k}{n_k} \quad \text{if } n_k \neq 0$$

$$ki = \text{NaN} \quad \text{if } n_k = 0$$

In the event that there is no keyword found in annotation's original anchor, keyword index is not applicable.

Keyword index essentially measures how much of a percentage of keywords that are still left in the newer version of text when annotation is reattached. Keyword index



measures 0 if no keyword is left in the candidate annotation anchor; it measures 1 if all keywords are maintained in the candidate anchor.

### **Scaling of Semantic Closeness between Document Texts to [0, 1]**

In Chapter Three, we measured the semantic closeness of two documents by the semantic angles between them. Theoretically angles between two high dimensional vectors can range from 0 to 180 degrees. Semantic angles between two document vectors, however, range normally from 0 to 90 degrees, with 0 being identical and 90 far different.

Document vectors are vectors with non-negative values (cell values are frequency of occurrences of words if no local and global modifications are applied). This property of document vectors implies that document vectors are clustered only in a local sub-space of the high dimensional space.

We use a linear scale to quantify the semantic closeness between two document vectors. Assume the semantic angle measured between two document vectors is  $\theta$ . We define a **Semantic Similarity Index (si)** as

$$si = \frac{\theta}{90}$$

Semantic Similarity Index measure from 0 to 1, with 0 indicating two documents are identical and 1 indicating two documents are semantically quite different.

#### **4.1.2 Reattachment Confidence Index**

One of the most important components in designing the reattachment algorithm is to define an index which can be used to measure reattachment confidence among annotation anchors. We call this index the **Reattachment Confidence Index (rci)**, which is a quantitative measure of how close a candidate anchor is to the original anchor.

The reattachment confidence index includes contributions from the three location descriptors. The portion of the contribution from each individual location descriptor needs to be calibrated. We define the reattachment confidence index as the following,

$$rci = \alpha mi + \beta ki + \gamma (1-si) \quad \text{if } ki \neq \text{NaN}$$

$$rci = (\alpha mi + \gamma (1-si)) / (1 - \beta) \quad \text{if } ki = \text{NaN}$$

Where  $\alpha + \beta + \gamma = 1$ , ( $\alpha$ ,  $\beta$ , and  $\gamma > 0$ , indicating all contributions are positive).

Since all three indexes, meta-structure information index ( $mi$ ), keyword index ( $ki$ ) and semantic similarity index ( $si$ ) all measure between 0 to 1, the reattachment confidence index is a value between 0 and 1 as well, with a value of 1 indicating annotation anchors are identical and a value of 0 indicating annotation anchors are very different.

In the next section, we calibrate parameters of  $\alpha$ ,  $\beta$  and  $\gamma$ , which represent the proportions of the contribution to the reattachment confidence index from each individual index.

#### 4.1.3 Design and Calibration of the Reattachment Algorithm

To make the reattachment algorithm a useful solution, not only does it need an index to measure the closeness between the annotation's original anchor and the candidate anchor, the reattachment algorithm should also make one of the following three choices based on the valuation of the index:

1. The candidate anchor in the newer version of the document preserves great similarity to the original anchor. The annotation can be transferred to the newer version with high confidence.
2. Somewhat similar candidate anchors are found in the newer version of the document. The algorithm suggests users should decide if the annotation can be reattached.
3. No similar anchor can be found in the newer version of the document, and the annotation should be orphaned.

Theoretically, the three choices that the reattachment algorithm can make should correspond to three increasing levels of changes/modifications on the annotation's anchor text. If the annotation's anchor text is only lightly modified (including minor editing, deleting, rewording), the reattachment algorithm should favor the first choice. If the

annotation's anchor text is heavily modified or even deleted from the original version of the document, the reattachment algorithm should choose to orphan the annotation. When the modification is in the middle, the reattachment algorithm can leave the users to decide what they want to do with the annotations. Even in this case, the reattachment algorithm should suggest the possible reattachment candidates.

In order for the reattachment algorithm to decide which of these choices is appropriate to reposition an annotation based on the value of reattachment confidence index, two threshold values for the reattachment confidence index are needed.

We define  $\lambda$  as the lower threshold,

if  $rci < \lambda$ , the annotation should be orphaned;

and  $\eta$  as the upper threshold, ( $\eta > \lambda$ )

if  $rci > \eta$ , the annotation should be repositioned with high confidence;

There is one more constraint before we start to present the full definition of the reattachment algorithm. Throughout this thesis, we have strongly argued the importance of the semantics of the context where annotations reside. Annotation persistence is only meaningful for annotation anchors when they contain semantically similar content or context. The contribution by the semantic similarities between annotation anchors to the reattachment confidence index should take a much larger weight than those by keywords and the meta-structure information. This constraint implies

$$\gamma \gg \alpha$$

and

$$\gamma \gg \beta$$

At this point, we are now able to make a formal description of the design of the annotation reattachment algorithm.

### **Design of Annotation Reattachment Algorithm**

The problem:

- Two versions of documents, V1 and V2; V2 is a mutation of V1; the changes from V1 to V2 include, rewording, deleting, moving, or a combination of all three.
- Annotations are made on arbitrary locations on V1
- How to reposition annotations automatically on V2?

The solution:

- Select annotation's anchor text and surrounding context from V1
- Extract features (meta-structure information, keywords, and semantics) from the annotation's anchor text and surrounding context in V1.
- Select candidate anchor texts and surrounding contexts from V2 and extract features
- Derive meta-structure information index ( $mi$ ), keyword index ( $ki$ ), and semantic similarity index ( $si$ ) by comparing features between anchors of V1 and V2.
- Derive reattachment confidence index ( $rci = \alpha mi + \beta ki + \gamma (1-si)$ )
- Sort candidate annotation anchors by  $rci$  values.
- Take the candidate annotation anchor with the highest  $rci$  ( $ric_{max}$ ). Compare  $ric_{max}$  with the lower and the upper  $rci$  threshold. If it is greater than upper  $rci$  threshold ( $\eta$ ), annotation is repositioned to the candidate anchor with great confidence. If it is lower than lower  $rci$  threshold ( $\lambda$ ), annotation should be orphaned. If it is in between  $[\lambda, \eta]$ , the reattachment algorithm recommends the user to decide if the annotation should be repositioned among a selection of candidate annotation anchor locations.
- The parameters of  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\eta$  and  $\lambda$  are positive values between 0 and 1, with constraints of  $\gamma \gg \alpha$ ,  $\gamma \gg \beta$ , and  $\eta > \lambda$ .

## Calibration of Annotation Reattachment Algorithm

There are total of five parameters in the design of annotation reattachment algorithm,

- $\alpha$ ,  $\beta$  and  $\gamma$  are all positive values (between 0 and 1) representing the proportion of contributions to reattachment confidence index from the three individual indices.
- $\eta$  and  $\lambda$  are upper and lower *rci* thresholds. Based on these values, the reattachment algorithm makes decisions on whether and where the annotation should be reattached.

To calibrate the parameters in the annotation reattachment algorithm, a small training test is usually performed. In the training test, the document with versions is pre-analyzed with mutations between versions clearly marked. In the training document, text segments with modifications between versions are pre-marked to fall into one of the following three categories:

- Lightly modified; minor editing, rewording, with the semantics largely preserved between versions. The reattachment algorithm should reposition the annotation on this text segment with high confidence.
- Moderately modified; a large part of the text is modified and reworded, and the semantics between versions overlap. The system may not be able to reposition the annotation on this text segment with high confidence.
- Heavily modified; total deletion, or rewritten with semantics largely different in the newer version. The system should orphan the annotation.

The annotation reattachment algorithm is run against the training document, and the parameters are adjusted to make the system recommendation on the repositioning of annotations consistent with those recommended by human judgment.

In this study, we obtained a textbook (yet to be published) from MIT Press written by two professors from the Swedish Institute of Computer Science. The textbook has two versions, a pre-edition version and post-edition version. The textbook is an introductory level college computer science book. The book is about 900 pages long. It includes 13 chapters and 4 appendices. The following table is a top-level table-of-contents of the book.

<p>Table of Contents</p> <p>Chapters</p> <ol style="list-style-type: none"><li>1. Introduction</li><li>2. Declarative Computation Model</li><li>3. Declarative Programming Model</li><li>4. Declarative Concurrency</li><li>5. Message-Passing Concurrency</li><li>6. Explicit State</li><li>7. Object-Oriented Programming</li><li>8. Shared-State Concurrency</li><li>9. Relational Programming</li><li>10. Graphical User Interface Programming</li><li>11. Distributed Programming</li><li>12. Constraint Programming</li><li>13. Language Semantics</li></ol> <p>Appendices</p> <ol style="list-style-type: none"><li>A. Mozart System Development Environment</li><li>B. Basic Data Types</li><li>C. Language Syntax</li><li>D. General Computation Model</li></ol>
---

**Table 4.2** Table of contents of the pre-edition textbook

Since we have only one book with versions available to test our theory, we use this book as both training and test data. In this study, we use Chapter 1 to calibrate the reattachment algorithm. We identify the version changes in Chapter 1, pre-mark them and then use the pre-marked results to calibrate the parameters in the reattachment algorithm.

One of the important steps in LSA is to build the latent semantic high-dimensional space by analyzing a large domain corpus. We don't have a large domain corpus on computer science. In this study, we used the texts from the two book versions as an approximation of the domain corpus. We feel that because of the diverse contents of the book and length of the texts, the semantic space built from the book are a good approximation to the domain semantic space, and that it is powerful enough to cluster text segments of the book by their semantic contents.

To build a high dimensional semantic space, we first perform the text pre-processing on the book. Since LSA is a study on text only, all figures/pictures and equations/formulations were removed. We segmented texts from the book (in two versions) naturally by paragraph boundaries. We feel they are usually long enough to contain unique and strong semantics. We made no attempt to group paragraphs or segment them further into larger or smaller units, say, by their semantic similarities.

From the study of the texts in Chapter One of the textbook, we identify several modifications ranging from minor to major. We list them as followings,

Minor Changes:

Case 1:

Pre-edition	Functional abstraction. The definition of Comb uses the existing function Fact in its definition. It is always possible to use existing functions when defining new functions. Using functions to build abstractions is called functional abstraction.
Post-edition	Functional abstraction. The function Comb calls Fact three times. It is always possible to use existing functions to help define new functions. This principle is called <i>functional abstraction</i> because it uses functions to build abstractions.

Case 2:

Pre-edition	<p>A program can give wrong results even after it is proved correct. This could happen if the system on which it runs is not implemented correctly. How can we be sure that the system satisfies the semantics? Verifying system is a major undertaking. This requires verifying the operating system, the hardware, and the physics upon which the hardware is based! These are all important tasks, but they are beyond the scope of the book. We place our trust in the Mozart developers, software, hardware companies, and physicists.</p>
Post-edition	<p>A program that is proved correct can still give incorrect results, if the system on which it runs is incorrectly implemented. How can we be confident that the system satisfies the semantics? Verifying this is a major task: it means verifying the compiler, the run-time system, the operating system, and the hardware! This is an important topic, but it is beyond the scope of the present book. For this book, we place our trust in the Mozart developers, software companies, and hardware manufacturers.</p>

Case 3:

Pre-edition	<p>We would like to have our program executing several concurrent activities, with each activity running on its own pace. This concept is called concurrency. There should be no interference between the activities, unless we decide there is a need of communication between them. This is how the real world works outside of the system. We would like to be able to do this inside the system as well.</p>
Post-edition	<p>We would like our program to have several independent activities, each of which executes at its own pace. This is called concurrency. There should be no interference between the activities, unless the programmer decides that they need to communicate. This is how the real world works outside of the system. We would like to be able to do this inside the system as well.</p>



Case 1 to Case 3 are classical modifications a document text can go through between versions. The modifications in all three cases are considered minor, although some editing and rewording are made to the text. The semantics between versions are kept intact. The reattachment algorithm should reposition these cases with a high degree of confidence.

### Deletion

Case 4:

Pre-edition	This chapter has introduced the following computation models: Declarative model (chapters 2 and 3). Declarative programs define mathematical functions. They are the easiest to reason about and to test. The declarative model is important also because it contains many of the ideas that will be used in later, more expressive models.
-------------	---

Case 5:

Pre-edition	Concurrent declarative model (chapter 4). Adding dataflow concurrency gives a model that is still declarative but that allows a more flexible, incremental execution. Lazy declarative model (section 4.5). Adding laziness allows calculating with potentially infinite data structures. This is good for resource management and program structure.
-------------	---

Case 6:

Pre-edition	Stateful model (chapter 6). Adding explicit state allows writing programs whose behavior changes over time. This is good for program modularity. If written well, i.e., using encapsulation and invariants, these programs are almost as easy to reason about as declarative programs.
-------------	--

Case 7:

Pre-edition	Object-oriented model (chapter 7). Object-oriented programming is a programming style for stateful programming with data abstractions. It makes it easy to use powerful techniques such as polymorphism and inheritance.
-------------	--

Case 8:

Pre-edition	Shared-state concurrent model (chapter 8). This model adds both concurrency and explicit state. If programmed carefully, using techniques for mastering interleaving such as monitors and transactions, this gives the advantages of both the stateful and concurrent models.
-------------	---

Case 9:

Pre-edition	In addition to these models, the book covers many other useful models such as the declarative model with exceptions (section 2.7), the message-passing concurrent model (chapter 5), the relational model (chapter 9), and the specialized models of part II.
-------------	---

Case 4 – 9 are all deletions from the pre-edition textbook. They are summary statements for future chapters. Ideally, we would like to orphan them. If semantics of some paragraphs in the future chapters bear significant similarities to these summaries, we may wish to let users decide if they want to move annotations.

We did not find a case where we considered the modification of the text is larger than a minor change, but less than a heavy one. Nevertheless, we feel this represents an adequate training set to calibrate the parameters of the annotation reattachment algorithm.

By following the design constraints of the reattachment algorithm parameters we set earlier, after trial and error, we find the follow sets of parameters satisfy our model requirements.

$$\alpha = 0.15$$

$$\beta = 0.15$$

$$\gamma = 0.75$$

$$\eta = 0.70$$

$$\lambda = 0.50$$

By adopting this set of parameters, the *rci* is defined as,

$$rci = 0.15mi + 0.15ki + 0.75(1-si)$$

The annotation reattachment algorithm will then automatically reposition annotations on the newer version of the document as following:

- Reposition annotation if a candidate anchor exist where  $\max rci > 0.70$
- Orphan annotation if all candidate anchors'  $rci < 0.50$
- Present users candidate anchors if  $\max rci$  is between 0.50 and 0.70.

In the following, we present the result for each case.

Minor Changes: Case 1:

<p>Annotation anchor text in pre-edition</p> <p><i>rci</i> Ranking (in post-edition)</p>	Text	Functional abstraction. The definition of <u>Comb</u> uses the existing function <u>Fact</u> in its definition. It is always possible to use existing functions when defining new functions. Using functions to build abstractions is called functional abstraction.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts <ul style="list-style-type: none"> <li>○ 1.3 functions</li> </ul> </li> </ul>
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ comb                      0.9190</li> </ul>
<hr/>		
<p>Rank 1<sup>st</sup>:</p> <p><i>rci</i> = 0.8151 <i>mi</i> = 1 <i>ki</i> = 1 <i>si</i> = 0.2642</p>	Text	Functional abstraction. The function <u>Comb</u> calls <u>Fact</u> three times. It is always possible to use existing functions to help define new functions. This principle is called <i>functional abstraction</i> because it uses functions to build abstractions.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts <ul style="list-style-type: none"> <li>○ 1.3 functions</li> </ul> </li> </ul>
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ comb</li> </ul>
<hr/>		
<p>Rank 2<sup>nd</sup></p> <p><i>rci</i> = 0.4515 <i>mi</i> = 1 <i>ki</i> = 0 <i>si</i> = 0.5626</p>	Text	Because it uses functions to build abstractions. In this way large programs are like onions with layers upon layers of functions calling functions.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts <ul style="list-style-type: none"> <li>○ 1.3 functions</li> </ul> </li> </ul>
	Keywords	(no keyword maintained)
<hr/>		
<p>Rank 3<sup>rd</sup></p> <p><i>rci</i> = 0.3524 <i>mi</i> = 0 <i>ki</i> = 0 <i>si</i> = 0.4966</p>	Text	Functional programming consists of defining functions on complete values, where the functions are true functions in the mathematical sense. A language in which this is the only possible way to calculate is called a pure functional language. Let us examine how the declarative model relates to pure functional programming.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 2 declarative computation model <ul style="list-style-type: none"> <li>○ 2.7 advanced topics <ul style="list-style-type: none"> <li>▪ 2.7.1 functional programming languages</li> </ul> </li> </ul> </li> </ul>
	Keywords	(no keyword maintained)

In Case 1, the text is about *functional abstraction*. Users are very likely to annotate on this paragraph since this is the first time the book introduces this concept. It represents a term which bears significant semantics. It is interesting to note though, both “functional” and “abstraction” are not considered keywords in our entropy measure. The entropy value of “function” is 0.5300 and entropy of “abstraction” measures 0.4736. Both are greater than the threshold of 0.25. From the observation of the ranked lists, since semantic similarity index contributes to a much bigger weight to the *rci*, it is generally the case that semantic similarity decays with the ranking list. This case is an interesting exception and demonstrates the importance of the contributions from the unique words (“comb” in this case) and the meta-structure.

Case 2:

Annotation anchor text in pre-edition	Text	A program can give wrong results even after it is proved correct. This could happen if the system on which it runs is not implemented correctly. How can we be sure that the system satisfies the semantics? Verifying system is a major undertaking. This requires verifying the operating system, the hardware, and the physics upon which the hardware is based! These are all important tasks, but they are beyond the scope of the book. We place our trust in the Mozart developers, software, hardware companies, and physicists.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts               <ul style="list-style-type: none"> <li>○ 1.6 correctness</li> </ul> </li> </ul>
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ verifying      0.8677</li> <li>✓ verifying      0.8677</li> <li>✓ trust            0.8236</li> <li>✓ developers    0.7575</li> <li>✓ companies     1</li> </ul>
RCI Ranking (in post-edition)		
Rank 1 <sup>st</sup> :	Text	A program that is proved correct can still give incorrect results, if the system on which it runs is incorrectly implemented. How can we be confident that the system satisfies the semantics? Verifying this is a major task: it means verifying the compiler, the run-time system, the operating system, and the hardware! This is an important topic, but it is beyond the scope of the present book. For this book, we place our trust in the Mozart developers, software companies, and hardware manufacturers.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts               <ul style="list-style-type: none"> <li>○ 1.6 correctness</li> </ul> </li> </ul>
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ verifying</li> <li>✓ verifying</li> <li>✓ trust</li> <li>✓ developers</li> <li>✓ companies</li> </ul>
<i>rci</i> = 0.8377 <i>mi</i> = 1 <i>ki</i> = 1 <i>si</i> = 0.2318		

<p>Rank 2<sup>nd</sup></p> <p><i>rci</i> = 0.3054 <i>mi</i> = 1 <i>ki</i> = 0 <i>si</i> = 0.7780</p>	Text	A program is correct if it does what we would like it to do. How can we tell whether a program is correct? Usually it is impossible to duplicate the program's calculation by hand. We need other ways. One simple way which we used before is to verify that the program is correct for outputs that we know. This increases confidence in the program, but it does not go very far. To prove correctness in general we have to reason about the program. This means three things:
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts <ul style="list-style-type: none"> <li>○ 1.6 correctness</li> </ul> </li> </ul>
	Keywords	(no keyword maintained)
<p>Rank 3<sup>rd</sup></p> <p><i>rci</i> = 0.2993 <i>mi</i> = 1 <i>ki</i> = 0 <i>si</i> = 0.7867</p>	Text	We use mathematical techniques to reason about the program using the semantics. We would like to demonstrate that the program satisfies the specification.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts <ul style="list-style-type: none"> <li>○ 1.6 correctness</li> </ul> </li> </ul>
	Keywords	(no keyword maintained)

In Case 2, the text is talking about program errors which are possibly caused by systems other than its own. The rewriting of the text clearly is more succinct, but the semantics of the text are kept intact. Annotations on the text on pre-edition should clearly be transplanted to the post-edition version with high degree of confidence.

Case 3:

Annotation anchor text in pre-edition          RCI Ranking (in post-edition)	Text	<p>We would like to have our program executing several concurrent activities, with each <u>activity</u> running on its own <u>pace</u>. This concept is called concurrency. There should be no <u>interference</u> between the activities, unless we decide there is a need of communication between them. This is how the real world works outside of the system. We would like to be able to do this inside the system as well.</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts               <ul style="list-style-type: none"> <li>○ 1.10 concurrency</li> </ul> </li> </ul>
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ activity      0.7777</li> <li>✓ pace          1</li> <li>✓ interference    0.8015</li> </ul>
Rank 1 <sup>st</sup> :   <i>rci</i> = 0.7790 <i>mi</i> = 1 <i>ki</i> = 0.6667 <i>si</i> = 0.2442	Text	<p>We would like our program to have several independent activities, each of which executes at its own <u>pace</u>. This is called concurrency. There should be no <u>interference</u> between the activities, unless the programmer decides that they need to communicate. This is how the real world works outside of the system. We would like to be able to do this inside the system as well.</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts               <ul style="list-style-type: none"> <li>○ 1.10 concurrency</li> </ul> </li> </ul>
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ pace</li> <li>✓ interference</li> </ul>
Rank 2 <sup>nd</sup>   <i>rci</i> = 0.3192	Text	<p>We introduce concurrency by creating threads. A thread is simply an executing program like the functions we saw before. The <u>di</u>ference is that a program can have more than one thread. Threads are created with the thread instruction. Do you remember how slow the original Pascal function was? We can call Pascal inside its own thread. This means that it will not keep other calculations from continuing. They may slow down, if Pascal really has a lot of work to do. This is because the threads share the same underlying computer. But none of the threads will stop. Here is an example:</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts               <ul style="list-style-type: none"> <li>○ 1.10 concurrency</li> </ul> </li> </ul>



<i>mi</i> = 1 <i>ki</i> = 0 <i>si</i> = 0.7582	Keywords	(no keyword maintained)
Rank 3 <sup>rd</sup>	Text	This creates a new thread. Inside this new thread, we call {Pascal 30} and then call Browse to display the result. The new thread has a lot of work to do. But this does not keep the system from displaying 99*99 immediately.
<i>rci</i> = 0.2647 <i>mi</i> = 1 <i>ki</i> = 0 <i>si</i> = 0.8361	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts <ul style="list-style-type: none"> <li>○ 1.10 concurrency</li> </ul> </li> </ul>
	Keywords	(no keyword maintained)

Case 4 to Case 10 are deletions. In the following table, we report the results of C4, C7 and C8.

Case 4:

<p>Annotation anchor text in pre-edition</p> <p>RCI Ranking (in post-edition)</p>	Text	<p>This chapter has introduced the following computation models: Declarative model (chapters 2 and 3). Declarative programs define mathematical functions. They are the <u>easiest</u> to reason about and to test. The declarative model is important also because it contains many of the ideas that will be used in later, more expressive models.</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts <ul style="list-style-type: none"> <li>○ 1.17 where do we go from here</li> </ul> </li> </ul>
	Keywords (1-entropy)	<p>✓ easiest                      0.8236</p>
<p>Rank 1<sup>st</sup>:</p> <p><i>rci</i> = 0.4122 <i>mi</i> = 0 <i>ki</i> = 0 <i>si</i> = 0.4111</p>	Text	<p>This book sticks to the traditional usage of declarative as stateless and imperative as stateful. We call the computation model of Chapter 2 "declarative", even though later models are arguably more declarative, since they are more expressive. We stick to the traditional usage because there is an important sense in which the declarative model really is declarative according to the literal meaning. This sense appears when we look at the declarative model from the viewpoint of logic and functional programming:</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 6 explicit state</li> </ul>
	Keywords (1-entropy)	<p>(no keyword maintained)</p>
<p>Rank 2<sup>nd</sup></p> <p><i>rci</i> = 0.3648 <i>mi</i> = 1 <i>ki</i> = 0 <i>si</i> = 0.6932</p>	Text	<p>This chapter has given a quick overview of many of the most important concepts in programming. The intuitions given here will serve you well in the chapters to come, when we define in a precise way the concepts and the computation models they are part of.</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts <ul style="list-style-type: none"> <li>○ 1.17 where do we go from here</li> </ul> </li> </ul>
	Keywords	<p>(no keyword maintained)</p>

<p>Rank 3<sup>rd</sup></p> <p><i>rci</i> = 0.3330  <i>mi</i> = 0  <i>ki</i> = 0  <i>si</i> = 0.5242</p>	Text	There exist many computation models that differ in how expressive they are and how hard it is to reason about programs written in them. The declarative model is one of the simplest of all. However, as we have explained, it has serious limitations for some applications. There are more expressive models that overcome these limitations, at the price of sometimes making reasoning more complicated. For example, concurrency is often needed when interacting with the external world. When such interactions are important then a concurrent model should be used instead of trying to get by with just the declarative model.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 4 declarative concurrency <ul style="list-style-type: none"> <li>○ 4.7 limitations and extensions of declarative programming <ul style="list-style-type: none"> <li>▪ 4.7.5 picking the right model</li> </ul> </li> </ul> </li> </ul>
	Keywords	(no keyword maintained)

In case 4, the text is a summary paragraph on future chapters 2 and 3. It talks about declarative models and programming. Although paragraphs are semantically related to declarative models and programming, their semantics are not strong enough to overcome the mismatch of meta-structure information and keywords. They are orphaned in this case.

Case 7:

Annotation anchor text in pre-edition  RCI Ranking (in post-edition)	Text	Object-oriented model (chapter 7). Object-oriented programming is a programming style for stateful programming with data abstractions. It makes it easy to use powerful techniques such as polymorphism and inheritance.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts               <ul style="list-style-type: none"> <li>○ 1.17 where do we go from here</li> </ul> </li> </ul>
	Keywords (1-entropy)	✓ polymorphism 0.8602
Rank 1 <sup>st</sup> :		
<i>rci</i> = 0.4498 <i>mi</i> = 0 <i>ki</i> = 0 <i>si</i> = 0.3574	Text	Stateful model with inheritance. Inheritance is the essential difference between object-oriented programming and most other kinds of stateful programming. It is important to emphasize that inheritance is a programming technique; the underlying computation model of object-oriented programming is simply the stateful model (or the shared-state concurrent model, for concurrent object-oriented programming). Object-oriented languages provide linguistic support for inheritance by adding classes as a linguistic abstraction.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 7 object oriented programming               <ul style="list-style-type: none"> <li>○ Motivations                   <ul style="list-style-type: none"> <li>▪ 7.1.1 inheritance</li> </ul> </li> </ul> </li> </ul>
	Keywords (1-entropy)	(no keyword maintained)
Rank 2 <sup>nd</sup>		
<i>rci</i> = 0.3848	Text	This chapter introduces a particularly useful way of structuring stateful programs called object-oriented programming. It introduces one new concept over the last chapter, namely inheritance, which allows to define ADTs in incremental fashion. However, the computation model is the same stateful model as in the previous chapter. We can loosely define object-oriented programming as programming with encapsulation, explicit state, and inheritance. It is often supported by a linguistic abstraction, the concept of class, but it does not have to be. Object-oriented programs can be written in almost any language.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 7 object oriented programming</li> </ul>

<i>mi</i> = 0 <i>ki</i> = 0 <i>si</i> = 0.4503	Keywords	(no keyword maintained)
Rank 3 <sup>rd</sup>	Text	chapter 7 covers object oriented programming and shows how to program with inheritance.
<i>rci</i> = 0.3793 <i>mi</i> = 0 <i>ki</i> = 0 <i>si</i> = 0.4580	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts <ul style="list-style-type: none"> <li>○ 1.14 classes</li> </ul> </li> </ul>
	Keywords	(no keyword maintained)

Like Case 4, Case 7 is also deleted from the pre-edition version. It is a summary statement for chapter 7, object oriented programming. The most semantic similar statements are all from Chapter 7, but as Case 4, they are not labeled as moderate modifications, hence Case 7 is orphaned. Case 8 is similar below.

Case 8:

Annotation anchor text in pre-edition  RCI Ranking (in post-edition)	Text	Shared-state concurrent model (chapter 8). This model adds both concurrency and explicit state. If programmed carefully, using techniques for mastering interleaving such as monitors and transactions, this gives the advantages of both the stateful and concurrent models.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 1 introduction to programming concepts             <ul style="list-style-type: none"> <li>○ 1.17 where do we go from here</li> </ul> </li> </ul>
	Keywords (1-entropy)	✓ mastering 1.0
<hr/>		
Rank 1 <sup>st</sup> :  <i>rci</i> = 0.3795 <i>mi</i> = 0 <i>ki</i> = 0 <i>si</i> = 0.4577	Text	Shared state concurrent model (see chapter 8 defined in section 8.1). This is the declarative model extended with both explicit state and threads. This model contains concurrent object oriented programming. The concurrency is more expressive than the declarative concurrent model since it can use explicit state to wait simultaneously on one of several events occurring this is called nondeterministic choice. Reasoning with this model is the most complex since there can be multiple histories interacting in unpredictable ways.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 4 declarative concurrency             <ul style="list-style-type: none"> <li>○ 4.7 limitations and extensions of declarative programming                 <ul style="list-style-type: none"> <li>▪ 4.7.6 extended models</li> </ul> </li> </ul> </li> </ul>
	Keywords (1-entropy)	(no keyword maintained)
<hr/>		
Rank 2 <sup>nd</sup>  <i>rci</i> = 0.3624 <i>mi</i> = 0 <i>ki</i> = 0 <i>si</i> = 0.4824	Text	Why not use declarative concurrency? Given the inherent difficulty of programming in the shared-state concurrent model, an obvious question is why not stick with the declarative concurrent model of Chapter 4? It is enormously simpler to program in than the shared-state concurrent model. It is almost as easy to reason in as the declarative model, which is sequential.
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 8 shared state concurrency</li> </ul>
	Keywords	(no keyword maintained)

Rank 3 <sup>rd</sup>  <i>rci</i> = 0.3608 <i>mi</i> = 0 <i>ki</i> = 0 <i>si</i> = 0.4846	Text	<p>Too much concurrency is bad. There is a model, the maximally concurrent model, that has even more concurrency than the stateful concurrent model. In the maximally concurrent model, each operation executes in its own thread. Execution order is constrained only by data dependencies. This has the greatest possible concurrency. The maximally concurrent model has been used as the basis for experimental parallel programming languages. But it is both hard to program in and hard to implement efficiently (see Exercise). This is because operations tend to be fine-grained compared to the overhead of scheduling and synchronizing. The shared-state concurrent model of this chapter does not have this problem because thread creation is explicit. This allows the programmer to control the granularity. We do not present the maximally concurrent model in more detail in this chapter. A variant of this model is used for constraint programming (see Chapter 12).</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 8 shared state concurrency <ul style="list-style-type: none"> <li>○ 8.2 programming with concurrency <ul style="list-style-type: none"> <li>▪ 8.2.1 overview of the different approaches</li> </ul> </li> </ul> </li> </ul>
	Keywords	(no keyword maintained)

## 4.2 Evaluation of Robust Annotation Persistence Scheme

Section 4.1 finalized the design of the annotation persistence scheme. In this section, we evaluate this model. As indicated earlier, we calibrated reattachment parameters using texts from Chapter 1 of the available textbook. We evaluate the model using the other chapters.

We have studied the pre-edition and post-edition texts of all the other chapters in the textbook. We mark paragraphs of text that possess modifications between versions. We also label them as one of the three choices of the modifications, i.e. lightly modified, moderately modified or heavily modified.

Our model is then applied to all text paragraphs with modifications. We classify them with the rules we set up in our annotation persistence scheme. The algorithm results are then compared with the perceptions of a human reader.

In this chapter, we first present a few examples where pre-edition texts are modified. We then present a table summarizing the model evaluation results.

### Sample Evaluation Results

In the following, we take a snapshot of a section in Chapter Five of the pre-edition (Figure 5.1 and Figure 5.2), along with its companion section in the post-edition version (Figure 5.3 and Figure 5.4). Reading both sections, it is easy to see the modifications between versions. First, there are large chunks of text at the beginning of the chapter 5 in the pre-edition that are deleted in the Chapter Five of the post-edition. Three paragraphs of introductory comments on “extending the declarative concurrent models” are also modified. After we run each paragraph in the snapshot section of Chapter Five in pre-edition through the algorithm, we present the findings by our model.



# Message-Passing Concurrency

Only then did Atreyu notice that the monster was not a single, solid body, but was made up of innumerable small steel-blue insects which buzzed like angry hornets. It was their compact swarm that kept taking different shapes.

– *The Neverending Story*, Michael Ende (1929–1995)

Message passing is a programming style in which a program consists of independent entities that interact by sending each other messages asynchronously, i.e., without waiting for a reply. Message passing is important in three areas:

- It is the basic framework for multi-agent systems, a discipline that views complex systems as a set of interacting “agents.” Agents are independent entities that work toward their own, local goals. If the interaction is designed properly, then the agents can also achieve global goals. For example, resource allocation can be done efficiently by selfish agents that interact according to mechanisms inspired by a market economy [177, 224].
- It is the natural style for a distributed system, i.e., a set of computers that can communicate with each other through a network. It is natural because it reflects the structure of the system and its costs. Distributed systems are becoming ubiquitous because of the continued expansion of the Internet. Older technologies for programming distributed systems, such as RPC, CORBA, and RMI, are based on synchronous communication. Newer technologies, such as Web services, are asynchronous. The techniques of this chapter apply directly to asynchronous technologies. (The particularities of programming distributed systems are explored further in chapter 11.)
- It lends itself well to building highly reliable systems. Since the message-passing entities are independent, if one fails the others can continue executing. In a properly designed system, the others reorganize themselves to continue providing a service. This idea is used by the Erlang language, which is used in telecommunications and high-speed networking (see section 5.7).

We define a computation model for message passing as an extension of the declarative concurrent model. We then use this model to show how to program with message passing.

**Figure 5.1** First page of Chapter Five in pre-edition

### *Extending the declarative concurrent model*

The declarative concurrent model of the last chapter cannot have observable nondeterminism. This limits the kinds of programs we can write in the model. For example, we saw that it is impossible to write a client/server program where the server does not know which client will send it the next message.

The message-passing concurrent model extends the declarative concurrent model by adding just one new concept, an asynchronous communication channel. This means that any client can send messages to the channel at any time and the server can read all the messages from the channel. This removes the limitation on what kinds of programs we can write. A client/server program can give different results on different executions because the order of client sends is not determined. This means that the message-passing model is nondeterministic and therefore no longer declarative.

We use a simple kind of channel called a port that has an associated stream. Sending a message to the port causes the message to appear on the port's stream. A useful programming technique is to associate a port with a stream object. We call the resulting entity a port object. A port object reads all its messages from the port's stream, and sends messages to other port objects through their ports. Each port object is defined by a recursive procedure that is declarative. This keeps some of the advantages of the declarative model.

### *Structure of the chapter*

The chapter consists of the following parts:

- Section 5.1 defines the message-passing concurrent model. It defines the port concept and the kernel language.
- Section 5.2 introduces the concept of port objects, which we get by combining ports with stream objects.
- Section 5.3 shows how to do simple kinds of message protocols with port objects.
- Section 5.4 explains how to design programs with concurrent components. It defines the basic concepts and gives a methodology for managing the concurrency.
- Section 5.5 gives a case study of this methodology. It uses port objects to build a lift control system.
- Section 5.6 shows how to use the message-passing model directly, without using the port object abstraction. This can be harder to reason about than using port objects, but it is sometimes useful.
- Section 5.7 gives an introduction to Erlang, a programming language based on port objects that is used to build highly reliable systems.
- Section 5.8 explains one advanced topic: the nondeterministic concurrent model, which is intermediate in expressiveness between the declarative concurrent model and the message-passing model of this chapter.

**Figure 5.2** Second page in Chapter 5 of pre-edition

## Chapter 5

# Message-Passing Concurrency

“Only then did Atreyu notice that the monster was not a single, solid body, but was made up of innumerable small steel-blue insects which buzzed like angry hornets. It was their compact swarm that kept taking different shapes.”

The Neverending Story, *Michael Ende* (1929–1995)

In the last chapter we saw how to program with stream objects, which is both declarative and concurrent. But it has the limitation that it cannot handle observable nondeterminism. For example, we wrote a digital logic simulator in which each stream object knows exactly which object will send it the next message. We cannot program a client/server where the server does not know which client will send it the next message.

We can remove this limitation by extending the model with an asynchronous communication channel. Then any client can send messages to the channel and the server can read them from the channel. We use a simple kind of channel called a *port* that has an associated stream. Sending a message to the port causes the message to appear on the port’s stream.

The extended model is called the *message-passing concurrent model*. Since this model is nondeterministic, it is no longer declarative. A client/server program can give different results on different executions because the order of client sends is not determined.

A useful programming style for this model is to associate a port to each stream object. The object reads all its messages from the port, and sends messages to other stream objects through their ports. This style keeps most of the advantages of the declarative model. Each stream object is defined by a recursive procedure that is declarative.

Another programming style is to use the model directly, programming with ports, dataflow variables, threads, and procedures. This style can be useful for building concurrency abstractions, but it is not recommended for large programs because it is harder to reason about.

**Figure 5.3** First page of Chapter 5 in post-edition

## Structure of the chapter

The chapter consists of the following parts:

- Section 5.1 defines the message-passing concurrent model. It defines the port concept and the kernel language. It also defines port objects, which combine ports with a thread.
- Section 5.2 introduces the concept of port objects, which we get by combining ports with stream objects.
- Section 5.3 shows how to do simple kinds of message protocols with port objects.
- Section 5.4 shows how to design programs with concurrent components. It uses port objects to build a lift control system.
- Section 5.5 shows how to use the message-passing model directly, without using the port object abstraction. This can be more complex than using port objects, but it is sometimes useful.
- Section 5.6 gives an introduction to Erlang, a programming language based on port objects. Erlang is designed for and used in telecommunications applications, where fine-grained concurrency and robustness are important.
- Section 5.7 explains one advanced topic: the nondeterministic concurrent model, which is intermediate in expressiveness between the declarative concurrent model and the message-passing model of this chapter.

## 5.1 The message-passing concurrent model

The message-passing concurrent model extends the declarative concurrent model by adding ports. Table 5.1 shows the kernel language. Ports are a kind of communication channel. Ports are no longer declarative since they allow observable nondeterminism: many threads can send a message on a port and their order is not determined. However, the part of the computation that does not use ports can still be declarative. This means that with care we can still use many of the reasoning techniques of the declarative concurrent model.

### 5.1.1 Ports

A *port* is an ADT that has two operations, namely creating a channel and sending to it:

- {NewPort *S* *P*}: create a new port with entry point *P* and stream *S*.

Copyright © 2001-3 by P. Van Roy and S. Haridi. All rights reserved.

**Figure 5.4** Second page of Chapter 5 in post-edition

## Paragraph 1: Orphaned annotation

Annotation anchor text in pre-edition  RCI Ranking (in post-edition)	Text	Message passing is a programming style in which a program consists of independent entities that interact by sending each other messages asynchronously, i.e., without waiting for a reply. Message passing is important in three areas:
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)
Rank 1 <sup>st</sup> :  <i>rci</i> = 0.4208 <i>mi</i> = 1 <i>ki</i> = NaN <i>si</i> = 0.7033	Text	We can remove this limitation by extending the model with an asynchronous communication channel. Then any client can send messages to the channel and the server can read them from the channel. we use a simple kind of channel called a port that has an associated stream. Sending a message to the port causes the message to appear on the port's stream.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)
Rank 2 <sup>nd</sup> :  <i>rci</i> = 0.3925 <i>mi</i> = 1 <i>ki</i> = NaN <i>si</i> = 0.7377	Text	Section 5.3 shows how to do simple kinds of message protocols with port objects.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)

Since this paragraph is deleted from pre-edition, after examine the post-edition, we decide any annotation made against this paragraph should be orphaned. Our annotation persistence system makes the right decision.

## Paragraph 2: orphaned annotation

Annotation anchor text in pre-edition          RCI Ranking (in post-edition)	Text	<p>It is the basic <u>framework</u> for <u>multi-agent</u> systems, a <u>discipline</u> that views complex systems as a set of interacting "<u>agents</u>." Agents are independent entities that work toward their own, local goals. If the interaction is designed properly, then the <u>agents</u> can also achieve global <u>goals</u>. For example, resource <u>allocation</u> can be done efficiently by <u>selfish agents</u> that interact according to <u>mechanisms</u> <u>inspired</u> by a market economy [177, 224].</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 5 message passing concurrency</li> </ul>
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ framework 0.9118</li> <li>✓ multi 0.7952</li> <li>✓ agent 0.8016</li> <li>✓ discipline 0.7525</li> <li>✓ agents 0.8896</li> <li>✓ agents 0.8896</li> <li>✓ goals 0.8791</li> <li>✓ agents 0.8896</li> <li>✓ goals 0.8791</li> <li>✓ allocation 0.8602</li> <li>✓ agents 0.8896</li> <li>✓ mechanisms 0.7738</li> <li>✓ inspired 0.8602</li> </ul>
Rank 1 <sup>st</sup> :	Text	<p>To design a concurrent application, the first step is to model it as a set of concurrent activities that interact in well defined ways. Each concurrent activity is modeled by exactly one concurrent component. A concurrent component is sometimes known as an <u>agent</u>. <u>Agents</u> can be reactive have no internal state or have internal state. The science of programming with <u>agents</u> is sometimes known as <u>multi agent</u> systems often abbreviated as mas. Many different protocols of varying complexities have been devised in mas. This section only briefly touches on these protocols. In component based programming <u>agents</u> are usually considered as quite simple entities with little intelligence built in. In the artificial intelligence community <u>agents</u> are usually considered as doing some kind of reasoning.</p>

$rci = 0.2487$ $mi = 0$ $ki = 0.4615$ $si = 0.7434$	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 5 message passing concurrency             <ul style="list-style-type: none"> <li>○ 5.4 program design for concurrency                 <ul style="list-style-type: none"> <li>▪ 5.4.1 programming with concurrent components</li> </ul> </li> </ul> </li> </ul>
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ multi 0.7953</li> <li>✓ agent 0.8016</li> <li>✓ agents 0.8896</li> <li>✓ agents 0.8896</li> <li>✓ agents 0.8896</li> <li>✓ agents 0.8896</li> </ul>

This paragraph of text is deleted from the pre-edition. Our examination of the book versions suggests annotations on this text should be orphaned. Our annotation persistence scheme correctly makes the decision.

### Paragraph 3: orphaned annotation

Annotation anchor text in pre-edition	Text	It is the natural style for a distributed system, i.e., a set of computers that can communicate with each other through a network. It is natural because it reflects the structure of the system and its costs. Distributed systems are becoming ubiquitous because of the continued expansion of the Internet. Older technologies for programming distributed systems, such as RPC, CORBA, and RMI, are based on synchronous communication. Newer technologies, such as Web services, are asynchronous. The techniques of this chapter apply directly to asynchronous technologies. (The particularities of programming distributed systems are explored further in chapter 11.)
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 5 message passing concurrency</li> </ul>
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ reflects 0.9118</li> <li>✓ costs 0.8602</li> <li>✓ becoming 0.8016</li> <li>✓ ubiquitous 0.9118</li> <li>✓ continued 0.8602</li> <li>✓ expansion 1</li> <li>✓ older 0.8602</li> <li>✓ rpc 0.8236</li> <li>✓ corba 0.9118</li> <li>✓ services 1</li> <li>✓ explored 0.824</li> </ul>
	RCI Ranking (in post-edition)	



<p>Rank 1<sup>st</sup>:</p> <p><i>rci</i> = 0.3793  <i>mi</i> = 0  <i>ki</i> = 0.0909  <i>si</i> = 0.4776</p>	<p>Text</p>	<p>A distributed system is a set of computers that are linked together by a network distributed systems are ubiquitous in modern society. The canonical example of such a system the internet has been growing exponentially ever since its inception in the late 1970s. The number of host computers that are part of it has been doubling each year since 1980. The question of how to program a distributed system is therefore of major importance this chapter shows one approach to programming a distributed system. For the rest of the chapter we assume that each computer has an operating system that supports the concept of process and provides network communication. Programming a distributed system then means to write a program for each process such that all processes taken together implement the desired application. For the operating system a process is a unit of concurrency. This means that if we abstract away from the fact that the application is spread over different processes this is just a case of concurrent programming. Ideally distributed programming would be just a kind of concurrent programming and the techniques we have seen earlier in the book would still apply.</p>
	<p>MetaInfo</p>	<ul style="list-style-type: none"> <li>▪ chapter 5 message passing concurrency <ul style="list-style-type: none"> <li>○ 5.4 program design for concurrency <ul style="list-style-type: none"> <li>▪ 5.4.1 programming with concurrent components</li> </ul> </li> </ul> </li> </ul>
	<p>Keywords (1-entropy)</p>	<p>✓ ubiquitous      0.9118</p>

As the case of paragraph 2, our model correctly decides to orphan annotations.

#### Paragraph 4: orphaned annotation

Annotation anchor text in pre-edition	Text	It lends itself well to building highly reliable systems. Since the message-passing entities are independent, if one fails the others can continue executing. In a properly designed system, the others reorganize themselves to continue providing a service. This idea is used by the Erlang language, which is used in telecommunications and high-speed networking (see section 5.7).
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	<ul style="list-style-type: none"> <li>✓ highly 08015</li> <li>✓ reliable 1</li> <li>✓ reorganize 0.9118</li> <li>✓ providing 0.9190</li> <li>✓ service 0.8236</li> <li>✓ telecommunications 0.9118</li> </ul>
	RCI Ranking (in post-edition)	
Rank 1 <sup>st</sup> :  <i>rci</i> = 0.3974 <i>mi</i> = 1 <i>ki</i> = 0.1667 <i>si</i> = 0.6822	Text	Section 5.6 gives an introduction to erlang a programming language based on port objects. erlang is designed for and used in telecommunications applications where fine grained concurrency and robustness are important.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	✓ telecommunications 0.9118

The annotation persistence model makes the right decision to orphan annotations for this paragraph of text.

Paragraph 5: system suggests users to decide

Annotation anchor text in pre-edition  RCI Ranking (in post-edition)	Text	We define a computation model for message passing as an extension of the declarative concurrent model. We then use this model to show how to program with message passing.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)
Rank 1 <sup>st</sup> :  <i>rci</i> = 0.6194 <i>mi</i> = 1 <i>ki</i> = NaN <i>si</i> = 0.4622	Text	Section 5.7 explains one advanced topic the nondeterministic concurrent model which is intermediate in expressiveness between the declarative concurrent model and the message passing model of this chapter.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)
Rank 2 <sup>nd</sup>  <i>rci</i> = 0.4795 <i>mi</i> = 1 <i>ki</i> = NaN <i>si</i> = 0.6320	Text	The extended model is called the message passing concurrent model. Since this model is nondeterministic it is no longer declarative. A client server program can give different results on different executions because the order of client sends is not determined.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)

Since this paragraph is deleted from pre-edition, after examine the post-edition, we decide any annotation made against this paragraph should be orphaned. The annotation persistence system fails to orphan an annotation in this case, but rather chooses to let the user to decide. This is labeled as a misclassification.

## Paragraph 6: annotation reattached

Annotation anchor text in pre-edition	Text	The declarative concurrent model of the last chapter cannot have observable nondeterminism. This limits the kinds of programs we can write in the model. For example, we saw that it is impossible to write a client/server program where the server does not know which client will send it the next message.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)
RCI Ranking (in post-edition)		
Rank 1 <sup>st</sup> :  <i>rci</i> = 0.7092 <i>mi</i> = 1 <i>ki</i> = NaN <i>si</i> = 0.3531	Text	In the last chapter we saw how to program with stream objects which is both declarative and concurrent. But it has the limitation that it cannot handle observable nondeterminism. For example we wrote a digital logic simulator in which each stream object knows exactly which object will send it the next message. We cannot program a client server where the server does not know which client will send it the next message.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)

System correctly predicted that annotation should be repositioned in the newer version in this case.

## Paragraph 7: annotation reattached

Annotation anchor text in pre-edition	Text	The message-passing concurrent model extends the declarative concurrent model by adding just one new concept, an asynchronous communication channel. This means that any client can send messages to the channel at any time and the server can read all the messages from the channel. This removes the limitation on what kinds of programs we can write. A client/server program can give different results on different executions because the order of client sends is not determined. This means that the message-passing model is nondeterministic and therefore no longer declarative.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)
RCI Ranking (in post-edition)		
Rank 1 <sup>st</sup> :  <i>rci</i> = 0.7365 <i>mi</i> = 1 <i>ki</i> = NaN <i>si</i> = 0.3200	Text	The extended model is called the message passing concurrent model. Since this model is nondeterministic it is no longer declarative. A client server program can give different results on different executions because the order of client sends is not determined.
	MetaInfo	▪ chapter 5 message passing concurrency
	Keywords (1-entropy)	(no keyword found)

This is an interesting case. The study of this case brings up a constraint we have imposed in our model implementation. If the constraint is relaxed, we think it represents an immediate improvement we can make to our model design.

When we study the corresponding content in pre-edition and post-edition versions, we find that the initial paragraph in the pre-edition was modified in several ways. It not only went through rewording, but it was also decomposed into two contiguous paragraphs in post-edition,

*The message-passing concurrent model extends the declarative concurrent model by adding just one new concept, an asynchronous communication channel. This means that any client can send messages to the channel at any time and the server can read all the messages from the channel. This removes the limitation on what kinds of programs we can write. A client/server program can give different results on different executions because the order of client sends is not determined. This means that the message-passing model is nondeterministic and therefore no longer declarative.*

becomes the following in post-edition,

*We can remove this limitation by extending the model with an asynchronous communication channel. Then any client can send messages to the channel and the server can read them from the channel. We use a simple kind of channel called a port that has an associated stream. Sending a message to the port causes the message to appear on the port's stream.*

*The extended model is called the message passing concurrent model. Since this model is nondeterministic it is no longer declarative. A client server program can give different results on different executions because the order of client sends is not determined.*

Although the model picked up one of the paragraph and suggested repositioning annotations to it, it nevertheless reveals one of the problems existing in the design and implementation of the annotation persistence model.

Since the system selects candidate anchor by single paragraphs, if the document modifications include separation of a large paragraph into several smaller paragraphs, the chances that the model will misclassify the case becomes high.

Through out the model evaluation, we have taken single paragraph of text as a unit of operation, both in textbook latent semantic analysis and annotation's original/candidate anchor selection.

It makes perfect sense in LSA to build the latent semantic space by segmenting document by its natural paragraph boundary as long as the most paragraphs contain enough semantic information. If the most of the paragraphs after segmentation are very short (e.g. a very short sentence), the LSA space built on them may not be a good latent semantic space representation. After latent semantic space is built, however, any two texts (regardless if they are from the paragraphs with which LSA space is built upon) can

be compared semantically. In our case, we can perform semantic comparisons between any pair of single paragraphs, or any pair of groups of paragraphs.

For annotation's original anchor text selection, by allowing only single paragraph selected for original anchor, we are making a constraint on users which disallows them to annotate multiple continuous paragraphs.

For annotation's candidate anchor text selection, by choosing only one paragraph, we are ignoring the possibility that a combination of multiple continuous paragraphs maybe more similar to the annotation's original anchor text.

The problem can be solved by introducing a dynamic paragraph grouping scheme.

We propose the following:

- First, after the first run of the annotation persistence scheme, we identify the candidate anchor locations with every candidate text being a single paragraph. We rank them.
- Second, for each candidate anchor, the upper and lower neighboring paragraphs (need to be continuous though) are combined to find the collective groups (again need to span continuously in the newer version) with the highest *rci* value
- Third, re-rank the list.
- Fourth, follow the guidelines to make annotation reposition decisions.

We are running short to implement this automatic optimization idea to the annotation persistence scheme. However, we performed a manual test on the above idea by combining the paragraph before the selected paragraph with the selected paragraph from above table to form a single text. We then measure the *rci* of between this combined text with the query document. The *rci* came in as 0.7347 (which is very close to 0.7365).

If we follow the above strategy, we still chose the uncombined solution as our reattached annotation anchors. The dynamic paragraph grouping scheme does not change the reattachment decisions. However, we think the dynamic paragraph grouping scheme will be effective on the following cases:

1. Original anchor contains multiple paragraphs, thus candidate may contain multiple paragraphs as well.

2. if the original anchor text is decomposed into many smaller continuous paragraphs.

The effect of dynamic grouping on the performance of the scheme should be studied in the future.



## Paragraph 8: Annotation reattached

Annotation anchor text in pre-edition          RCI Ranking (in post-edition)	Text	<p>We use a simple kind of channel called a port that has an associated stream. Sending a message to the port causes the message to appear on the port's stream. A useful programming technique is to <u>associate</u> a port with a stream object. We call the resulting entity a port object. A port object reads all its messages from the port's stream, and sends messages to other port objects through their ports. Each port object is defined by a recursive procedure that is declarative. This keeps some of the advantages of the declarative model.</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 5 message passing concurrency</li> </ul>
	Keywords (1-entropy)	<p style="text-align: center;">✓ associate                      0.867736096909849</p>
Rank 1 <sup>st</sup> :    <i>rci</i> = 0.7760 <i>mi</i> = 1 <i>ki</i> = 1 <i>si</i> = 0.3200	Text	<p>A useful programming style for this model is to <u>associate</u> a port to each stream object. The object reads all its messages from the port and sends messages to other stream objects through their ports. This style keeps most of the advantages of the declarative model. Each stream object is defined by a recursive procedure that is declarative.</p>
	MetaInfo	<ul style="list-style-type: none"> <li>▪ chapter 5 message passing concurrency</li> </ul>
	Keywords (1-entropy)	<p style="text-align: center;">✓ associate</p>

The system correctly predicted where annotations should be repositioned in the newer version.

## Results Summary

The following table presents a summary of the model predictions compared to human perceptions.

In the whole text, we identified a total of 89 cases of minor modifications; we expect the annotation persistence scheme to classify them as candidates for repositioning with a high degree of confidence.

We also identified total 36 deletions and heavy modifications; we expect the annotation persistence scheme to classify them as candidates for orphaning annotations.

There are 8 cases, where we think the modification is in the moderate range, and we expect the annotation persistence scheme to classify them as cases for users to choose.

Here are the comparisons,

Human perceptions		Model predictions		
		Reattach annotations	Suggest users to decide	Orphaned annotations
Light modifications	89	82 (92%)	7 (8%)	
Moderate modifications	8		6 (75%)	2 (25%)
Heavy modifications	36		5 (13%)	31 (86%)

**Table 4.3** Comparison of model predictions of annotation persistence decisions and human perceptions of the document modifications

Table 4.3 provides convincing evidence that the model predicts most of the annotation persistence decisions correctly based on human perceptions of the degree of annotation text modification.

## Discussion of the Results and the Model

In most of the moderate level text modification cases and several cases where only light modifications are applied to pre-edition text, the pre-edition texts are modified in such a way that its contents are expanded and the pre-edition paragraph is decomposed into several smaller paragraphs in the newer version. The similarity of the pre-edition paragraph with each smaller post-edition paragraph is lessened. We believe this is the cause for many misclassifications. Implementing the dynamic paragraph grouping scheme is foreseeable capable to improve the prediction capability of the annotation persistence scheme in this scenario.

In the evaluation of model effectiveness, we recognize that since each person may agree/disagree on the degrees of the text modifications, his/her decision on whether the annotations should be reattached/orphaned/suggested-for-users-to-decide may be different. This suggests that we may have introduced bias into our final results (Table 4.3). To eliminate human bias, clearly an evaluation of the system by a large number of readers is preferred in order to evaluate the effectiveness of the system.

The design of the annotation persistence scheme, especially the calibration of the *rci* and the *rci* thresholds to determine the annotation repositioning decisions needs further discussion. Since we use only a small test example to calibrate the contributions of each location descriptor and *rci* threshold, its effectiveness is in question, especially in the boundary cases.

When making decisions on what to do with annotations based on *rci* values, we paid attention only to the candidate anchor with the highest *rci* value. It is often the case that the distribution of *rci* values (also the component indices, *mi*, *ki* and *si*) tells a great deal about the decision on annotation repositions. What if there are more than one *rci* values which are bigger than upper *rci* index? Clearly, this represents one subject of future study to improve the model selection decisions.

# Chapter 5

## Conclusion

---

### 5.1 Thesis Summary

Annotation persistence is a vital component when designing a digital annotation system. In Chapter One, we present an overview the state-of-art research on annotation system design and annotation persistence methodologies.

We started by reviewing the annotation taxonomy of annotation forms and functions. We adopted Marshall's view that annotation can be either implicit or explicit based on its content, while its location can be within-text or marginal. We defined a generic representation of digital annotation, where each annotation object is composed of its "content" and "anchor". Among them, "anchor" contains location reference information which is used to address into a document. In this section, we also identified the benefits digitization brings to annotation, as well as the challenges digital annotations face. We elaborated on one of the challenges *annotation persistence over dynamic documents* which is the focus of this study. A large-scale annotation software study conducted in Microsoft indicated that the inability to reattach annotations once a document changes can be costly, and in fact, it was the primary reason people stopped using the entire system.

In the second section of the Chapter One, we reviewed the state-of-art research on annotation systems and architecture. We elaborated on a few annotation systems developed in industry and at research institutes that represent the forefront of research in annotation system development.

In the third section of the Chapter One, we concluded that annotation persistence over document versions is a complicated and challenging problem, as documents can go through various types of changes among versions. Before we reviewed the state-of-art

annotation re-anchoring methods, we defined robust criteria for the annotation re-anchoring mechanism. We concluded in our review that none of the current mechanisms proposed in the literature fully satisfies our robust criteria.

In the last part of Chapter One, we revisited our problem and stated that annotation persistence over dynamic documents can be formulated as a specialized information retrieval problem. We designed the annotation's anchor to include three location descriptors to capture three important characteristics of the annotation's anchor information:

- *Meta-structure information location descriptor*
- *Keyword location descriptor*
- *Semantic concept location descriptor*

To meet the annotation persistence robust criteria, we require the reattachment algorithm design to follow the following design guideline; in the case when there are only minor modifications made to an annotation's anchor text and surrounding context, the reattachment algorithm should pick the right location in the revised document with high confidence; in the case when there are increasing changes to the documents, the reattachment algorithm should present a list of possible answers with a ranking by confidence scores; in the case when the document changes are radical, the reattachment algorithm should orphan the annotations rather than reattach them with low confidence.

In Chapter Two, we started with an evaluation of an important nature language phenomenon, Zipf's Law. We then investigated Luhn's theory on word's cut-offs. Luhn proposed two cut-offs to Zipf's curve. The ranked word frequency spectra are then separated into three groups, common words, rare words and words in the middle of the spectra which contain strong content discriminating abilities.

Common words are also called stopwords in IR. As common words don't carry the semantics of the contents, a stopword list is usually used in document indexing to eliminate all common words. This helps to significantly reduce the space overhead of indices for natural language texts.

Luhn stated that words in the middle range of the spectra of the ranked word list have strong resolving power; they are strong contributors to the content and semantics of

documents. Because of their significant semantic load and relatively rich occurrences in the document corpus, they should be used exclusively to compare semantics.

Rare words compose a large portion of the word vocabulary. We claim infrequent words have strong indexing power, i.e. the ability for words to resolve document locations with strong certainty.

We use entropy to measure a word's indexing power. Entropy values are normalized such that all word entropy falls within  $[0 - 1]$ . *Normalized Word Entropy* measures the word distributions for each word in a text corpus. The more uniformly a word is distributed in the corpus, the higher the *Normalized Word Entropy* is, and the closer the entropy value approaches 1. The more skewed the word's distribution in the corpus, the lower the *Normalized Word Entropy* is. In the case when a word appears only once in a text corpus, the *Normalized Word Entropy* of this word is 0.

The smaller the *Normalized Word Entropy*, the larger the word indexing power. Thus words with very low normalized word entropy are better keywords and can be used to resolve the parent document locations.

In Chapter Three, we reviewed and evaluated Latent Semantic Analysis. Two well-known language phenomena which plague the information retrieval performance of lexical word matching are synonymy and polysemy. Synonymy means that an object can be referred in many ways, i.e. people use different words to represent the same semantic subject. Polysemy is the problem of a word having more than one specific meaning. LSA offers a dampening effect on synonyms, though the effect on polysemy is less pronounced.

LSA assumes there exist implicit higher semantic structures in term-document associations. Terms tend to be similar if they appear in the same kind of documents, whether or not they actually occur within identical word contexts in those documents. Documents are semantically close if they have many similar words in common, and semantically distant if they have few words in common.

LSA starts with the construction of the term-document matrix, with each cell representing the co-occurrences between words and documents. In this section, we reviewed the operations which are normally performed in IR to prepare a text, such as lexical analysis of text, treating stopwords, stemming and selection of terms for term-

document matrix. We presented the global and local weighting strategies for each cell of the term-document matrix.

The central component of LSA is the singular value decomposition of the term-document matrix. The SVD reveals important latent semantic structure by decomposing the term-document matrix into three sets of matrices, a left orthogonal matrix, a right orthogonal matrix and a diagonal matrix. An important step in LSA is the dimension reduction after SVD. By dimension reduction, LSA is able to extract the major latent semantic structure, and at the same time to eliminate noise and unreliability of word usages.

The term-document matrix is a large sparse matrix. In this chapter, we discussed numerical solutions to decompose a large sparse matrix.

LSA has been an active research subject. In this chapter, we reviewed the many applications to which LSA has been applied. Since LSA is a completely automatic method, it has been applied to a wide range of problems.

In the last part of the chapter, we evaluated LSA against a text corpus of three years of Wall Street Journal articles. The text corpus in this study is not a calibrated text collection, so it is impossible to measure the precision and recall performance of the LSA retrieval system. Still, the evaluation against the text corpus allows us to evaluate three things, 1) the tools developed in this study; 2) the computation cost of LSA; 3) the sample retrieval results and overall effectiveness of the LSA system.

The retrieval results by LSA present us an opportunity to see its power of retrieving words and terms that are semantically close. In word retrieval, we picked words in three different semantic concept spaces, namely financial, medical and political. The results clearly showed the ability of LSA to cluster words based on their semantic meanings. In document retrieval, the results showed LSA is able to pick up documents which are semantically similar, though they might not share any common words.

In Chapter Four, we completed the design of the robust annotation persistence scheme. We finalized the anchor representation design by answering three questions, how meta-structure information is parsed, how the keyword threshold is determined, and how to quantify the semantic closeness between documents into a continuous numeric value ranging from 0 to 1. We went on further to define the three indexes ( $mi$ ,  $ki$ , and  $si$ ) to

represent the quantitative matching of three location descriptors between annotation anchors. *mi* measures the match of the meta-structure information between annotation anchors. *ki* measure the percentage of keywords contained in the original annotation anchor left in candidate annotation anchors. *si* measures the semantic similarities between annotation anchors. The reattachment confidence index, which is a linear scoring index with contributions from the three individual indexes, is used to measure the closeness between original annotation anchors and candidate anchors.

We finished the design of the reattachment algorithm by declaring two RCI threshold values. The upper threshold value represent the threshold based on which the annotation persistence scheme will decide if the annotation should be reattached with a high degree of confidence. The lower threshold value represent the threshold based on which the annotation persistence scheme will orphan annotations.

A textbook with pre-edition and post-edition versions was used to calibrate and evaluate the annotation persistence model. We calibrated the model with the data from portions of the textbook and went on to evaluate the model with the rest of the data from the textbook.

In the evaluation process of the textbook with versions, we identify the modifications of the text among versions. We mark the texts in the pre-edition which are subsequently modified in the post-edition into one of the three categories, lightly modified, moderately modified and heavily modified. We assume that degrees of the text modifications coincide with the three decisions we can make on annotation reattachment, 1) whether annotations should be reattached with high degree of confidence; 2) whether candidate anchors should be left for users to decide; or 3) whether the annotation should be orphaned.

The results of the evaluation of the model showed us that the model is very effective in reattaching annotations with a low level of misclassification errors. Among the cases where we find only light modifications are made to the texts, the model predicts that, for 92% cases, annotations can be reattached with high degree of confidence. Among the cases where heavy modifications are made to the text, the model predicts 86% of cases where annotations should be orphaned.



## **5.2 Future Research**

In concluding this thesis, in this section, we would like to point out the improvements we can make to the proposed annotation persistence scheme. We address the need for mass evaluation of the model. We also explore the applications of the model to the World Wide Web.

### **Dynamic Paragraph Grouping**

In the last section of Chapter Four, we elaborated the need and the strategy of dynamic paragraph grouping when making annotation persistence operations. The strategy of dynamic paragraph grouping will improve the performance of the model in the following two scenarios: 1) if users make annotations on multiple continuously distributed neighboring paragraphs; 2) if the original annotation anchor text is decomposed into many smaller continuously distributed paragraphs. Dynamic paragraph grouping is an easy step to add to the current annotation persistence mechanism.

### **Model Parameter Calibrations**

A thorough study on the selection of the parameters of the model is called. How do we select the entropy cut-off point to select keywords? Are there any automatic and better ways of selecting *rci* index coefficients during reattachment confidence index calibration? Can we classify annotation decisions based on the distribution of the *rci* values, rather than using two fixed upper and lower threshold values?

### **Annotation Multi-Referencing**

One of the advantages of the digital form brings to digital annotation is the digital annotation's capability of having multi-references. Annotations can point to multiple locations in a document, or even multiple locations in multiple documents. The mechanism proposed in this study can be applied directly to this scenario as well.

### **Mass Evaluation**

Mass evaluation is required to measure the effectiveness of the proposed annotation persistence scheme. In the evaluation of model effectiveness, we recognize that since

each person may agree/disagree on the degrees of the text modifications, his/her decision on whether the annotations should be reattached/orphaned/suggested-for-users-to-decide may be different. This suggests that we may have introduced bias into our final result summary. To eliminate human bias, clearly an evaluation of the system by a large number of readers is preferable in order to evaluate the effectiveness of the system.

### **Application to Web Documents**

One of the goals of developing annotation software is to allow users to annotate web documents. The annotation persistence scheme proposed in this study is only applicable if domain knowledge about the document can be obtained. Although it is difficult to get the information directly for web documents, there are some ways to infer that information. For example, we can check out the key index words and derive the domain of the knowledge by examining the collective keywords. We can look up the domain or the URL which may contain useful information to derive domain knowledge as well. Metadata for web documents continues to evolve and web annotation represents a major research direction.

## Appendix

### **Sample Document Retrieval Results of LSA**

---

The average length of each document in the corpus of the three years of Wall Street Journal articles is 245 words. The lengths of all documents vary considerably. There are documents that contain only one or two sentences as well as documents that are considerably long. In the following we randomly pick three query documents with varying lengths and retrieve the top five semantically mostly similar documents from the corpus.

The query document in Table A.1 is about the rise of the British industrial production rate. The first three retrieved documents are all about British industrial production rate as well. The fourth ranked and the fifth ranked documents are about different economic indicators from different countries. Their semantic distances are further from the top three ranked documents.

Query document	seasonally adjusted british industrial production increased one point five percent in january from a year earlier and zero point four percent from december the central statistical office said .
Retrieved document	
Rank 1 (14.31 <sup>0</sup> )	british industrial production rose three point seven percent in november from a year earlier but declined zero point three percent from october the central statistical office said in a provisional report . output by manufacturing industries alone grew five percent from the year before but fell zero point five percent from october
Rank 2 (15.22 <sup>0</sup> )	british industrial production rose four point seven percent in june from a year earlier but fell zero point nine percent from may the central statistical office said in a preliminary report .
Rank 3 (15.67 <sup>0</sup> )	british industrial production in july rose one point eight percent from june but was down zero point five percent from a year earlier according to provisional data released by the united kingdom central statistical office .
Rank 4 (16.16 <sup>0</sup> )	swiss consumer prices rose one point six percent in january from a year earlier and zero point three percent from december the government said .
Rank 5 (16.31 <sup>0</sup> )	wholesale prices in switzerland rose two point two percent in july from a year earlier and zero point one percent from june the government said .

**Table A.1** LSA query result of a document about “British industrial production”

In Table A.2, it is clear that the top ranked retrieval document is semantically very close the query document. The rest of the documents, however, are all about common stock buybacks, but they are talking about different companies.

<p><b>Query document</b></p> <p><b>Retrieved document</b></p>	<p>norwest corporation a minneapolis bank holding company said its board authorized buying back from time to time as many as two million common shares .</p> <p>norwest said the repurchased shares will be used to meet periodic stock issuance requirements under the company's employee savings and investment plan . they also will be used for employee stock options a shareholder dividend reinvestment plan conversions of convertible debentures and for other corporate purposes .</p> <p>norwest said it has about forty four point five million common shares outstanding . the buy back represents four point five percent of the total .</p>
<p>Rank 1 (20.71<sup>0</sup>)</p>	<p>norwest corporation said it plans to buy as many as nine hundred thousand shares of its common stock from time to time in the open market and private transactions .</p> <p>the bank holding company said the shares will be used for its employee savings and investment plan for conversions of convertible debentures and for other corporate purposes .</p> <p>at june thirtieth norwest had thirty point three million common shares outstanding .</p>
<p>Rank 2 (23.44<sup>0</sup>)</p>	<p>the holding company for eastchester savings bank said it will distribute rights to buy preferred stock to common stock of record october sixteenth . under certain circumstances the rights would allow holders to buy common shares of eastchester or its buyer at half price . the company currently has four million shares outstanding .</p>
<p>Rank 3 (24.13<sup>0</sup>)</p>	<p>puget sound bancorp said its directors have authorized the repurchase of as many as five hundred thousand of the company's fifteen point nine million common shares outstanding .</p> <p>the bank holding company said it expects to use the shares for an employee stock purchase and stock option plan .</p>
<p>Rank 4 (24.15<sup>0</sup>)</p>	<p>national city corporation said its directors approved the repurchase of one million of its common shares .</p> <p>the bank holding company said the stock will be held as treasury stock until it is reissued for stock option plans or in connection with continuing conversion of national city's eleven point two five percent convertible subordinated debentures</p> <p>national city has forty point nine million common shares outstanding . about three hundred thousand shares were issued</p>

	<p>last year under stock option programs for senior and middle management .</p> <p>separately shareholders voted to triple the number of authorized but unissued common shares to one hundred fifty million . national city has about fourteen billion dollars in assets .</p>
<p>Rank 5 (25.00<sup>0</sup>)</p>	<p>wesbanco incorporated said its board approved the repurchase of as many as eighteen thousand of its common shares .</p> <p>the bank holding company said the stock would be used for the company's employee stock ownership plan possible acquisitions and other corporate purposes . wesbanco has about two million shares outstanding .</p>

**Table A.2** LSA query result of a document about “common stock buyback”

Table A.3's query document is a very long document. It is about Du Pont, one of its chemical product c.f.c.s., and the fact that Senate's trying to stop Du Pont from producing the environmentally hazardous product. Documents ranked from 1 to 5 show the clear semantic closeness decay from the query document.

<p><b>Query document</b></p>	<p>newspaper ads run by du pont company in the mid nineteen seventies have come back to nag the chemical giant .</p> <p>three environmentalist u. s. senators have dredged up the ads in which du pont promised it would stop producing chemicals known as chlorofluorocarbons if they were found to harm the environment .</p> <p>more than a decade of scientific studies later thirty one countries have signed a treaty declaring chlorofluorocarbons known as c. f. c.s a menace to the earth's protective ozone layer . and du pont is still the biggest u. s. producer of c. f. c.s .</p> <p>the time has arrived for the du pont company to fulfill that pledge wrote senators . max baucus d. montana robert stafford r. vermont and david durenberger r. minnesota to du pont chairman richard heckert .</p> <p>the lawmakers suggested du pont stop production of c. f. c.s within a year because they deplete the ozone layer that shields the earth from the sun's ultraviolet rays which can cause skin cancer and other environmental damage . but du pont's mr. heckert wrote back refusing saying that would be unwarranted and counterproductive and more drastic than scientific evidence justifies .</p> <p>in its ads and separately in testimony on capitol hill du pont had promised should reputable evidence show that some fluorocarbons cause a health hazard through depletion of the ozone layer we are prepared to stop production of the offending compounds . since then the u. s. has banned use of most c. f. c.s which also have been called fluorocarbons in aerosol sprays . c. f. c.s also are blamed for a seasonal hole each year in the ozone layer over antarctica .</p> <p>the senate plans to vote next week to ratify the international pact to freeze and then to roll back by fifty percent world production of c. f. c.s by mid nineteen ninety nine . c. f. c.s developed by du pont in the nineteen thirties are widely used as cooling agents in air conditioners and in refrigerators and in making plastic foams and computer cleaning solvents</p> <p>wilmington delaware based du pont supports the treaty but argues that at the moment scientific evidence does not point to the need for dramatic c. f. c. emission reductions such as a</p>
------------------------------	--

<p><b>Retrieved document</b></p>	<p>complete stop of production .</p> <p>steve seidel a senior analyst with the environmental protection agency says there is scientific consensus that chlorine from c. f. c.s attacks and destroys ozone molecules in the upper atmosphere . however he said there is dispute about how drastically c. f. c.s must be reduced to stem dangerous levels of ozone depletion .</p>
<p>Rank 1 (16.63<sup>0</sup>)</p>	<p>du pont company said it plans to phase out production of environmentally harmful chlorofluorocarbons in a move that could pressure other producers and nations to help stop the destruction of the earth's protective ozone layer .</p> <p>the chemicals and energy concern didn't specify when it would end production . but it indicated the phase out and substitution of environmentally safer products would take several years and require world wide cooperation .</p> <p>wilmington delaware based du pont adopted the new policy after analyzing data compiled by an international scientific panel that linked chlorofluorocarbons or c. f. c.s to the ozone depletion problem .</p> <p>du pont which estimates that it makes twenty five percent of the world's c. f. c.s markets the compound under the trademark freon . the company said its goal is an orderly phase out of fully halogenated c. f. c. production coupled with the introduction of alternative chemicals and technologies . fully halogenated c. f. c.s contain chlorine which has been found to destroy the ozone layer .</p> <p>c. f. c.s are used as cooling agents in refrigerators and air conditioners as cleaning agents and in making plastic foam . substitutes for most major applications haven't been adopted . c. f. c.s had been widely used as propellants in aerosol containers but the u. s. banned that practice in the nineteen seventies after c. f. c.s were found to destroy ozone molecules in the upper atmosphere .</p> <p>ozone depletion is a major environmental problem . the ozone layer screens out harmful ultraviolet rays that can cause skin cancer eye ailments and other health problems as well as environmental damage .</p> <p>analysts said du pont's withdrawal from the industry could boost prices of products that use c. f. c.s but would have little effect on the chemical giant . the company said c. f. c.s accounted for less than two percent of earnings and sales last year . that means its c. f. c. sales totaled about six hundred million dollars contributing as much as thirty five million dollars to du pont's nineteen eighty seven profit of one point seven nine billion dollars .</p>



allied signal incorporated the second largest u. s. producer of c. f. c.s said it is waiting for the national aeronautics and space administration to release the complete international study on which du pont based its decision . a du pont scientist was on the study panel but the company said it based its analysis on an executive summary of the data released earlier this month . in the meantime morristown n. j. based allied signal isn't taking any steps to curtail its c. f. c. production though it is working on development of substitutes .

though analysts saw little effect from du pont's gradual withdrawal from c. f. c. production the news apparently sparked a sell off . du pont shares fell three point one two five dollars to close at eighty two dollars and fifty cents in new york stock exchange composite trading yesterday . the issue was the biggest loser among the dow jones industrials and one of the ten biggest decliners overall . allied signal shares also slid closing at thirty two dollars down one point one two five dollars in big board trading .

referring to the ozone depletion problem we take this very seriously an allied signal spokesman said and we are in agreement with du pont that any solution is going to take an international approach .

pennwalt corporation the nation's third largest producer of c. f. c.s also called the ozone depletion issue a global problem and called for a world wide end to c. f. c. production as soon as practical . but the philadelphia based company didn't indicate it is taking any unilateral steps to curb production .

joseph p. glas director of du pont's freon products division said the company hopes to drum up support for an international treaty known as the montreal protocol which calls for reductions in c. f. c. use . the treaty was reached last september and signed by thirty one nations but it can't take effect until at least eleven countries representing two thirds of the world's c. f. c. production ratify it . so far only mexico and the u. s. have done so .

mr. glas said his vision is to stick to the timetable set in the protocol which would reduce c. f. c. production twenty percent by nineteen ninety three and then very quickly start stepping it down to a ninety five percent reduction by two thousand three . the treaty uses nineteen eighty six as a base year and would reduce production by only fifty percent . the protocol has come under criticism from environmentalists as too lenient .

in a rare show of support environmental groups generally applauded du pont's action . this is an excellent example of corporate environmental leadership that ought to be emulated world wide said daniel j. dudek senior economist with the

	<p>environmental defense fund . this really is a breakthrough . obviously du pont has been a large part of the problem but du pont surprised everyone and the policy shake up is going to be a large part of the solution .</p> <p>geoffrey webb international director of friends of the earth said i think there could be a domino effect world wide . the industry has tried to band together to put up a common front and du pont's action could break the log jam . nevertheless he criticized the company for not establishing a firm timetable .</p> <p>du pont's action reverses its previously reported stand made earlier this month when three environmentalist u. s. senators dredged up newspaper ads that du pont ran in the mid nineteen seventies . in the ads and separately in testimony on capitol hill du pont had promised to stop producing c. f. c.s if they were found to harm the environment . the three senators citing the treaty declaring the chemicals a menace called on du pont to fulfill its pledge . du pont chairman richard heckert wrote to the lawmakers saying their suggestion to stop c. f. c. production within a year would be unwarranted and counterproductive and more drastic than scientific evidence justifies .</p> <p>du pont's phase out plan some analysts said will mean that prices of some consumer and industrial products that use c. f. c.s are likely to rise because substitutes will be more expensive and new designs may be needed for products such as compressors for air conditioners .</p> <p>john henry an analyst at shearson lehman hutton incorporated predicted du pont's decision ultimately could boost its earnings . they will be introducing new products and are entitled to a higher profit margin he said .</p>
<p>Rank 2 (16.990)</p>	<p>saving the earth's protective ozone layer isn't going to be an easy job . du pont company's acknowledgment last week that the ozone is at risk has focused attention on the severity and urgency of the problem . du pont's solution is simple stop making the chemicals that are believed to be the culprit . but the world has become so heavily dependent on those chemicals known as chlorofluorocarbons or c. f. c.s that accomplishing the goal will be risky and costly .</p> <p>the task will require world wide cooperation and good substitutes . it will cost hundreds of millions of dollars in research and plant construction . moreover weaning the world too quickly from c. f. c.s could eliminate tens of thousands of jobs trigger bankruptcies and even cause new health risks .</p> <p>du pont has certainly sent a strong signal to those who thought they'd wait it out and not do anything acknowledges kevin fay executive director of the alliance for responsible c. f. c.</p>

policy which represents producers and users of c. f. c.s . but how fast can we do it .

adds karl loos vice president of chemicals and plastics at arthur d. little a boston consulting firm we're in turmoil . we're all looking for the answers and they aren't clear right now

invented in the nineteen thirties c. f. c.s make refrigerators and air conditioners produce cool air . they're used in plastic foam and cleaning agents . they were once used as propellants in aerosol containers until the practice was banned in the u. s. in the late nineteen seventies . without c. f. c.s food would spoil office workers would wilt and cars would be less comfortable . in the u. s. alone c. f. c.s represent a twenty eight billion dollar industry that employs about seven hundred fifteen thousand people in five thousand companies .

it wasn't until nineteen seventy four that scientists raised the possibility that c. f. c.s might be eating away at the ozone the layer of stratosphere that screens out the sun's harmful ultraviolet rays . the rays can cause skin cancer eye ailments and other health problems as well as environmental damage to crops and fish populations .

c. f. c. producers du pont is the largest of the five major u. s. suppliers began back then looking for substitutes . the research effort today involves hundreds of scientists . du pont estimates it has invested about thirty million dollars so far including ten million dollars last year . it expects to spend even more this year . though other companies are spending somewhat less the problem has been given top priority in the industry officials say the stakes are high . companies may invest hundreds of millions of dollars in new plant construction to make substitute products so they want to be sure the substitutes aren't toxic and don't fail to serve the purpose . early on for example du pont thought it had come up with a good substitute for cleaning electronic equipment and then discovered the compound caused sterility in male rats .

products may be obsolete by the time they are ready to market . such was the case in the late nineteen seventies when pennwalt corporation developed a replacement for c. f. c.s in aerosol cans . by the time the company had finished testing the product for toxicity aerosol users had moved on and decided to use a hydrogen compound instead .

we were left with an approved product and no market says peter miller manager of pennwalt's isotron division .

there also may be production problems . at allied signal incorporated the second largest c. f. c. producer in the u. s. bernard sukornick director of fluorocarbon research cites the risks of investing in commercial plant production for a new

product before it is known from pilot studies whether the final manufacturing process is indeed adequate .

we're taking considerable risks and shortcuts says mr. sukornick we're risking much more from a financial point of view than this type of industry usually takes .

in the rush to find substitutes it is difficult to assess whether a product is the best possible outcome . as a scientist i can't stand here and tell you that i have gone through every combination says mr. sukornick . i know i haven't looked at the problem in enough ways to be satisfied intellectually but i have no time . i have to take what i've got and go with it .

for manufacturers that use c. f. c.s in their products the risks aren't as easy to define . they are potentially devastating if the process moves too quickly . under the worst circumstances producers would stop making c. f. c.s before substitutes are available . shortages would develop prices would skyrocket and manufacturers of appliances such as refrigerators would go bankrupt .

du pont's plan to cease c. f. c. production has heightened anxieties . although the company promises an orderly transition to the total phase out it could stop making c. f. c.s at any time . if others followed and substitutes weren't available shortages could develop that would put some customers out of business .

in addition the du pont announcement runs the risk of encouraging congress to legislate restrictions that the industry would find unpalatable according to industry officials .

if du pont says we choose to get out of this du pont goes on says mr. fay of the c. f. c. alliance . but g. e. can't just say we won't make refrigerators . it's a much tougher position for user industries . adds arnold braswell president of the air conditioning and refrigeration institute which represents manufacturers we're very nervous .

yet another challenge is cooperation . while du pont has boldly stated its intentions to eventually cease production of c. f. c.s it hasn't said when it will do so no other producer has yet jumped on the band wagon . du pont estimates that it makes twenty five percent of the world's c. f. c.s but recognizes that it can't solve the world's problem alone .

indeed du pont's plan goes significantly beyond the montreal treaty signed by thirty one countries last fall . the treaty calls for a fifty percent reduction in nineteen eighty six levels of c. f. c. production by nineteen ninety eight but not a total phaseout .

for c. f. c. users the dilemma is different . richard barnett chairman of the c. f. c. alliance says equipment manufacturers won't retool their plants until they can know with certainty which c. f. c. substitutes the producers will stick with . when

	<p>these customers retool he says it has got to be right . they can't afford to change two or three times .</p>
<p>Rank 3 (17.23<sup>0</sup>)</p>	<p>the u. s. by an eighty three to zero vote in the senate became the first major producer and consumer of ozone depleting chemicals to ratify a treaty limiting their production .</p> <p>thirty one countries signed the treaty that was reached in montreal last september to curb the global use of chlorofluorocarbons also known as c. f. c.s . the senate's vote was considered crucial to winning enough support world wide to put the treaty into effect by january nineteen eighty nine .</p> <p>the treaty can't take effect unless ratified by at least eleven countries representing at least two thirds of the world's c. f. c. production . mexico is the only other country to ratify so far . japan's parliament is expected to vote before june u. s. officials said .</p> <p>c. f. c.s are widely used as cooling agents in refrigerators and air conditioners as computer cleaning solvents and in making plastic foam . after it was discovered in the mid nineteen seventies that chlorine from c. f. c.s attack and destroy ozone molecules in the upper atmosphere the u. s. canada and some scandinavian countries banned the use of c. f. c.s in most aerosols though not for other uses .</p> <p>depletion of the earth's ozone layer is of concern because the ozone screens out harmful ultraviolet rays which can cause skin cancer cataracts and environmental damage . scientists blame c. f. c.s for a seasonal hole in the ozone layer over antarctica . the national aeronautics and space administration today plans to issue new estimates on global ozone depletion .</p> <p>the treaty would hold production of the most commonly used c. f. c.s at nineteen eighty six levels . then production would be reduced twenty percent by mid nineteen ninety four and fifty percent by nineteen ninety nine . the agreement also calls for a freeze on consumption of related chemicals known as halons in nineteen ninety two .</p> <p>many environmentalists argue the treaty doesn't go far enough in restricting c. f. c.s to save the ozone layer . however the agreement will prompt the search for substitute chemicals . meanwhile prices for depleted supplies of c. f. c.s are expected to at least double .</p> <p>five u. s. companies make c. f. c.s and now sell about seven hundred fifty million dollars a year of the compounds . the compounds in turn are used in products and services that bring in billions of dollars a year according to the chemical industry . under the treaty companies making and using c. f. c.s in nineteen eighty six would be assigned quotas which could be traded . the</p>

	<p>two largest producers are du pont company and allied signal incorporated .</p> <p>the u. s. accounts for about one third of world c. f. c. production . other big producers are the soviet union japan and several european nations .</p>
<p>Rank 4 (18.35<sup>0</sup>)</p>	<p>in nineteen thirty thomas midgley junior a general motors corporation chemist stood before a scientific audience and inhaled a whiff of a new refrigerator coolant to prove its safety mr. midgley showed no ill effects . since then uses of that chemical and related substances have blossomed in products ranging from air conditioners to throwaway packaging .</p> <p>but governments world wide and thousands of u. s. companies including g. m. now face hard choices as evidence mounts that such compounds are eating away the fragile natural layer of ozone that shields the earth from ultraviolet rays . u. s. officials warned recently for instance that forty million more americans than previously expected face skin cancer over the next century if global use of ozone destroying chemicals isn't checked . substantial ozone loss is also likely to harm crucial plant and sea life and affect the world's climate many researchers believe .</p> <p>evidence of widespread chemical depletion of the ozone layer which begins eight miles above the earth's surface is far from conclusive . scientists are still debating the causes behind recent reports of ozone loss over the south pole and elsewhere . yet the risks of waiting for science to find definitive answers is too great officials of some governments say . of particular concern some say is that the huge reservoir of ozone destroying chemicals already in the atmosphere is growing and is expected to persist for decades . if we can't control these chemicals now we probably won't get a second chance says victor buxton a canadian environmental official .</p> <p>this week some forty five nations including the u. s. and canada are meeting in geneva switzerland in a bid to hammer out global limits on ozone destroying chemicals . some activists believe the process if successful could set the pattern for resolving other industrial pollution dilemmas that threaten all nations .</p> <p>getting industry to curb its appetite for these compounds however may be a problem . u. s. chemical makers aren't racing to create safer products . auto makers like g. m. may face costly plant overhauls for example . while some u. s. manufacturers are cutting down on these chemicals others aren't even when</p>

alternatives exist . and american industry uses only about thirty percent of the world's production of such compounds

chlorofluorocarbons or c. f. c.s the most pervasive ozone destroying chemicals are used as coolants in refrigerators and air conditioners as solvents in electronics manufacturing and in making plastic foam insulation and foam cups and packages .

the largest producer is du pont company which markets c. f. c. products under the brand name freon . other major producers include allied signal incorporated pennwalt corporation kaiser aluminum and chemicals corporation and racon incorporated . there are other ozone destroying compounds such as halons a chemical group used in high tech firefighting equipment .

the ozone debate began in nineteen seventy four when two university of california chemists f. sherwood roland and mario j. molina argued that c. f. c.s don't decompose in the lower atmosphere as do most other compounds . instead they theorized c. f. c.s slowly drift into the upper atmosphere where they eventually break down starting a complex chemical reaction that destroys ozone a naturally occurring form of oxygen .

reports of the chemists' work sparked a massive u. s. consumer boycott of aerosol deodorants hair sprays and similar products that depended on c. f. c.s . in nineteen seventy eight the environmental protection agency banned c. f. c.s as propellants in most aerosols resulting in a forty percent decrease in u. s. industrial demand for the chemicals .

the action however proved only a stopgap . though the propellant ban still is in effect c. f. c. sales in the u. s. have since zoomed back to pre aerosol ban levels largely sparked by the explosive growth of such products as foam throwaway packaging and increasing demand for c. f. c.s as a solvent by electronics manufacturers . researchers say it doesn't matter how c. f. c.s are used whether in a spray can or a hamburger package eventually they all are released into the atmosphere .

global sales of the compounds are also rising because among other things most european nations didn't follow the u. s. aerosol ban .

lacking easy targets such as spray cans consumer action alone isn't likely to blunt the growing use of c. f. c.s . the family car for example is filled with c. f. c. based products ranging from the coolant in its air conditioner to the padding for its seat and dashboard .

because of the compounds' pervasive use u. s. companies that eluded regulation in the nineteen seventies probably won't escape again in any tightening of regulations . they know the handwriting is on the wall says kathleen a. wolf an analyst with rand corporation a santa monica california based consulting

concern . in recent months a number of u. s. concerns such as g. m. ford motor company and international business machines corporation have formed in house task forces to study ways to reduce c. f. c. use .

the mobile air conditioner found in the vast majority of american cars is a major source of c. f. c. pollution in the u. s. c. f. c. based coolants are frequently released into the air when the systems undergo repair . while some u. s. cities such as tampa florida are moving to have bus air conditioner repairmen trap the compounds for recycling the impact of recycling on the problem is limited . engineers say about sixty five percent of a car's air conditioner coolant leaks before it ever gets to a repair shop .

auto makers say they are attacking the problem on several fronts . g. m. engineers for example are looking at tightening existing systems with new gaskets and other devices says richard klimisch the company's top environmental official . but rather than shoulder the huge financial burden of developing new types of air conditioners g. m. is relying on chemical companies to develop alternative coolants he says .

as was the case during the first ozone layer controversy battle lines are forming over who c. f. c. producers or auto makers will pay for research and development of environmentally safer compounds . g. m. officials say both du pont and the british based i. c. i. industries p. l. c. have agreed only to produce test batches of a possible ozone safe air conditioner replacement coolant known as f. c. one thirty four a .

they're interested in making f. c. one thirty four a but there are questions over who is going to pay for the toxicity testing and the types of sales guarantees they want from us mr. klimisch says . f. c. one thirty four a was first developed in the late nineteen seventies but both chemical and auto makers dropped work on the compound when government pressure to expand the scope of c. f. c. regulations eased in the early nineteen eighties .

u. s. chemical producers clearly aren't rushing to develop substitutes . morristown n. j. based allied signal for example dispatched sales teams nationwide this summer in a bid to get electronics manufacturers to switch to c. f. c.s from other solvents according to chemical distributors . an allied signal spokesman declined to comment on the matter .

du pont after disputing the c. f. c. ozone link for twelve years recently conceded that the compounds could pose a future ozone threat and called for global limits on their use . company officials say they have restarted research into f. c. one thirty four a but aren't planning heavy spending on it until regulatory action or consumer demand justify it . the principal problem with f. c. one forty three a production is the lack of a chemical catalyst to



produce the substance in commercial quantities .

u. s. chemical producers however may soon face competitive pressures . i. c. i. industries has also restarted work on f. c. one thirty four a . a japanese chemical maker diakin kogyo company and a german concern hoechst a. g. hold f. c. one thirty four a production patents says richard lagow a university of texas chemistry professor and a consultant to the e. p. a. on the c. f. c. issue . both du pont and i. c. i. officials say f. c. one thirty four a development is at least five years away .

some companies aren't waiting . at digital equipment corporation plants in andover massachusetts and salem n. h. water based systems recently replaced c. f. c.s in some electronics cleaning processes says james rogers a company environmental manager . he says the maynard massachusetts based producer of computers and computer parts has set the immediate goal of capping c. f. c. use followed by a phase out where practical . we feel the consequences of underreacting to the ozone threat are worse than the consequences of overreacting he says .

other companies could apparently do without the c. f. c.s they currently use . for instance while some mcdonald's corporation foam packages for its mcdbl hamburgers contain c. f. c.s others don't . a spokeswoman for the oak brook illinois based fast food company says c. f. c. based packages don't keep a hamburger hotter but simply reflect c. f. c. use by some of mcdonald's foam package suppliers . mcdonald's hasn't yet decided whether to switch suppliers the spokeswoman says . she adds however that the company has met with suppliers in recent weeks to discuss the issue .

world wide the pressure also is building to control c. f. c.s . a global approach is critical say some regulators and activists because an estimated seventy percent of the world's c. f. c. use occurs outside the u. s. intensified demand from developing nations is also projected in future years as their purchases of air conditioners refrigerators and other items grow .

the global freeze proposals under study at the geneva meeting this week would exploit the law of supply and demand . by capping c. f. c. production some regulators argue prices of ozone destroying chemicals would rise forcing users to seek alternatives . that would provide chemical makers with the incentive to develop safer products they say .

richard benedick the state department official heading the u. s. delegation at geneva said recently that he expects the world's nations to agree by july on the need to limit ozone destroying chemicals . after reaching a general accord nations can debate specific limits on the chemicals' uses revising them downward or

	<p>upward as scientific data build he added .</p> <p>others are less sanguine . european chemical producers for instance still advocate substantial c. f. c. growth . if the global talks stall senator john chafee r. r. i. recently said he will push for legislation restricting imports of some c. f. c. based products . the e. p. a. is also under a court ordered may deadline to decide on added c. f. c. regulations .</p> <p>science may not be able to provide policy makers with firm guidance . preliminary data from antarctica for example indicate that annual ozone loss there may result from both chemical and as yet unexplained regional and seasonal factors says robert dezafra an atmospheric physicist at the state university of new york stony brook .</p> <p>yet mr. dezafra a member of the recent u. s. scientific expedition to antarctica says he is worried . every day he says the cloud of c. f. c. in the atmosphere is building pushing the world further along a process it won't be able to reverse . says mr. dezafra we got a late start studying this problem and we're getting an even later start finding a solution .</p>
<p>Rank 5 (20.49<sup>0</sup>)</p>	<p>the environmental protection agency said it is considering fees on windfall profits that may result from new production quotas for chemicals thought to be depleting the earth's ozone layer .</p> <p>the e. p. a.'s comment came as the agency as expected set production quotas that could sharply drive up the price of chlorofluorocarbons or c. f. c.s .</p> <p>the quotas which affect the five u. s. producers of c. f. c.s would bring the country into compliance with a treaty signed in montreal last september aimed at reducing world wide production . under the quotas producers cannot exceed their nineteen eighty six production of c. f. c.s in any one year .</p> <p>the quotas go into effect in july nineteen eighty nine if eleven of the treaty signatories representing two thirds of world production have ratified the treaty by january first nineteen eighty nine . at the end of last month thirty seven nations had signed the treaty and six nations had ratified it including the u. s. .</p> <p>we want producers to produce substitutes faster and users to use substitutes faster said eileen claussen an e. p. a. official for air and radiation issues .</p> <p>but producers argued that the profits would help pay the expense of switching over to substitutes .</p> <p>du pont company which last week announced that it would no longer be making c. f. c.s by the year two thousand said the fee would be unnecessary because the industry is already</p>

	<p>moving rapidly toward finding substitutes . du pont the world's largest producer of c. f. c.s is planning to spend thirty million dollars this year on research and development of substitutes and said higher prices it receives for its products will never become profits .</p> <p>you'll end up taking away the incentive for alternatives development or for conservation said joseph steed environmental manager for du pont's c. f. c. division .</p> <p>pennwalt corporation the third largest u. s. producer said its research budget has increased by four times and that it expects to modify or close plants soon to phase out all its c. f. c. production eventually .</p> <p>if somebody thinks we're going to make money out of all this they're dreaming said peter miller manager of pennwalt's c. f. c. division . we'll do everything we can to fight the fee .</p> <p>the e. p. a. said it will also consider holding an auction of rights to produce the limited amount of c. f. c.s as another way to curtail any windfall profits .</p> <p>c. f. c.s are used in car and building air conditioning cleaning agents and plastic foam . scientists believe c. f. c.s contribute to the breakdown of the ozone layer which screens out ultraviolet rays . excess ultraviolet rays can cause skin cancer and other health and environmental problems .</p> <p>laurie hays in philadelphia contributed to this article .</p>
--	---

**Table A.3** LSA query result of a document about “Du Pont company”

## References

---

- AMBIA (1996). Re:mark. *Markup and review for Adobe Acrobat software*. User's Guide. [Http://www.ambia.com/remark.html](http://www.ambia.com/remark.html).
- Baeza-Yates, R. and Ribeiro-Neto, B. (1999) *Modern Information Retrieval*. Addison-Wesley.
- Barger, D., Gupta, A., Grudin, J., Sanocki, E. (1999) *Annotations for streaming video on the web: system design and usage Studies*. Proceedings of the Eighth International World Wide Web Conference (WWW8).
- Bartell, T. Brian, Cottrell, W. Garrison and Belew, K. Richard K (1992) *Latent Semantic Indexing is an optimal special case of multidimensional scaling*. SIGIR Forum (ACM Special Interest Group on Information Retrieval), p. 161-167.
- Bernheim, A. J. and Barger, D. (2001) *Robust anchoring annotations using keywords*, Technical Report MSR-TR-2001-107.
- Bernheim, A.J., Barger, David, Gupta, A., Cadiz, J. J. (2001) *Robust annotation positioning in digital documents*, SIGCHI'01, Seattle, WA.
- Berry, M. W. (1992) *Large scale singular value computations*, Internat. J. Supercomputer Appl., 6, pp. 13-49.
- Berry, M. W., Do, T., O'Brien, G., Krishna, V., and Varadhan, S. (1996) *SVDPACKC (Version 1) user's guide*. Computer Science Department, University of Tennessee.
- Berry, M. W., Dumais, S. T., O'Brien, G. W. (1995) *Using linear algebra for intelligent information retrieval*, SIAM Review, Vol. 37, No. 4, pp. 573-595.
- Brown, P.J. and Brown, H. (2003) *Annotation: a Step towards the read/write document*. [http://www.dcs.ex.ac.uk/~pjbrown/papers/annotaiton\\_guide\\_experiment.pdf](http://www.dcs.ex.ac.uk/~pjbrown/papers/annotaiton_guide_experiment.pdf).
- Cardiz, J., Gupta, A. and Grudin, J. (2000) *Using web annotations for asynchronous collaboration around documents*. Proceedings of CSCW'00, Philadelphia, PA.
- Croft, B. (2001) *Class notes*. CMPSCI 646: Information Retrieval.

- Davis, J. and Huttenlocher, D. (1995). *Shared annotation for cooperative learning*. Proceedings of the 1995 Conference on Computer Supported Cooperative Learning (CSCL 1995).
- Deerwester, S., Dumais, S. T., Furnas, G. W., and Landauer, T. K. (1988) *Indexing by Latent Semantic Analysis*, J. of the American Society for Information Science. 41(6): 391-407.
- Dumais, S. T. (1991) *Improving the retrieval of information from external sources*, Behavior Res. Meth., Instruments, Computers, 23, pp. 229-236.
- Dumais, S. T. (1994) *Latent Semantic Indexing (LSI) and TREC-2*, In D. Harman (Ed.), The Second Text Retrieval Conference (TREC2), National Institute of Standards and Technology Special Publication 500-215, pp. 105-116.
- Dumais, S. T. (1997) *Using Latent Semantic Indexing (LSI) for information retrieval, information filtering and other things*, Cognitive Technology Conference, April 4.
- Dumais, S. T. and Nielsen, J. (1992) *Automating the assignment of submitted manuscripts to reviewers*, in SIGIR'92: Proceedings of 15<sup>th</sup> Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Denmark, ACM Press, pp. 233-244
- E-quill. <http://www.e-quill.com/>.
- Foltz, P. W. (1990) *Using latent semantic indexing for information filter*, in Proc. ACM Conference on office Information Systems (COIS), pp. 40-47.
- Foltz, P. W. and Dumais, S. T. (1992) *Personalized information delivery: an analysis of information filtering methods*, Communications ACM, 35, pp. 51-60.
- Frakes, W. and Baeza-Yates, R. (1992) *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, NJ, USA.
- Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., and Lochbaum, K. E. (1988) *Information retrieval using a singular value decomposition model of latent semantic structure*, Proc. SIGIR, pp. 465-180.
- Gronbak, K., Sloth, L. and Orbak, P. (1999) *Webvise: browser and proxy support for open hypermedia structuring mechanisms on the WWW*, Proceedings of the Fifth International World Wide Web Conference, Toronto.
- Harman, D. (1986) *An experimental study of factors important in document ranking*, In Proceedings of ACM SIGIR, Pisa, Italy.
- HyperNix, <http://www.hypernix.com>.

- Jones, S. K. (1972) *A statistical interpretation of term specificity and its applications in retrieval*, Journal of Documentation, 28(1), 11-21.
- Kahan, J. and Koivunen, M-R. (2001) *Annotea: an open RDF infrastructure for shared web annotations*, Proceedings of the WWW10 International Conference, Hong Kong.
- Kwon, O-H, Kim, M-C, and Choi, K-S. (1994) *Query expansion using domain-Adapted, weighted thesaurus in an extended boolean model*. CIKM 94, Proceedings of the Third International Conference on Information and Knowledge Management, P. 140-146.
- Landauer, T. K., Foltz, P. W., and Laham, D. (1998) *An introduction to Latent Semantic Analysis*, Discourse Processes, 25, 259-284.
- Landauer, T. K. and Littman, M. L. (1990) *Fully automatic cross-language document retrieval using latent semantic indexing*, in Proc. 6<sup>th</sup> Annual Conference of the UW Center for the New Oxford English Dictionary and Text Research, UW Center for the New OED and Text Research, Waterloo, Ontario, pp. 31-38.
- Lee, J. H., Kim, W. Y., Kim, M. H. and Lee, Y. J. (1993) *On the evaluation of boolean operators on the extended boolean retrieval framework*. Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, P. 291-297.
- Luhn, H. P. (1958) *The automatic creation of literature abstracts*, IBM Journal of Research and Development, 2, 159-165.
- Marshall, C. C. (1997) *Annotation: from paper books to the digital library*, Proceedings of Digital Libraries, Philadelphia, PA.
- Marshall, C. C. (1998) *Toward an ecology of hypertext annotation*, Proceedings of HyperText '98, Pittsburgh, PA.
- Nielsen, J., Phillips, V. L., and Dumais, S. T. (1994) *Information retrieval of imperfectly recognized handwriting*, Behavior and Information Technology.
- NovaWiz, <http://www.novawiz.com>.
- O'Hara, K. and Sellen, A. (1997) *A comparison of reading paper and online documents*. Proceedings of CHI'97, Conference on Human Factors in Computing Systems, Atlanta, GA.
- Ovsianniko, I. A., Arbib, M. A. and Mcneill, T. H. (1999) *Annotation technology*, Int. J. Human-Computer Studies 50, 329-362.

- Phelps, T. A. and Wilensky R. (1996) *Multivalent documents: inducing structure and behaviors in online digital documents*. Proceedings of the 29<sup>th</sup> Hawaii International Conference on System Sciences, Maui, Hawaii.
- Phelps, T. A. and Wilensky, R. (1997) *Multivalent annotations*, Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries, Pisa, Italy.
- Phelps, T. A., Wilensky, R. (2000) *Robust intra-document locations*. Computer Networks 33, 105-118.
- Rehder, B., Littman, M. L., Dumais S., and Landauer T. K., (1997). *Automatic 3-language cross-language information retrieval with latent semantic indexing*, in The Sixth Text Retrieval Conference, pages 103--110. National Institute of Standards and Technology Special Publication.
- Roscheisen, M., Mogensen, C. and Winograd, T. (1995) *Shared web annotations as a platform for third-party value-added information providers: architecture, protocols, and usage examples*. Technical Report CSDTR/DLTR.
- Salton, G. (1968) *Automatic Information Organization and Retrieval*, McGraw Hill, New York.
- Salton, Gerard. (1983) *Introduction to Modern Information Retrieval*. McGraw-Hill.
- Salton, Gerard. (1988) *Automatic Text Processing*. Addison-Wesley Publishing Company.
- Salton, G. and Buckley, C. (1996) *Term weighting approaches in automatic text retrieval*. Information Processing and Management Vol. 32 (4), P. 431-443.
- Shannon, C. E. (1948) *A mathematical theory of communication*, Bell System Tech. J., vol. 27, 379-423 and 623-656, July and October.
- Story, R. Roger (1996) *An explanation of the effectiveness of latent semantic indexing by means of a Bayesian regression model*. Information Processing and Management, Vol.32 (3), p.329-344.
- Streeter, L. A. and Lochbaum, K. E. (1988) *An Expert/Expert Locating System based on Automatic Representation of Semantic Structure*. In: Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications, Computer Society of the IEEE, San Diego, CA: 345 - 349.

Third Voice. <http://www.thirdvoice.com>.

UTok. <http://www.utok.com>.

Yee, K-P. The CritLink Mediator. <http://crit.org/critilink.html>.

Zadu. <http://www.zadu.com>.

Zipf, H. P. (1949), Human Behavior and the Principle of Least Effort. Addison-Wesley, Cambridge, Massachusetts.