AUTOMATIC TEXT SUMMARIZATION IN DIGITAL LIBRARIES

Angela Mlynarski
B.Sc., University of Lethbridge, 2003

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfilment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

Dedication

I dedicate this thesis to my daughter Kaila; you are my inspiration and my life. Always remember that you can accomplish anything you set your mind to.

Abstract

A digital library is a collection of services and information objects for storing, accessing, and retrieving digital objects. Automatic text summarization presents salient information in a condensed form suitable for user needs. This thesis amalgamates digital libraries and automatic text summarization by extending the Greenstone Digital Library software suite to include the University of Lethbridge Summarizer. The tool generates summaries, nouns, and noun phrases for use as metadata for searching and browsing digital collections.

Digital collections of newspapers, PDFs, and eBooks were created with summary metadata. PDF documents were processed the fastest at 1.8 MB/hr, followed by the newspapers at 1.3 MB/hr, with eBooks being the slowest at 0.9 MB/hr. Qualitative analysis on four genres: newspaper, M.Sc. thesis, novel, and poetry, revealed narrative newspapers were most suitable for automatically generated summarization. The other genres suffered from incoherence and information loss. Overall, summaries for digital collections are suitable when used with newspaper documents and unsuitable for other genres.

Acknowledgements

I would like to thank all the people who have helped me during this thesis project. First, I would like to thank my supervisor, Dr. Ian Witten who gave me the opportunity to pursue this thesis. I extend my gratitude and thanks to the New Zealand Digital Library group, with special thanks to Katherine Don and Michael Dewsnip, who taught me the inner workings of the Greenstone Digital Library Software Suite that was used in this project. I would also like to thank my co-supervisor, Dr. Yllias Chali and his research students who wrote the University of Summarizer that was used in this project. Special thanks goes to Dr. Dan O'Donnell who guided me through my thesis writing process. Thank you Dr. Stephen Wismath for your help in organizing and preparing for my thesis defence.

I wish to extend my special gratitude and thanks to my family. Mom and Dad, you have supported me in every stage of my education; without you, I would never have completed this thesis. To my daughter Kaila, you are my inspiration for going back to school and pursuing my degree. Thank you to my boyfriend Morgan, you never gave up on me, even when I felt like giving up on myself.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

We live in a wonderful era, with information readily available at our fingertips. Gone are the days when we were limited to trudging through bookcases at the public library, rummaging through the books, in hopes of finding that much needed information. The day has passed when our only means of shopping was fighting through crowded malls, desperately searching for that perfect Christmas gift. No longer are we restricted to reading current events in newspapers, seeing it on television, or listening to it on the radio. Today we live in the information era, a time when the information we seek is readily available at our fingertips. With access to the Internet and the entry of a few keywords on a search engine, we can be instantly rewarded with the information we are hunting. This process of extracting information from a larger accumulation is referred to as *Information Retrieval* (DiStefano, Rudestam, & Silverman, 2003, p. 231).

The Internet is the worldwide system of interconnected computer networks, designed to make online information accessible. Search engines are programs designed to find information stored on computers or computer networks; they are widely used for finding information on the Internet. In response to user specified queries, search engines commonly return a list of references matching the specified criteria. On the World Wide Web, the listing normally includes a link to the web page, the page's title, and an excerpt from the page with the query terms highlighted. Riffling through listings with only these bits of information guiding your way can be time consuming, and you still may not find the pertinent document. Finding the right document can be overwhelming and utterly frustrating.

Digital libraries (DL) are another means by which people can access information. Existing DLs hold collections of carefully chosen information focused on particular topics of interest. The information may be in textual, image, audio, or video form—or any mixture. Users can use DLs, in a similar manner to the World Wide Web, to search and retrieve material. Many excellent schemes exist for finding relevant information in DL collections—full-text searching, browsing based on available metadata, and schemes that depend on information automatically extracted or "mined" from the document text, such as phrases and keyphrases (keywords). Metadata is descriptive data that is associated with each document. Examples of document metadata include the document's title, author, creation date, file format, keywords, etc. It can be indexed and used for searching and browsing lists. One system users can use to build and manage DLs is the Greenstone Digital Library software suite (Greenstone).

1.1 Motivation for this Thesis

Reference librarians assist people in finding information. One way they do this is by evaluating potential sources of information (Kan and Klavan, 2002). DLs provide new methods for librarians to carry out library functions (Fox, Akscyn, Furuta, & Leggett, 1995, pp. 24-25). One problem with finding relevant text is that information retrieval (IR) systems do not provide means for quickly reading and inspecting documents (Hernandez & Grau, 2003, p. 117). Information supplied by a DL needs to be read and assimilated, and because of the vast amount of digital content that is available this can be an exhausting and time-consuming undertaking. It would be useful to supply DL readers with automated tools that enable them to examine the contents of individual documents or groups of documents at a glance.

Natural language processing (NLP) is a branch of artificial intelligence and linguistics. It uses computers to process human languages. A particularly useful task in NLP is automatic summarization. Its goal is to condense textual information into a coherent summary. Summarization provides users with a preview of the document (Parmanto, Ferrydiansyah, Saptono, Song, Sugiantara, & Hackett, 2005, p. 18). One system that produces summaries from documents is the University of Lethbridge Summarizer (Summarizer) (Brunn *et al*, 2002). This thesis focuses on the marriage of the Greenstone Digital Library Software Suite and the University of Lethbridge Summarizer.

1.2 Project Overview

This thesis explores the use of automatic text summarization in practical DLs. The project incorporates the Summarizer into the Greenstone system as an option that librarians can invoke when building collections. It was implemented in a flexible way, so that collection-builders can instruct Greenstone to:

- Extract summary metadata of length *n*

- Extract summary metadata based on the first *n* bytes of text input
- Extract all noun phrases as metadata that are used for searchable index or browsable hierarchy
- Extract all nouns as metadata that are used for searchable index or browsable hierarchy
- Provide options for handling genre documents

Although the aim of this project was not to alter the Summarizer's code, certain issues arose which required adjustments to be made in order to ensure its robustness. Testing was performed on collections with HTML, PDF, and plain text documents. These tests were executed on a 2.8 GHz P4 system with 512 KB cache, 2 GB of RAM. The operating system was Red Hat 9 Enterprise, running on an XFS file system on 10,000-RPM SCSI hard drives. This computer was not dedicated solely for our test purposes and other users had access to it.

1.3 Thesis Outline

The thesis is organized as follows. Chapter 2 introduces background information pertinent to the study of DLs and automatic text summarization. It is divided into four parts. First, the world of DLs is explored: what they are, how they are created, and a look at some existing DLs. Second is an in-depth profile of Greenstone, including who designed it, and how it creates, indexes, searches, and browses a DL collection. Third, is a study on automatic text summarization, including the methods used and what systems are available. Fourth, the Summarizer is outlined, including the stages it traverses to automatically generate summaries.

Chapter 3 is a comprehensive examination of the thesis project. This chapter contains three sections. The first section describes how to install the Summarizer into Greenstone. It explains what Greenstone plugins are, how the Summarizer is implemented as a plugin, and how to install it as a package. The second section examines issues that arose during implementation and how they were handled. The third section focuses on the special options written for the Summarizer in the thesis project, what they are, how to use them and where they would be used.

Chapter 4 is the technical evaluation portion of the thesis. Its purpose is to analyze the Summarizer as a part of Greenstone and determine if it is feasible to produce summaries for digital collections. Two types of feasibility issues are discussed: time and information loss. We will see that, given certain factors, feasibility is attainable. In this chapter, three areas are examined. The first looks at what factors play a role with regards to time; the second includes recommendations on solving time issues; and the third discusses information loss.

Much research has been done with automatic summarization using the newspaper genre. Chapter 5 focuses on how the Summarizer works with other genres, as well as the newspaper genre. Our hypothesis is that narrative-style genres produce more coherent summaries than non-narrative-styles. Three sections are included in this chapter. The first describes the genres; the second contains the analysis; and the third discusses when the Summarizer should be used and when it should not.

Chapter 6 concludes this thesis. This final chapter summarizes key points in the thesis, along with a section devoted to future work.

Chapter 2

Background

This research covers two areas: digital libraries and automatic text summarization. A background study of these topics will be covered in this chapter. The first half researches DLs: Section 2.1 covers an overview on DLs, while Section 2.2 examines Greenstone, which was used in the thesis project. The second half studies automatic text summarization: Section 2.3 encompasses automatic text summarization, whilst Section 2.4 explores the Summarizer used in the thesis project.

2.1 What is a Digital Library?

The idea of the DL was envisioned as early as 1945 when Vannevar Bush enraptured the world with his idea for a device he termed the "memex". Bush (1996) described it as

> a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility.

While he admitted that technology did not exist at that time to implement his device, his concepts were similar to those we see in today's DLs. In his vision, the memex would consist of a desk with a screen, keyboard, buttons, and internal storage facilities. Books, pictures, newspapers, and other documents would be stored on microfilm. His machine would also include a method for data entry, incorporate indexing for document retrieval, allow users to add notes in the document's margin, feature a means for viewing documents, and accommodate querying and browsing. In Bush's memex, query searches and browsing paths could be saved for future use. Today's DLs include similar functionalities: storage, indexing, document viewing, querying, and browsing. This section explores these concepts.

The term *digital library* has been defined in a multitude of ways. Gladney et al., (1994) compare DLs to conventional libraries:

> A *digital library* is a machine readable representation of materials, which might be found in a university library together with organizing information, intended to help users find specific information. A *digital library service* is an assemblage of digital computing, storage, and communications machinery together with the software needed to reproduce, emulate, and extend the services provided by conventional libraries based on paper and other material means of collecting, storing, cataloging, finding, and disseminating information. A full service digital library must accomplish all essential services of traditional libraries and also exploit digital storage, searching, and communication. (Gladney et al., 1994, p. 107)

Witten and Bainbridge (2003) define a DL as:

> a focused collection of digital objects, including text, video, and audio, along with methods for access and retrieval, and for selection, organization, and maintenance of the collection.

Based on these two viewpoints, a DL is a collection of services and information objects for storing, accessing, and retrieving digital objects. It supports users in handling digital objects by providing means to store, organize, index, maintain, access, retrieve, and present objects in a digital format. These services can be divided and employed by two communities: librarians and end users. The

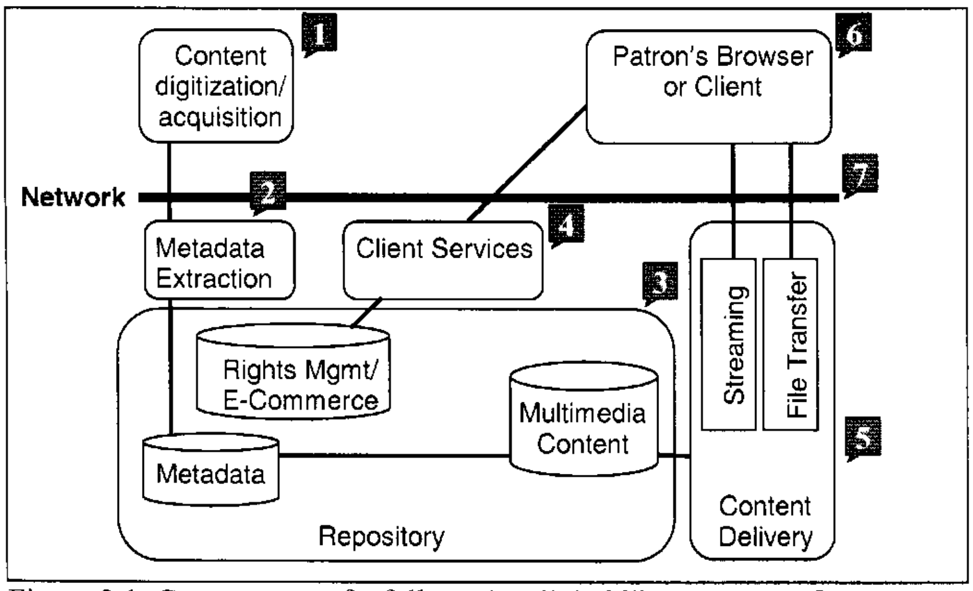


Figure 2.1: Components of a full service digital library system from *Digital library technology trends* (Pasquinelli, 2002, p. 9).

librarian's job is to select which objects to include in the collection, organize, and maintain it. End users require services to access, retrieve, and view digital information from the DL.

2.1.1 Digital Library Structure

Pasquinelli describes seven key components that are required in a fully developed DL environment (Figure 2.1):

1. Initial conversion of content from physical to digital form
2. The extraction or creation of metadata or indexing information describing the content to facilitate searching and discovery, as well as administrative and structural metadata to assist in object viewing, management, and preservation
3. Storage of digital content and metadata in an appropriate multimedia repository. The repository will include rights management capabilities to enforce intellectual property rights, if required. E-commerce functionality may also be present if needed to handle accounting and billing.
4. Client services for the browser, including repository querying and workflow
5. Content delivery via file transfer or streaming media

6. Patron access through a browser or dedicated client
7. A private or public network (Pasquinelli, 2002, p. 5)

Fox, Akscyn, Furuta, and Leggett (1995) report that DLs incorporate many support technologies including electronic publishing, hypermedia, education, data and information management. Electronic publishing is required to convert documents to electronic versions. For example, documents that are only available on paper need to be scanned. An optical character recognition (OCR) program processes scanned documents into a machine-readable format. Hypermedia presents multimedia objects, such as images, audio, and video, in a suitable fashion. According to Marchioni and Maurer (1995), DLs combine technology and information resources to support teaching and learning. Database management methods support direct usage of data collections in DLs, handle catalogs, administration, usage logs, and provides security. Information is converted, indexed, searched, and presented through IR techniques (Fox et al., 1995).

2.1.2 How to Create a Digital Library Collection

Loots, Camarzan, and Witten propose eight typical steps when creating a DL collection using Greenstone:

1. Selecting the documents to be included
2. Securing copyrights permissions to use these documents in the digital library
3. Scanning and OCR of the hard-copy documents which are not available in digital form to have a perfect digital format
4. Converting all documents to a format (integrating text and images) which can be imported into Greenstone (preferably HTML or Microsoft Word, but others are also covered at

varying levels of precision by a "plugin[1]" (Chapter 3, p. 42)

5. Tagging the chapters, paragraphs and images of the digital documents
6. Organizing the collection into a optimally structured digital library
7. Building the digital library using the Greenstone software
8. Printing and distributing the collection on CD-ROM and/or distributing it over the Internet (Loots et al., n.d., p. 1)

Section 2.2 will expound on the structure of Greenstone.

2.1.3 Advantages of Using Digital Libraries

DLs are advantageous over their traditional, paper counterparts. Advantages of DLs over their paper ones are similar to those of digital databases over their paper equivalents: data can be added faster, and DLs have better quality control and search functionality (Gadney et al., 1994, p.20). From the ecological perspective, using electronic versions reduces paper usage. From the technological perspective, Witten and Bainbridge (2002) maintain DLs

> are easier to access remotely, they offer more powerful searching and browsing facilities, and they serve as a foundation for new value-added services.

Sociologically, using the Internet to access collections enables users to share information globally and conveniently. Economically, DLs do not require large buildings to house the collections and can be maintained with fewer staff than traditional libraries.

2.1.4 Existing Digital Libraries

DLs are used for commercial, academic, and social uses. According to Gladney et al., (1994, p.2) self-contained commercial DL packages are used in a

[1] Plugins are used to parse import documents and will be explained in Chapter 3.

variety of industries including kiosk systems, public utilities, law firms, newspapers, engineers, pharmaceutical firms, law enforcement agencies, publishers, and universities. These industries use commercial DL packages for managing public information, correspondence, knowledge systems, electronic archives, technical documentation, approval testing, and publications.

A number of commercial DL systems also serve the academic area, such as Lexis, ProQuest, and the Association for Computing Machinery (ACM) Digital Library. Lexis (http://www.lexisnexis.com) is "the first commercial, full-text legal information service" designed to "help legal practitioners research the law more efficiently." Lexis provides accessibility to information, pertaining to legal, tax, and regulatory information, via the Internet, hardcopy print, or on CD-ROM.

ProQuest (http://www.proquest.com) is a subscription-based online information service that "provides access to thousands of current periodicals and newspapers, many updated daily and containing full-text articles from 1986."

The ACM Digital Library (http://portal.acm.org/dl.cfm) is a

> full-text repository of papers from publications that have been published, co-published, or co-marketed by ACM and other publishers.

ACM, an international organization, is a resource for computing literature. Papers may be purchased online or are available via member subscription. Non-members can perform basic searching and browsing; members can perform advanced searching on a combination of keyword, phrase, author, ISBN or type, publication date, item type, and/or ACM Computing Classification (CCS) term. In addition, members have unlimited access to full-text documents.

Numerous non-subscription based online DLs are available for academic and non-academic use. Two commonly used ones are the Math Digital Library (MathDL), and the University of California Music Library (UCLA). MathDL (http://mathdl.org) is a free source of mathematical information available online for both teachers and students. It is published by the Mathematical Association of America and is part of the National Science Digital Library (NSDL). MathDL is made up of The Journal of Online Mathematics and its Applications (JOMA), Digital Classroom Resources (DCR), Convergence, and Open Source Sharable Mathlets (OSSLETS). Collections can be browsed via categories, or searched via keywords. The advanced search feature allows categorical searching on singular or multiple subjects based on a combination of author, keywords, and/or range of publication dates.

The UCLA Music Library (http://digital.library.ucla.edu/apam) is a historical research archive containing the history of popular American music from 1790 to the present. Digital sheet music and performances belonging in the public domain are freely accessible. The archive can be searched via keywords, or browsed via name, title, cover art subject, or date.

DLs may also be used for self-archiving. Self-archiving is the process of depositing digital documents into a publicly accessible online repository. Documents and their metadata are submitted via a simple web interface. Readers can access, read, download, and print papers. The purpose is to maximize research findings online, which in turn maximizes visibility, accessibility, and usage. By analyzing 119,924 conference articles in computer science and related disciplines, Lawrence (2001) showed an average of 336% more citations to online articles compared to offline articles published in the same venue.

Marshall and Bly (2004) explored the social impact of DL usage. Participants were interviewed on how they saved and shared published information. They found that the process of sharing information "fosters social cohesion" and that it is important that DLs "become a venue for encountering and sharing information beyond that which meets immediate needs."

2.2 The Greenstone Digital Library Software Suite

The New Zealand Digital Library (NZDL) Project at the University of Waikato developed and distributed Greenstone (http://www.greenstone.org) in cooperation with the United Nations Educational, Scientific and Cultural Organization (UNESCO) and Humanitarian Information for All Programme (Human Info NGO). Greenstone is a complete software suite for building and distributing DL collections. Information is presented in an organized fashion and is publishable on the Internet, computer network, or on CD-ROM. Users can browse collections or search for specific words or phrases in the document text. Greenstone is available for Windows 95/98/NT/2000/XP, Linux, BSD, UNIX, and Mac OSX operating systems. It is released under the GNU General Public License and is free to download from http://sourceforge.net/projects/greenstone. All distributions include four main languages: English, French, Spanish, and Russian. Interfaces for thirty-four additional languages are available, including, Arabic, Chinese, Dutch, German, Greek, Hindi, Italian, Japanese, Maori, Polish, Thai, Ukrainian, and Vietnamese.

2.2.1 How Greenstone Creates a Digital Library Collection

This section's purpose is to briefly explain the process Greenstone uses to create DL collections. This necessitates explaining three key areas of Greenstone: the *collection configuration file*, *import phase*, and *build phase*. Greenstone handles DL collections containing a multitude of document formats. It uses a specific file, called the *configuration file*, to chronicle information pertaining to how a collection is structured, organized, and presented. All documents must undergo pre-processing to be presented in a collection. Two important processes are administered to create the collection: *import* and *build*. First, in the *import phase*, documents are gathered, converted to a standard format, and metadata is extracted. Second, in the *build phase*, data structures and indexes are created to enable searching and browsing on the collection.

Understanding the Collection's Configuration File

Greenstone automatically generates a *configuration file* for each collection. Although it is not necessary, users can edit the file to tailor the collection's structure, organization, and presentation. Figure 2.2 depicts an automatically generated text document Greenstone uses as the collection configuration file; Figure 2.3 is an example of a customized collection configuration file (Pan American Health Organization / Regional Office of the World Health Organization [PAHO/WHO], 1999). The default file (Figure 2.2) contains five sections consisting of administrative information, indexes, plugins, classifiers, and collection metadata. The customized configuration file (Figure 2.3) contains a formatting section. The default file will be examined first; the

```
 1  creator        creator@email.address
 2  maintainer     maintainer@email.address
 3  public         true

 4  indexes        document:text document:Title document:Source
 5  defaultindex   document:text

 6  plugin      ZIPPlug
 7  plugin      GAPlug
 8  plugin      TEXTPlug
 9  plugin      HTMLPlug     -smart_block
10  plugin      EMAILPlug
11  plugin      PDFPlug
12  plugin      RTFPlug
13  plugin      WordPlug
14  plugin      PSPlug
15  plugin      ImagePlug
16  plugin      NULPlug
17  plugin      ArcPlug
18  plugin      RecPlug      -use_metadata_files

19  classify    AZList -metadata Title
20  classify    AZList -metadata Source

21  collectionmeta collectionname       "Name of the Collection"
22  collectionmeta iconcollection       "Collection Image"
23  collectionmeta collectionextra      "Information About the Collection"
24  collectionmeta .document:text       "text"
25  collectionmeta .document:Title      "titles"
26  collectionmeta .document:Source     "filenames"
```

Figure 2.2: Greenstone's default collection configuration file.

customized file will be analyzed later.

In the default file the first section contains administrative information, including the creator's email address (line 1), maintainer's email address (line 2), and a flag stating whether the collection is accessible from the DL's main page (line 3). Collections are visible when `public` is set to `true` and not visible when it is set to `false`. A "hidden" collection can be advantageous in situations requiring information within a specific collection to be kept private. The second section (lines 4 and 5) specifies which indexes to create in Greenstone's *build phase*. The third section (lines 6-18) tells Greenstone what plugins to use while

```
1   creator      kjm18@cs.waikato.ac.nz
2   maintainer   greenstone@cs.waikato.ac.nz
3   beta         true
4   public       true

5   indexes      section:Title section:text,Title paragraph:text,Title
                 document:text,Title
6   defaultindexsection:text,Title

7   plugin       GMLPlug
8   plugin       ArcPlug
9   plugin       RecPlug

10  classify     Hierarchy -hfile CL.T.txt -metadata CL.T -buttonname Title
                 -sort Title
11  classify     Hierarchy -hfile CL.C.txt -metadata CL.C -buttonname Subject
                 -sort Title
12  classify     Hierarchy -hfile CL.O.txt -metadata CL.O –buttonname
                 Organization -sort Title

13  collectionmeta  collectionname  "Virtual Disaster Library"
14  collectionmeta  iconcollection  "_httpprefix_/collect/paho/images/paho.gif"
15  collectionmeta  iconcollectionsmall
                    "_httpprefix_/collect/paho/images/pahosml.gif"
16  collectionmeta  collectionextra "The Virtual Disaster Library was
    developed in November 1999, by PAHO/WHO (Pan American Health
    Organization / Regional Office of the World Health Organization). It
    contains 250 publications (25,000 pages) of experiences, ideas and
    solutions to advance the cause of disaster reduction."

17  collectionmeta  .section:text,Title      "chapters"
18  collectionmeta  .paragraph:text,Title    "paragraphs"
19  collectionmeta  .document:text,Title     "book titles"
20  collectionmeta  .section:Title           "section titles"

21  format SearchVList \
      "<td valign=top>[link][icon][/link]</td><td>{If}{[parent(All':'):Title],
    [parent(All': '):Title]:}[link][Title][/link]</td>"
22  format DocumentText     "<h3>[Title]</h3>\\n\\n<p>[Text]"
23  format DocumentButtons "Expand Text|ExpandContents|Detach|Highlight"
24  format DocumentImages true
25  format HelpBookDocs     true
```
Figure 2.3: Custom built collection configuration file.

processing the import documents in the *import phase*. Greenstone's *import* and *build phases* will be explained later. Plugins are used to parse import documents; Chapter 3 will explain what plugins are and how they are used. The fourth section (lines 19 and 20) details which classifiers to employ during the *build phase*. Classifiers will be explained later. What classifiers are and how they are adopted in Greenstone's browsing feature. The fifth section (lines 21-26) defines the collection level metadata; it is denoted by `collectionmeta`. Four different types of `collectionmeta` are shown:

1. *collectionname*: identifies the name of the collection
2. *iconcollection*: gives the path to the collection's image file
3. *collectionextra*: contains information about the collection; such as its topic, when it was created, how to browse and search, etc
4. *index metadata*: includes descriptions of the searchable indexes defined in lines 1 and 2

A flag indicating whether the collection is a beta version (in the testing stage) (line 3) has been added to the customized configuration file (Figure 2.3). Beta collections are indicated when `beta` is set to `true`. The customized file demonstrates the three different levels of indexes (line 5): document, section, and paragraph. These index levels will be explained afterwards.

Two image files are declared in the collection metadata section: *iconcollection* and *iconcollectionsmall.* The location of the large image file (Figure 2.4a) representing the collection on the main DL page is defined by `iconcollection` (line 14); `iconcollectionsmall` (line 15) designates the file location for the collection's small image file (Figure 2.4b) that appears on the user interface pages. Information about the Virtual Disaster Library (Figure 2.4b) is defined by `collectionextra` on line 16 of the customized configuration file (Figure 2.3) and descriptions of the collection's searchable indexes are defined as index metadata on lines 17-20. For example, the section-

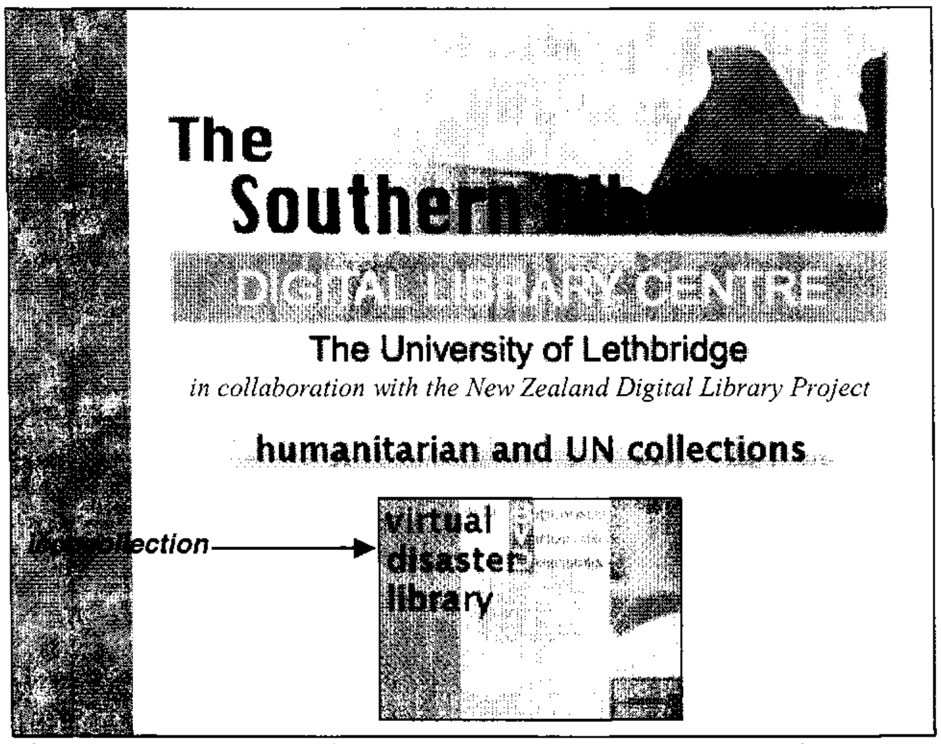
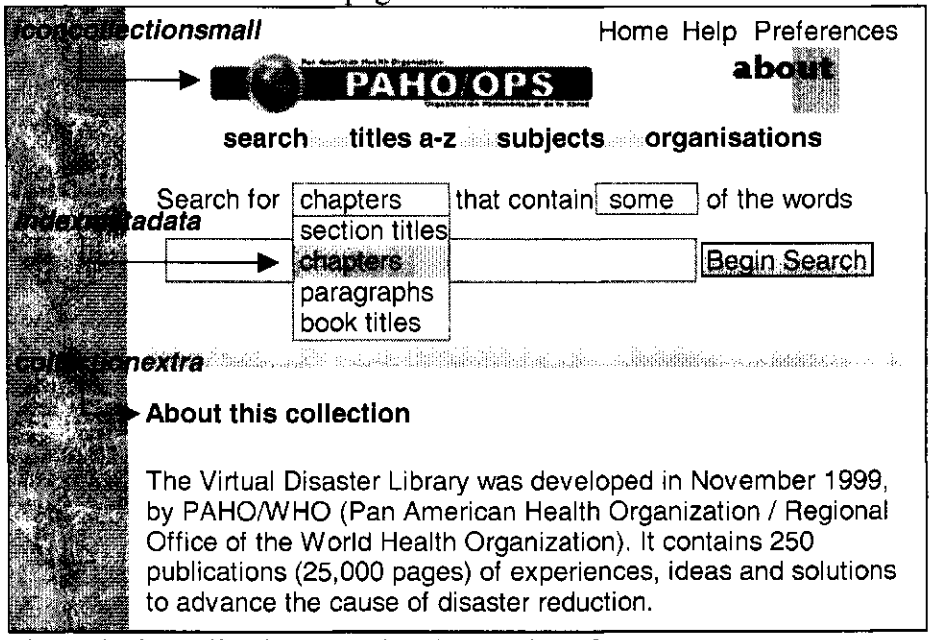

The Southern Alberta
DIGITAL LIBRARY CENTRE
The University of Lethbridge
in collaboration with the New Zealand Digital Library Project
humanitarian and UN collections
iconcollection
virtual disaster library

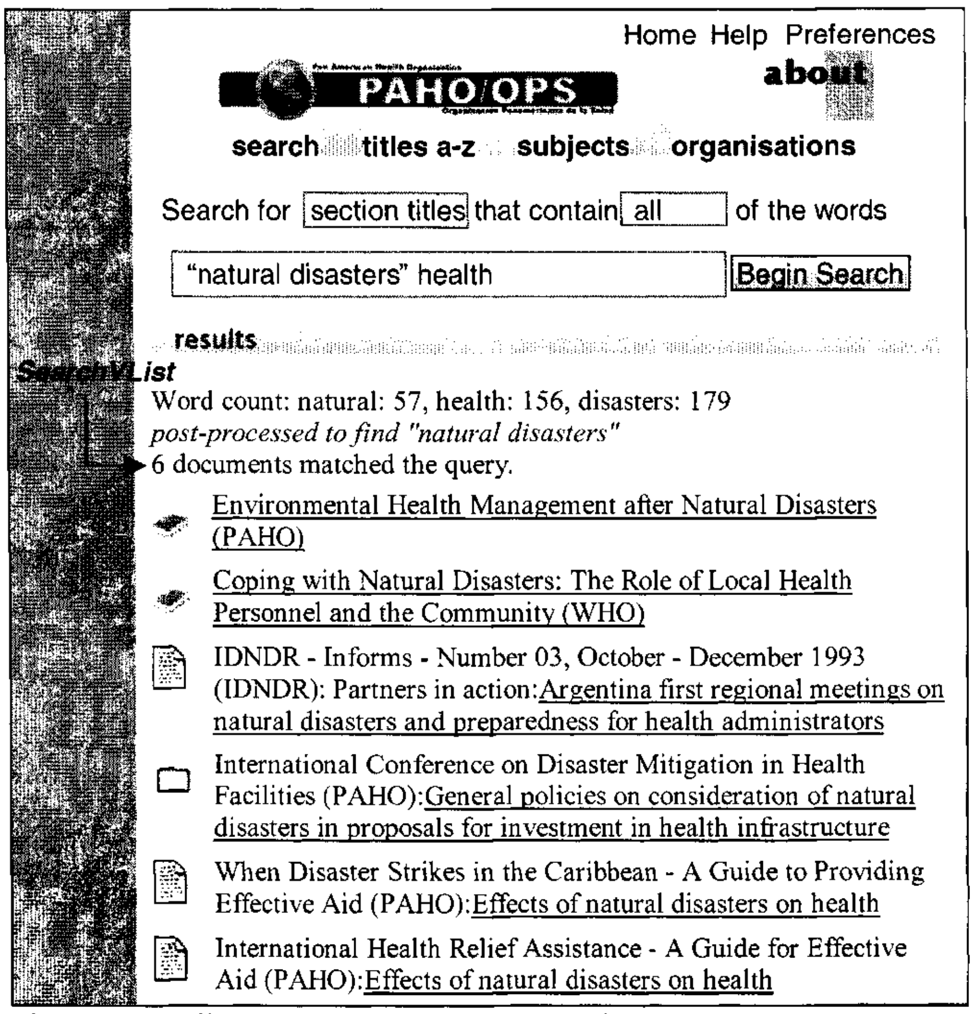
Figure 2.4a: Image metadata (`iconcollection`) representing a collection on a DL's main page.



The Virtual Disaster Library was developed in November 1999, by PAHO/WHO (Pan American Health Organization / Regional Office of the World Health Organization). It contains 250 publications (25,000 pages) of experiences, ideas and solutions to advance the cause of disaster reduction.

Figure 2.4b: Collection metadata in user interface pages: *iconcollectionsmall*, *index metadata*, and *collectionextra*.

level searchable index for `Title` metadata (`.section:Title`) is referred to as "`section titles`" in line 20 (Figure 2.3), and illustrated in Figure 2.4b.



Search for section titles that contain all of the words

"natural disasters" health

Word count: natural: 57, health: 156, disasters: 179
post-processed to find "natural disasters"
6 documents matched the query.

Environmental Health Management after Natural Disasters (PAHO)

Coping with Natural Disasters: The Role of Local Health Personnel and the Community (WHO)

IDNDR - Informs - Number 03, October - December 1993 (IDNDR): Partners in action:Argentina first regional meetings on natural disasters and preparedness for health administrators

International Conference on Disaster Mitigation in Health Facilities (PAHO):General policies on consideration of natural disasters in proposals for investment in health infrastructure

When Disaster Strikes in the Caribbean - A Guide to Providing Effective Aid (PAHO):Effects of natural disasters on health

International Health Relief Assistance - A Guide for Effective Aid (PAHO):Effects of natural disasters on health

Figure 2.5: Collection formatting in user search pages: *SearchVList*

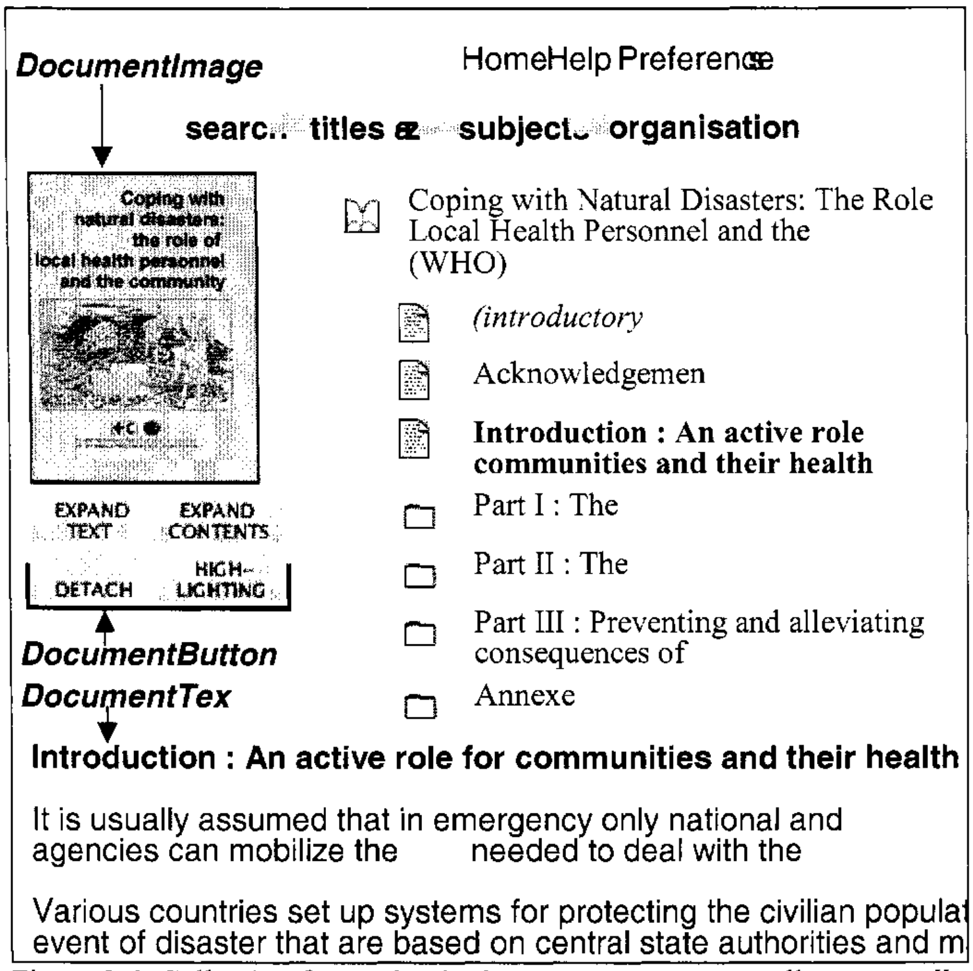
As previously stated, the customized version demonstrates some of the formatting options available in Greenstone. Web pages viewed in Greenstone are generated dynamically; any alterations to the format strings in the collection's configuration file (lines 21-25) will be seen without having to re-build the entire collection. Two types of formatting exist: one for the searching/browsing lists, the other for the document's appearances. Figure 2.5 illustrates how `SearchVList` is used to format the *SearchResults* list. Its formatting is shown in line 21 of Figure 2.3:

```
format SearchVList \
"<td valign=top>[link][icon][/link]</td><td>{If}{
[parent(All':'):Title],[parent(All':'):Title]:}[link]
[Title][/link]</td>"
```

The code `[link][icon][/link]` states that the document icon is a hyperlink to the document. `{If}{[parent(All':'):Title],[parent (All': '):Title]:}` specifies that if the document resides as a subsection inside another section, then the parent section's title is listed. This is useful in cases where more than one subsection have identical titles. For example, "Effects of natural disasters on health" is the title for two of the documents in Figure 2.5. Supplying their parent section titles help identify each document. The document's title is used as a hyperlink to the document by specifying `[link] [Title][/link]`. Lines 22-25 (Figure 2.3) specify the documents' appearances (Figure 2.6). Document text formatting is stated by `DocumentText` in line 22:

```
format DocumentText "<h3>[Title]</h3>\\n\\n<p>
[Text]"
```

The document's title is presented in the `<h3>` heading style above the document's text. `DocumentButtons` (line 23) lists the buttons displayed on document pages: *Expand Text*, *Expand Contents*, *Detach*, and *Highlight*. When selected, *Expand Text* expands the entire text within the current section. *Expand Contents* expands the table of contents to allow the user to view all sections and subsections. Clicking on *Detach* opens the document in a new browser window. If the document was accessed through the search page, clicking on the *Highlight*



Coping with Natural Disasters: The Role Local Health Personnel and the (WHO)

(introductory

Acknowledgemen

Introduction : An active role communities and their health

Part I : The

Part II : The

Part III : Preventing and alleviating consequences of

Annexe

Introduction : An active role for communities and their health

It is usually assumed that in emergency only national and agencies can mobilize the needed to deal with the

Various countries set up systems for protecting the civilian popula event of disaster that are based on central state authorities and m

Figure 2.6: Collection formatting in document pages: *iconcollectionsmall*, *index metadata*, and *collectionextra*

button causes all the query terms that were used in the search page to be highlighted in the document. For example, the query in Figure 2.5 uses the phrase "natural disaster" and the word "health". All occurrences of these would be highlighted in the documents. `DocumentImages` (line 24) set to `true` enables a cover image to be displayed on the document pages.

Preparing Documents for Greenstone

In its *import phase*, Greenstone fetches files that will be used in the collection. This phase converts files from their original form to a standardized document form used by Greenstone for archiving. Document processing is achieved by using *plugins*; different document types require different plugins. For example, *TEXTPlug*, *PDFPlug*, *PSPlug*, *WordPlug*, *HTMLPlug*, *EMAILPlug*, and *RTFPlug* plugins handle .txt, .pdf, .ps, .doc, .html, email, and rich text files respectively. Utilizing the correct plugin is essential to building a sound collection. Without it, Greenstone does not know how to handle various file types, resulting in documents appearing gibberish, blank, or being omitted during the import phase. A deep look at Greenstone plugins will be given in Chapter 3.

Building the Digital Collection

Text compression and full-text indexes are created during Greenstone's *build phase*. Information on how the indexes will appear in the collection is specified by *classifiers* in the collection's configuration file. Classifiers will be discussed in the browsing section. Collections utilizing search tools on document metadata allow easy searching and browsing.

2.2.2 How Indexing and Searching Works

Index structures are automatically generated from each collection's document and supporting file. Indexes can be derived from full-text and document metadata. Greenstone has three levels of searchable indexes: *document*, *section*, and *paragraph*. The *document* index contains the full document text. A query on specific word(s) returns a list of all the documents containing these word(s). Documents can be divided into sections by using `<Section>` tags. The *section* level index contains text between these `<Section>` tags. In the *paragraph* level index, each paragraph is treated as a separate document.

Greenstone uses two programs for indexing: Managing Gigabytes (MG) and MGPP (MG++). MG (Witten, Moffat, and Bell, 1999) incorporates compression techniques for full text indexing. Users can choose between different indexes for searching. Queries can be performed on all the words specified, or some of them. MG++ (Don, n.d.) is an extension of MG that was developed for Greenstone at the University of Waikato. MG++ provides word-level indexes for proximity and fielded searching. Proximity searching allows users to search for terms based on the distance between them. For example, if the user wanted to search for "Dorothy and the Wizard in Oz" but was unsure of the title would use "Dorothy NEAR5 Oz". The list of all documents containing "Dorothy" within 5 terms of "Oz" on either side would be returned: "Dorothy and the Wizard in Oz", "Dorothy of Oz", "Dorothy's Mystical Adventures in Oz", and "Dorothy Returns to Oz". Fielded searching permits searching based on metadata. For example, a search for "`[Dorothy]:Title & [Oz]:Title & [Frank Baum]: Author`", where the search terms are denoted inside the square brackets and the name of the fields or metadata are to the right of the colon, would search for the terms "Dorothy" and "Oz" within the "Title" metadata, and "Frank Baum" within the "Author" metadata. This query returns the book entitled "Dorothy and the Wizard in Oz" written by "Frank Baum".

A preferences section allows users to specify more advanced search control features, including turning on/off case-folding, stemming, advanced query mode to allow Boolean operators, large query interface, and search history mode.

Additional features can be also changed, such as changing the default language, the default web browser software, and the interface format. Users can switch to a textual interface format, which is useful for those using large screen fonts or a speech synthesizer.

2.2.3 Navigating Through the Collection

DL collections may be browsed or searched. When browsing, lists are supplied that the user employs in order to examine documents (Witten, Boddie, & Thompson, n.d.). Greenstone integrates a number of useful classifiers to create these lists; a sampling includes the *Hierarchy*, *AZCompactList*, *DateList*, and *Phind* classifiers. The *Hierarchy* classifier organizes documents by classification and sub-classifications. *AZCompactList* produces alphabetically sorted lists separated by alphabet headings that allow users to browse the list in an orderly fashion. The *DateList* classifier creates document indexes and organizes them into lists based on date metadata. Lists are separated by year headings. Clicking on a specific year generates a listing of all documents pertaining to that year. The *Phind* classifier facilitates phrase browsing. This is an advanced feature permitting users to browse a list of phrases and documents based on a word or phrase. When a word or phrase is typed into the textbox, *Phind* returns a table of relevant phrases and documents. Selecting a given phrase will return more phrases and documents; selecting a given document opens it for viewing. Classifiers, therefore, help construct the collection's structure to ensure easy searching and navigating.

2.3 What is Automatic Text Summarization?

According to Mani (1999),

> the goal of automatic text summarization is to take a partially structured source text, extract information content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs.

In the 1950's, Luhn (1958) created the first automatically generated summaries he termed "auto-abstracts". Machine-readable documents were initially scanned by an IBM 704 data-processing machine. Using statistical analysis, he ranked each sentence in the document to find salient sentences. The highest-ranking sentences were extracted to produce the "auto-abstract".

2.3.1 Methods Used for Text Summarization

Two approaches are utilized in creating summaries: extraction and abstraction. In the sentence extraction approach, each sentence is assigned a score using a combination of statistical heuristics. Those ranking the highest are deemed the most salient and are extracted to include in the summary. The abstraction approach involves simplifying and condensing text. When text summaries are created manually using the abstraction approach, humans read the text, reinterpret it, and rewrite it in their own words. This approach requires understanding of the topics and the ability to rewrite the text. Producing abstracts involves complex heuristics to identify central topics, interpret the topics, and generate the summary. The abstraction approach is much more difficult to program than extraction, thereby making extraction the more commonly used approach in automatic text summarization.

While numerous heuristics exist to determine sentence saliency for sentence extraction, only a few will be described. The baseline heuristic scores sentences based on their order of appearance: the first sentence has the greatest score; the last sentence has the least. Similarly, the first sentence technique assigns the highest score to the first sentence of each paragraph. Some genres place importance on the position of the sentence in the document. For example, the first sentence of newspaper articles contains the most important terms and consequently, has the best score in the document. *Term frequency* (TF) is a commonly used approach that is also applied in information retrieval. Terms appearing more frequently are deemed more important than those appearing less frequently. Another approach compares terms within each sentence. Sentences containing multiple terms that also appear frequently in other sentences are prominent. Some heuristics score sentences based on parts of speech. For instance, in one heuristic, important sentences are those containing proper names; in another, pertinent sentences are those containing pronouns and adjectives.

2.3.2 Types of Automatic Text Summarization Systems

This section describes three types of text summarization systems: single document vs. multi-document, generic vs. query-based, and indicative vs. informative. A summary may be based either on a single document or on multiple documents. Single document summarization extracts salient sentences to create the summary. Multi-document summaries are not as straightforward; redundancy problems arise when the documents have the same content. Concatenating the salient sentences from each document can result in a repetition of similar sentences within the summary (Kan & Klavans, 2002, p. 36).

Content varies between generic-based and query-based summaries. Generic summaries base content on the document's main points. With a query-based summary, the user typically requests information on a specific theme. The summary is generated from document content pertaining to the requested theme.

Indicative summaries provide a general overview of the document. They highlights aspects of the document that would be of interest to the reader without supplying specific information. Informative summaries supply enough information for the summary to replace the actual document. They present the most important concepts. Query-based informative summaries retrieve aspects of the document relevant to the user's query.

2.3.3 Existing Text Summarization Systems

Text summarization is seen everywhere. We see it in headlines when we flip through the newspaper, we see search engines return web pages excerpts when we surf the Web, we see it on the back cover of books when we are deciding which book to buy, and we see it in abstracts of technical papers and theses. We will look at several examples of automatic text summarization systems.

Columbia's Newsblaster (McKeown et al., 2002) is an online system that provides daily updates on news articles. The service crawls through various news sites and retrieves articles, which are then sorted into groups covering the same event. Newsblaster uses Columbia's multi-document summarization system

(McKeown et al., 2001) to provide summaries on each event. The Columbia Summarizer uses a router to distinguish between different document types. It then invokes the appropriate summarization subsystem pertaining to that type. For example, biographical documents are summarized using the Dissimilarity Engine Multi-document Summarization (DEMS), while newspaper articles on the same event are summarized by MultiGen. MultiGen applies a sentence clustering approach in which documents are broken down into smaller text units and a similarity matrix is computed for each unit. These matrices identify similar paragraphs and cluster them together into *themes*. Themes are chronologically ordered by their publication dates. Closely related phrases within sentences of each theme are identified. MultiGen selects phrases, combines and generates them into an English sentence for each theme. When the summary is limited to a set length, scores are assigned to each theme based on their size, similarity score and significance.

Wireless technology enables users to access the Internet using handheld devices such as personal digital assistants (PDAs) and cellular phones. Viewing web pages on small screens can be difficult for users. The Power Browser Project (Buyukkokten, Garcia-Molina, & Paepcke, 2001) offers methods for summarizing web pages on handheld devices on two levels. At the upper level, web pages are partitioned into content units called Semantic Textual Units (STUs). STUs contain textual content such as paragraphs, lists, and image description text. These units are organized into a hierarchical structure. STUs may be nested, for example, each element within a list is nested within that list. Initially, each STU is collapsed to the top layer. Users may expand and collapse nested STUs on the handheld device. At the lower level, Power Browser uses five methods for displaying STUs: *incremental*, *all*, *keywords*, *summary*, and a *keyword/summary* combination. The first method, *incremental*, gradually reveals each STU. The second, *all*, expands the entire STU. The third, *keywords*, displays keywords extracted from the STU as its top level. The fourth, *summary*, presents the most salient sentence as its top level. The last method, *keyword/summary*, combines the third and fourth methods with the keywords as the STU's top level, and the summary as its second level. *Keywords* and *summary* sentences are found by using *TF/IDF* within sentence-clustering techniques. *TF/IDF* is a purely statistical technique that measures how frequently the phrase occurs within a given section, compared to other sections.

2.4 The University of Lethbridge Summarizer

Dr. Yllias Chali, Maheedhar Kolla, Nanak Singh, and Zhenshuan Zhang developed the Summarizer at the University of Lethbridge. The Summarizer used in this thesis is a slightly different version than the one currently available at the University of Lethbridge. For our purposes, only those features implemented in the version that has been installed with Greenstone will be covered. Please refer to Kolla (2004) for information on the Summarizer's additional features. As we have seen, summaries can be generated for numerous applications: single document, multi-document, generic-based, query-based, indicative, and informative. Although the Summarizer is designed for many of these uses, this thesis uses it to generate generic, indicative summaries from single-documents.

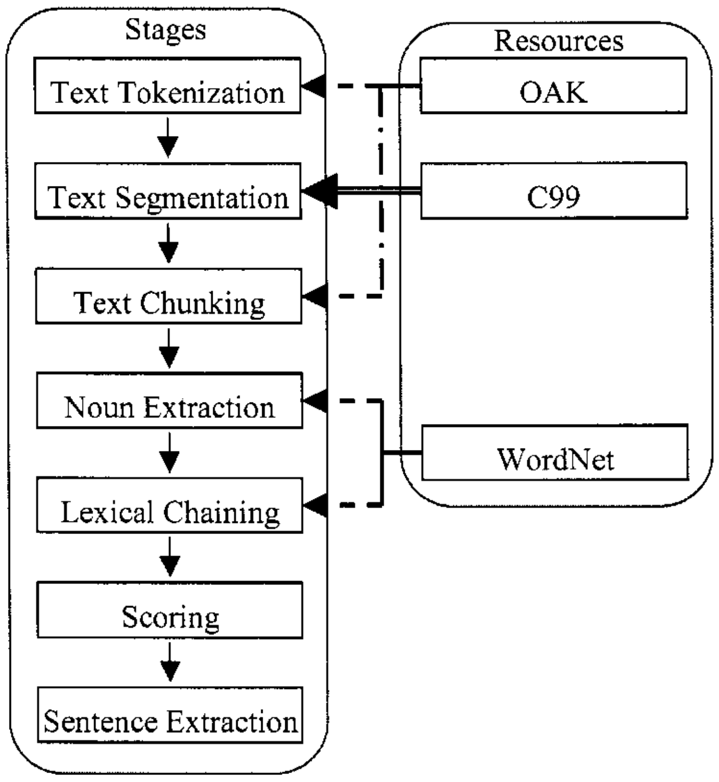


Figure 2.7: Summarizer architecture.

2.4.1 How the Summarizer Creates a Summary

The purpose of this section is to briefly explain the process the Summarizer uses to generate summaries. It uses the sentence extraction method of generating summaries, and traverses through seven stages (Figure 2.7): *text tokenization*, *text segmentation*, *text chunking*, *noun extraction*, *lexical chaining*, *scoring*, and *sentence extraction*. Input files undergo pre-processing in the first three stages: *text tokenization*, *text segmentation*, and *text chunking*. During these stages, the document is separated into segments and marked with part-of-speech tags. Candidate words are extracted in the *noun extraction* stage for generating lexical chains as topic indicators in the *lexical chaining* stage. Each sentence and segment is assigned a score in the *scoring* stage and the *sentence extraction* stage selects the highest-ranking sentences from the highest-ranking segments to include in the summary. Three outside resources: *OAK*, *C99*, and *WordNet*, required for completing tasks in the first five stages will be expounded upon later.

How the Summarizer Prepares Input Documents

A typical document normally encompasses several topics. Summarization necessitates analyzing each sentence and sorting it the appropriate topics. This is referred to as linear text segmentation. The segmentation algorithm requires each sentence to be placed on a separate line, also known as tokenization. The Summarizer uses *OAK* and *C99* to perform these two processes. Sekine developed the *OAK* system (Sekine, 2002b) at New York University. Sekine describes the system as

> a filter between text, splitted sentence, tokenized sentence, POS tagged sentence, chunked sentence, NE tagged sentence, dependency analyzed sentence, and so on (Sekine, 2002a, p. 4).

Source documents are split into topic segments and tagged. The segmentation process works with tokenized files. OAK takes a plain text formatted file as input, identifies sentences as text elements, and presents it in a usable form by tokenizing the source text.

C99 is a domain independent linear text segmentation algorithm that uses a combination of similarity and clustering to find topic boundaries in text. Once these boundaries are found, it partitions the input text into topical segments. Dr. Choi developed C99 (2000b) at the University of Manchester. The method is based on the idea that those blocks of text sharing common words belong to the same topic segment (Choi, 2000c).

The C99 algorithm uses the cosine similarity measure on tokenized text to compute the similarity between all pairs of sentences (Choi, 2000a) to generate a similarity matrix. Choi found that similarity values for short text segments are unreliable. His solution was to replace each value in the similarity matrix with its rank. The rank is defined as "the number of neighbouring elements with a lower similarity value" (Choi, 2000a). C99 applies a divisive clustering algorithm to locate the topic boundaries. Each topical segment is written to separate text files, which are used by the *text chunking* stage

Text Chunking

As shown in the Summarizer's architecture (Figure 2.7), lexical chains are computed to determine which sentences to extract for the summary. Nouns extracted from the text are required for computing lexical chains. Prior to this, source text is prepared by using OAK's chunker tool to perform text chunking. Text chunking is the process of splitting text into non-overlapping phrases (chunks) where syntactically related words belong to the same chunk (Sang & Buchholz, 2000, p. 127). For example, in the sentence:

```
[NP Several/JJ young/JJ musicians/NNS ] [VP will/MD
compete/VB ] [PP in/IN ] [NP this/DT year/NN ] [NP
's/POS music/NN festival/NN ] ./.
```

The square brackets contain text chunks. Next to each opening bracket is a part of speech tag indicating the chunk's type.

The *text chunking* stage concludes by outputting a file consisting of text chunks for each segment (created in the *text segmentation* phase) of the source document. Chunks consisting of noun phrases (NPs), referred to as NP chunking, are of particular interest, as we will see in Chapter 3. One option, written when the Summarizer was implemented in Greenstone, takes advantage of the text chunks written during this stage by extracting nouns and noun phrases for use as metadata in the DL collection. These options will be examined in Chapter 3.

Noun Extraction

The Summarizer uses *WordNet* to check the validity of nouns prior to extraction. *WordNet* (Miller et al., 2005) is an online lexical database that organizes English nouns, verbs, adjectives, and adverbs into synonym sets. It was developed under Miller's direction at Princeton University. The Summarizer is "based on the principle that nouns characterize the topic of a particular text" (Chali & Kolla, 2004, p. 107); it extracts nouns, compound nouns, and proper nouns from the text chunks. These extracted nouns are passed to the *lexical chain* stage to identify compound relations between words and compute lexical chains.

Lexical Chains

Morris and Hirst (1991) define lexical cohesion as "the cohesion that arises from semantic relationships between words". Lexical chains are sequences of semantically related words spanning topics of text. For example:

{soccer, football, contact sport, human action}

Following the above lexical chain, the word "soccer" is linked to "human action": "soccer" and "football" are synonyms; "football' is a kind of "contact sport', and "contact sport" is a kind of "human action".

Care needs to be taken when creating lexical chains because allowing unlimited transitivity can cause strange relations. In transitivity, if word *A* is related to word *B*, and word *B* is related to word *C*, then word *A* is also related to word *C*. For example:

{cow, sheep, wool, scarf, boots, hat, snow}

In the above chain, "cow" and "snow" are related by transitivity: "cow" and "sheep" are both mammals; "wool" comes "sheep"; "wool" can be used to make a "scarf"; a "scarf" is a piece of clothing, so is a "hat"; a "scarf", "boots", and a "hat" are worn when there is "snow" on the ground. Morris and Harris (1991) found that only allowing transitivity of one link is sufficient to compute intuitive chains and reduce non-intuitive relations.

Estimating similarity between two documents and finding similarity between their lexical chains are equivalent (Chen, 2004, p. 40). Lexical chains help interpret word meanings, identify discourse structure, and understand the overall main theme of the text. The Summarizer creates lexical chains from the nouns extracted in the *noun extraction* stage; these include regular, compound, and proper nouns. WordNet organizes English nouns, verbs, adjectives, and adverbs into synonym sets. The Summarizer queries WordNet to find semantic relations, synonyms and hypernyms. Synonyms are words that are interchangeable in some context, for example, *girl* and *miss*. A hypernym is a generic term used to designate a whole class of specific instances; for example, *animal* is a hypernym of *dog*.

The Summarizer begins creating lexical chains by using WordNet to find

7 sense of dog

Sense 1
dog, domestic dog, Canis familiaris
⇨ canine, canid

Sense 2
frump, dog
⇨ unpleasant woman, disagreeable woman

Sense 3
dog
⇨ chap, fellow, feller, lad, gent, fella, blighter, cuss

Sense 4
cad, bounder, blackguard, dog, hound, heel
⇨ villain, scoundrel

Sense 5
frank, **frankfuter**, hotdog, **hot dog**, dog, wiener, wienerwurst, weenie
⇨ sausage

Sense 6
pawl, detent, click, dog
⇨ catch, stop

Sense 7
andiron, firedog, dog, dog-iron
⇨ support

Figure 2.8: Synonym relationship: in the *WordNet* query for "dog", "frankfurter" and "hot dog" share the same synset Sense 5.

every sense of a word (Kolla, 2004). Each word sense is checked against one another to see if they are semantically related. Relations include:

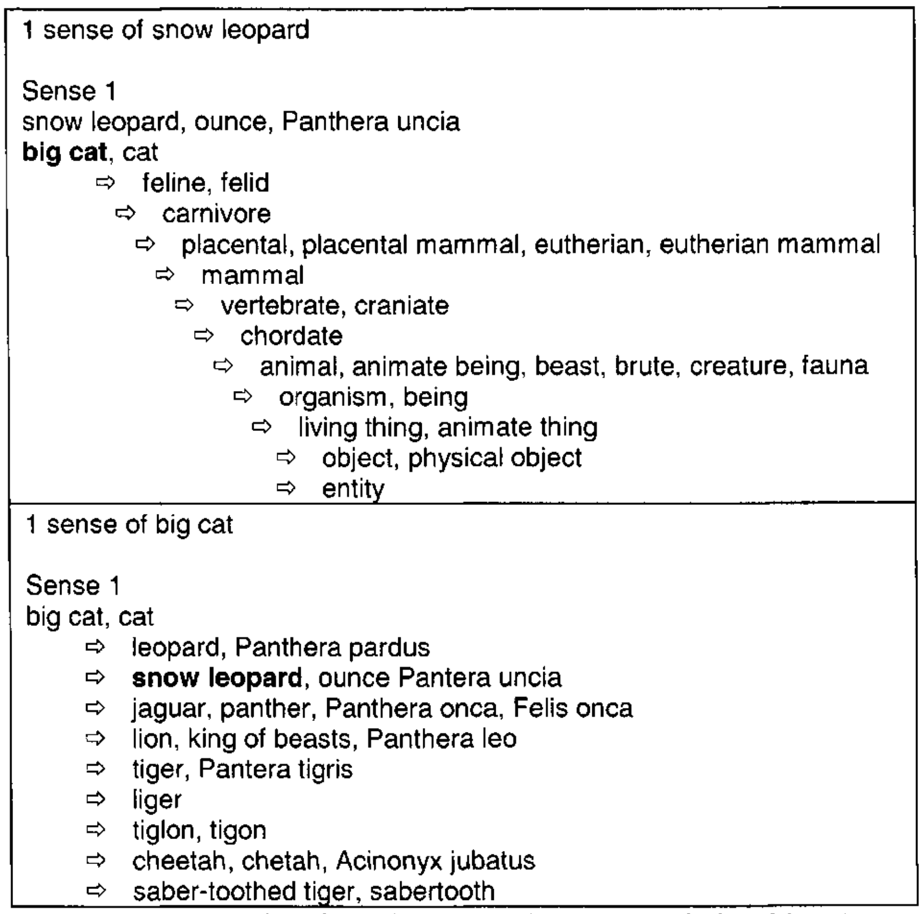
- Identical relation: words are syntactically and synonymously identical.

  E.g.: Morgan bought a **boat** today. It is the same model as Kaila's **boat**.

- Synonym relation: words belong to the same synset in WordNet and are interchangeable.

  E.g., The all beef **hot dog** is the best **frankfurter**.

  The words "hot dog" and "frankfurter" belong to the same synset (Figure 2.8); we could switch both without changing the meaning of the sentence.



```
1 sense of snow leopard

Sense 1
snow leopard, ounce, Panthera uncia
big cat, cat
       ⇨ feline, felid
         ⇨ carnivore
           ⇨ placental, placental mammal, eutherian, eutherian mammal
             ⇨ mammal
               ⇨ vertebrate, craniate
                 ⇨ chordate
                   ⇨ animal, animate being, beast, brute, creature, fauna
                     ⇨ organism, being
                       ⇨ living thing, animate thing
                         ⇨ object, physical object
                         ⇨ entity
```

```
1 sense of big cat

Sense 1
big cat, cat
     ⇨ leopard, Panthera pardus
     ⇨ snow leopard, ounce Pantera uncia
     ⇨ jaguar, panther, Panthera onca, Felis onca
     ⇨ lion, king of beasts, Panthera leo
     ⇨ tiger, Pantera tigris
     ⇨ liger
     ⇨ tiglon, tigon
     ⇨ cheetah, chetah, Acinonyx jubatus
     ⇨ saber-toothed tiger, sabertooth
```

Figure 2.9: Example of the hypernym/hyponym relationship: "snow leopard" (top image) is the specific term (hyponym) for "big cat" (bottom image).

- Hypernym/hyponym relation: a hypernym is a generic term for a class of specific items; a hyponym is a specific term for a member of a class. In other words, a hyponym is a *kind of* hypernym.

  E.g., **Snow leopards** are among the most beautiful **big cats** in the world. Here, "snow leopard" is a *kind of* "big cat", thus defining "snow leopard" as the hyponym and "big cat" as the hypernym (Figure 2.9).

- Siblings: words share the same hypernym.

  E.g. My **dog**, Samson, loves to play with my **cat**, Peace.

Sense 1
cat, true cat

- ⇨ feline, felid
  - ⇨ **carnivore**
    - ⇨ placental, placental mammal, eutherian, eutherian mammal
      - ⇨ mammal
        - ⇨ vertebrate, craniate
          - ⇨ chordate
            - ⇨ animal, animate being, beast, brute, creature, fauna
              - ⇨ organism, being
                - ⇨ living thing, animate thing
                  - ⇨ object, physical object
                    - ⇨ entity

Sense 1
dog, domestic dog, Canis familiaris

- ⇨ canine, canid
  - ⇨ **carnivore**
    - ⇨ placental, placental mammal, eutherian, eutherian mammal
      - ⇨ mammal
        - ⇨ vertebrate, craniate
          - ⇨ chordate
            - ⇨ animal, animate being, beast, brute, creature, fauna
              - ⇨ organism, being
                - ⇨ living thing, animate thing
                  - ⇨ object, physical object
                    - ⇨ entity

Figure 2.10: Example of the sibling relationship: top image show the hypernyms for "dog"; the bottom shows the hyponymns for "cat".

In Figure 2.10, the words "dog" and "cat" share the same hypernym, "carnivore". The hypernyms for "dog" and "cat" are identical after the first two levels.

Each relation is assigned a score and lexical chains are generated. Scores for relations are based on the distance between two concepts in WordNet's hierarchy: identical and synonym relations score 1.0, hypernym/hyponym relations score 0.5, and sibling relations score 0.33 (Kolla, 2004). Each chain's score is based on the sum of each relation within the lexical chain (equation 2.1):

$$score(chain) = \sum_{i=1}^{n} (score\ (R_i)) \tag{2.1}$$

where R_i is the semantic measure between two members of the lexical chain (Kolla, 2004). Generated lexical chains are written to a file to be used in the Summarizer's *scoring* stage.

How the Summarizer Chooses Salient Sentences

The Summarizer employs the sentence extraction method to generate summaries. Recall that documents are partitioned into topical segments. Each segment must be scored and ranked. Salient sentences are chosen from the top ranking segments to include as the summary. This section briefly explains the scoring procedure. First, each segment is ranked based on equation 2.2:

$$score(seg_i) = \sum_{j=1}^{m} \frac{score\ (chainMember_j, seg_i)}{s_j} \tag{2.2}$$

where $score\ (\ chainMember_j\ , seg_i\ )$ is the number of occurrences of a $chainMember_j$ in seg_i, m is their number, and s_i is the number of segments in which $chainMember_i$ occurs (Chali, Kolla, Singh, & Zhang, 2003, p. 149). Segments are sorted with the highest scoring at the top. Salient sentences are chosen from the top n segments.

Second, each sentence is ranked based on equation 2.3:

$$score(sentence_i) = \sum_{i=1}^{m} \frac{score\ (chainMember_j, sentence_j)}{sentences_j} \tag{2.3}$$

where $score(sentence_i)$ is the score of $sentence_i$, $score(chainMember_j,\ sentence_i)$ is the number of occurrences of the *chainMember*$_j$ in *sentence*$_i$, and *sentences*$_j$ is the number of sentences in that segment in which the *chainMember*$_j$ occurs (Chali, & Kolla, 2004).

The final stage extracts salient sentences for creating the summary. To review, each segment has been ranked and sorted with the top segment placed at the top. The same has been done for each sentence within these segments. The top ranking sentences are repetitively extracted from the top ranking segments until the desired summary length is achieved. For instance, the structure would include the first ranked sentence from the first ranked segment, the first ranked sentence from the second ranked segment, and so forth. Sentences are then sorted chronologically based on their position within the original text. For example:

Let: *Sentence A* = the first ranking sentence from the first ranking segment,

Sentence C = the first ranking sentence from the third ranking segment,

If *Sentence A* is physically positioned below *Sentence C* in the source text, then *Sentence A* will also be physically positioned below *Sentence C* in the summary, even though *Sentence A* has a higher score than *Sentence C.* Retaining the sentence positions helps the summary remain coherent.

2.5 Summary

This chapter supplied background information pertaining to DLs and automatic text summarization with an emphasis on the two systems used in this thesis: the Greenstone Digital Library Software Suite and the University of Lethbridge Summarizer.

A DL is a collection of services and information objects for storing, accessing, and retrieving digital objects. Greenstone is a complete software suite for building and distributing DL collections. Information is indexed, structured, published, and presented in a fashion to allow users to search or browse collections in an organized manner. Extracted metadata from source documents are indexed for searching and browsing.

Automatic text summarization extracts salient information from its source text and presents it in a usable, condensed form pertinent to the user's needs. The Summarizer creates generic, indicative summaries from single-documents. It employs lexical chains to detect topically related ideas. Lexical chains are used to identify salient sentences, which are extracted and combined to generate summaries. The next chapter describes the coupling of Greenstone and the Summarizer.

Chapter 3

Implementing the University of Lethbridge Summarizer in Greenstone

The purpose of this chapter is to describe the process used in installing the Summarizer in Greenstone. It is divided into three parts: how to install the Summarizer, issues that arose, and plugin options. First, Section 3.1 will explain Greenstone plugins, explore the *BasPlug* plugin, and discourse how to install the Summarizer as a package. Second, Section 3.2 will examine issues that arose after the Summarizer was put in place. Third, Section 3.3 will elaborate on the specific options within the Summarizer, what they are and where they should be used.

3.1 How to Install the Summarizer into Greenstone

This section describes the steps taken to install the Summarizer into Greenstone. Greenstone is easily extensible by adding new modules of code, referred to as plugins and classifiers.

> Plugins parse documents, extracting the text and metadata to be indexed. Classifiers control how metadata is brought together to form browsable data structures. Both are specified in an object-oriented framework using inheritance to minimize the amount of code written (Witten, McNab, Boddie, & Bainbridge, 2000, p. 7).

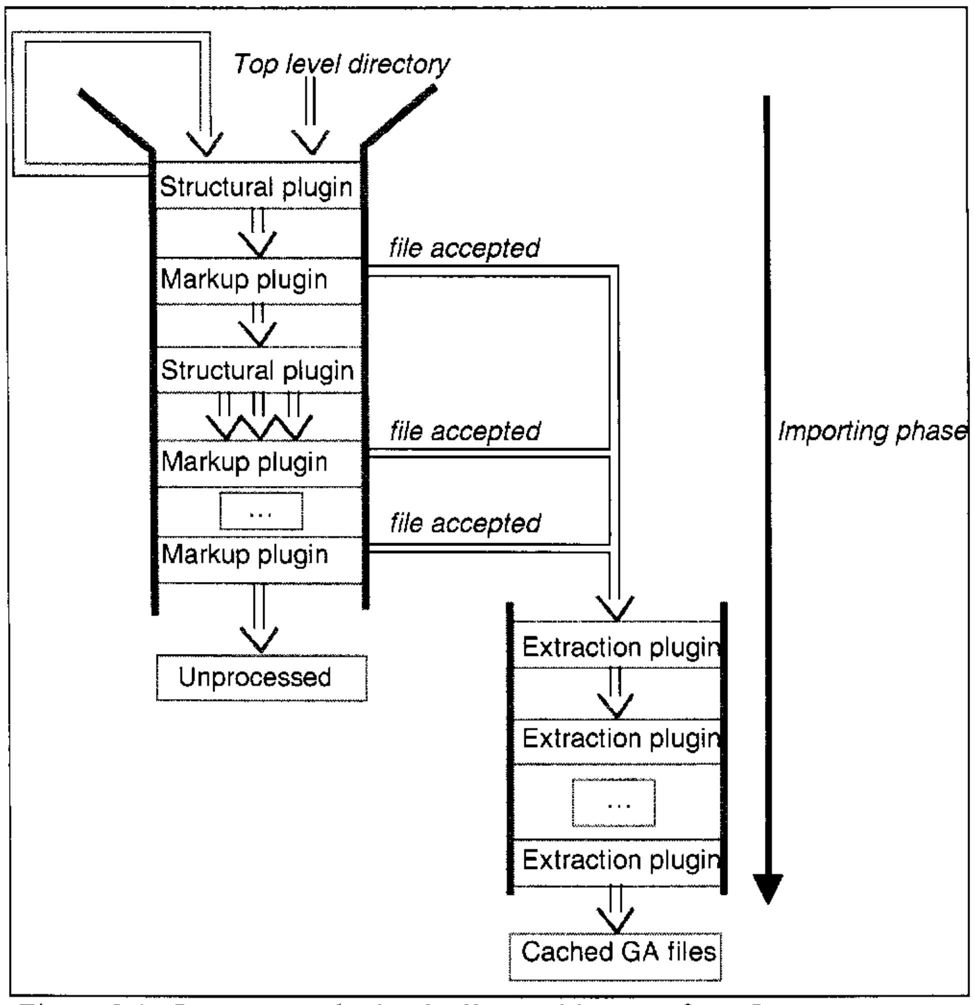


Figure 3.1: Greenstone plugin pipeline architecture from *Importing documents and metadata into digital libraries: Requirements analysis and an extensible architecture* (Witten, Bainbridge, Paynter, & Boddie, 2002a, p. 403).

The Summarizer extracts sentences from the text for generating a summary and sets it as metadata to be indexed, thus the Summarizer is implemented as a Greenstone plugin. This section will go into depth on what Greenstone plugins are, examine the *BasPlug* plugin, explore the Summarizer as a plugin option to the *BasPlug* plugin, and explain how to install the Summarizer as a Greenstone package.

```
creator
maintainer
public          true

indexes         document:text document:Title document:Source
defaultindex    document:text

plugin      ZIPPlug
plugin      GAPlug
plugin      TEXTPlug
plugin      HTMLPlug
plugin      EMAILPlug
plugin      PDFPlug      -convert_to_html -complex
plugin      RTFPlug
plugin      WordPlug
plugin      PSPlug
plugin      ArcPlug
plugin      RecPlug      -use_metadata_files

classify    AZList -metadata Title
classify    AZList -metadata Source

collectionmeta collectionname      "demo"
collectionmeta iconcollection      ""
collectionmeta collectionextra     ""
collectionmeta .document:text      "text"
collectionmeta .document:Title     "titles"
collectionmeta .document:Source    "filenames"
```

Figure 3.2: Collection configuration file.

3.1.1 How Greenstone Processes Documents

To review, plugins are used in Greenstone's import phase (Chapter 2, p. 22) to process imported documents, extract text, and set it as metadata to be indexed. Figure 3.1 illustrates the architecture of the pipeline flow of plugins (Witten, Bainbridge, Paynter, & Boddie, 2002a). They are processed in their order of appearance in the collection's configuration file (Figure 3.2). The top-level directory enters the first plugin of the pipeline. Each file is passed down the pipeline until it meets a plugin that can process it. If there is no applicable plugin, the file is not processed, a warning is issued, and the system progresses to the next file. Plugins use inheritance in order to reduce code redundancy and increase flexibility. This allows files to be processed by multiple plugins. There are three types of plugins (Witten, Bainbridge, Paynter, & Boddie, 2002b):

1. Structural plugins: function on generic file structures. For example, a filename that refers to a directory is processed by a structural plugin that traverses the directory and passes each file to its applicable plugin.
2. Mark-up plugins: process particular file formats or metadata types. Mark-up plugins are sub-categorized as:
    a. Conversion plugins: use external programs to convert documents from a specific file type to a different one. A subsequent plugin then converts it to the standard Greenstone Archive Format. For example, the *PDFPlug* plugin is a mark-up plugin that processes PDF file formatted documents. Files can be converted to either HTML or text, depending on which is specified in the collection's configuration file (Chapter 2, p. 15). PDF documents are converted to HTML by specifying `plugin PDFPlug -convert_to html` in the collection's configuration file. PDFPlug calls *pdftohtml* (Kruk, 2002), an external program, to convert documents from the PDF file format to the HTML file format.
    b. Splitter plugins: process files containing multiple documents and separates these documents for further processing. For example, the *EMAILPlug* plugin processes email messages. If there are multiple email messages in the document, it splits each message into a separate document for further processing.

3. Extraction plugins: extract metadata from the document's plain text and insert this metadata into the Greenstone's internal archive format file. Both specific metadata and structural information relating to the collection's document content can be automatically extracted.

Each plugin specifies the file format(s) handled, how to parse it, and whether the plugin is recursive (Witten et al., 2000). For instance, *PDFPlug*, *HTMLPlug* and *WORDPlug* handle PDF, HTML, and Word files respectively. Plugins parse the imported documents to extract metadata from them. For example, the *HTMLPlug* extracts readily available metadata from HTML files based on their metadata tags (e.g. `<title>`) (Witten, Boddie, & Thompson, n.d.). *RecPlug* is an example of a recursive plugin. It guarantees each directory structure is traversed by expanding the subdirectories and passing their contents to the pipeline (Witten, Bainbridge, & Boddie, 2001, p. 295).

The Summarizer belongs to the extraction plugin category because it produces summaries from documents and sets these summaries as metadata. Since it requires plain text input, it can be used on any file format from which plain text can be extracted. For instance, since plain text can be extracted from HTML, the Summarizer works in conjunction with any plugin that converts a file from one format to the HTML format (e.g. PDFPlug).

Plugins use the inheritance framework and are derived from the *BasPlug* plugin; any option written for BasPlug is inherited by any plugin derived from it. Writing the Summarizer as an option to BasPlug allows it to be inherited by all plugins, thus enabling it to handle any file format processed by plugins in which

```
< !DOCTYPE GreenstoneArchive [
< !ELEMENT Section (Description,Content,Section*)>
< !ELEMENT Description (Metadata*)>
< !ELEMENT Content (#PCDATA)>
< !ELEMENT Metadata (#PCDATA)>
<ATTLIST Metadata name CDATA #REQUIRED>
]>
```

Figure 3.3: Data Type Definition (DTD) for the Greenstone Archive Format file.

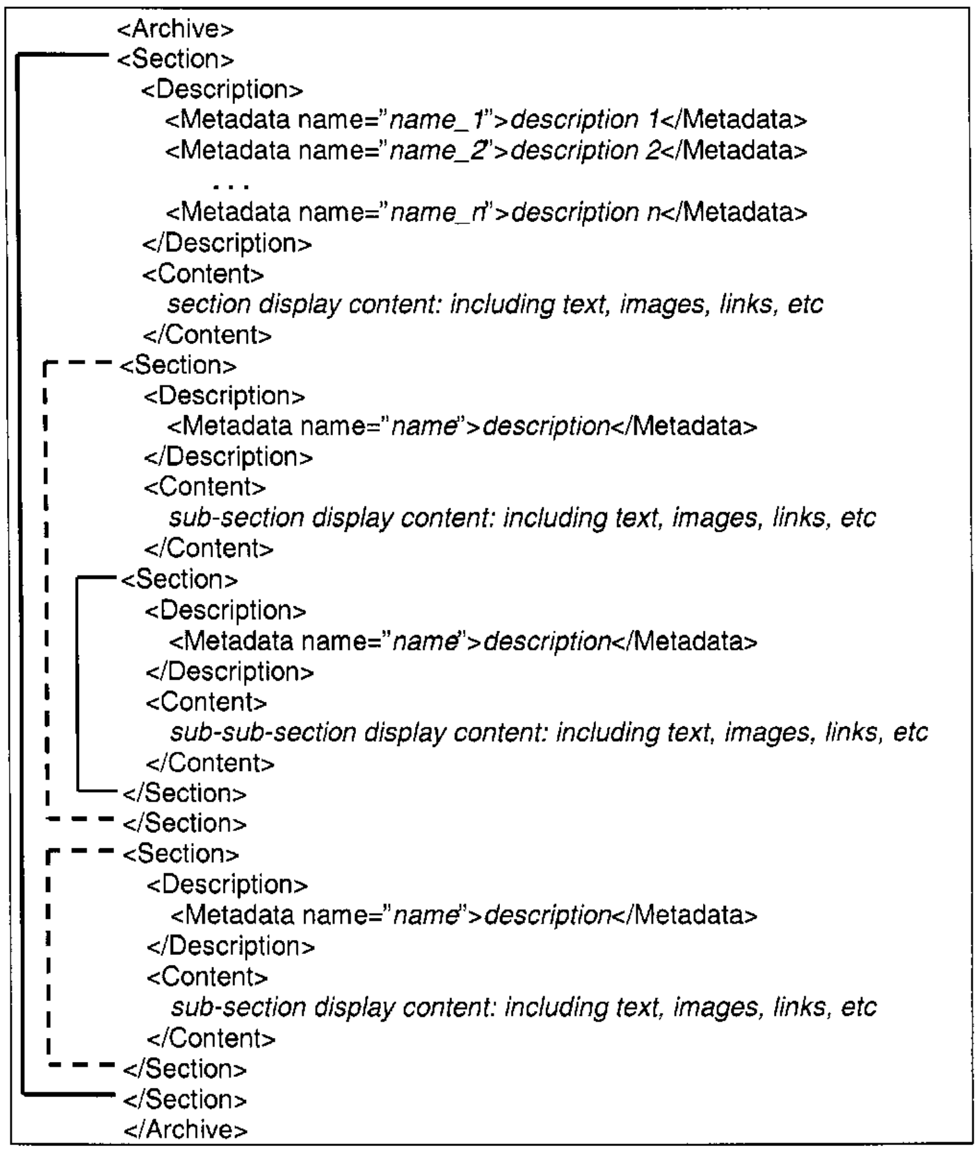
plain text or HTML can be extracted. This section will explore the BasPlug plugin in detail. The Summarizer module parses textual content from documents to produce summary metadata, excluding text such as headings, footnotes, figures, etc. Parsing ideal textual content for input will be discussed in Section 3.2.

3.1.2 Examining the BasPlug Plugin

As mentioned, Greenstone uses plugins to parse documents and extract metadata. This metadata can be used for indexing, searching and browsing. All plugins are written in the Perl language and are derived from BasPlug. According to the Greenstone Digital Library Developer's Guide (Bainbridge, McKay, & Witten, 2003a), BasPlug performs a variety of universally required operations:

- Converting documents of various file types to a standardized Greenstone Archive Format
- Managing document sections
- Assigning object identifiers (OID)

The Greenstone Archive Format is an XML style document that marks document sections and their metadata. Figure 3.3 is the document type definition (DTD) for the archive format. It uses an object hierarchy (Figure 3.4) with documents partitioned into sections. A document may contain only one section, or



```
<Archive>
<Section>
  <Description>
    <Metadata name="name_1">description 1</Metadata>
    <Metadata name="name_2">description 2</Metadata>
      . . .
    <Metadata name="name_n">description n</Metadata>
  </Description>
  <Content>
    section display content: including text, images, links, etc
  </Content>
<Section>
  <Description>
    <Metadata name="name">description</Metadata>
  </Description>
  <Content>
    sub-section display content: including text, images, links, etc
  </Content>
<Section>
  <Description>
    <Metadata name="name">description</Metadata>
  </Description>
  <Content>
    sub-sub-section display content: including text, images, links, etc
  </Content>
</Section>
</Section>
<Section>
  <Description>
    <Metadata name="name">description</Metadata>
  </Description>
  <Content>
    sub-section display content: including text, images, links, etc
  </Content>
</Section>
</Section>
</Archive>
```

Figure 3.4: Greenstone Archive Format architecture.

many sections. Each section of the document is recognized by a start tag, `<Section>`, and a corresponding end tag, `</Section>`.

Figure 3.4 illustrates how a document can contain both multiple sections and nested sections; the document in Figure 3.5 contains only one section. An example of a document containing multiple sections is a book with chapters where each chapter is a separate section. Each section, which can be partitioned into applicable sub-sections, is composed of `<Description>`, and

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE Archive SYSTEM
"http://greenstone.org/dtd/Archive/1.0/Archive.dtd">
<Archive>
  <Section>
    <Description>
      <Metadata name="lastmodified">1112286995</Metadata>
      <Metadata
name="gsdlsourcefilename">import/telusplanet.net/public/mtoll/birthcom.htm<
Metadata>
      <Metadata name="gsdldoctype">indexed_doc</Metadata>
      <Metadata name="Language">en</Metadata>
      <Metadata name="Plugin">HTMLPlug</Metadata>
      <Metadata name="FileSize">38979</Metadata>
      <Metadata name="Source">birthcom.htm</Metadata>
      <Metadata
name="URL">http://www.telusplanet.net/public/mtoll/birthcom.htm
</Metadata>
      <Metadata name="Keyword">southern alberta</Metadata>
      <Metadata name="Title">Birth of a Community</Metadata>
      <Metadata name="FileFormat">HTML</Metadata>
      <Metadata name="TextSummary">Early traders knew the country, but
this knowledge had not been recorded on maps. The trading post of Fort
Whoop-Up, established in 1867 near the junction of the St. Mary's and Old
Man River, was an early settlement in Southern Alberta. The North West Coal
Co. had a contract to deliver 5,000 tons of coal to the railroad at Medicine Hat,
but the barges only delivered 3,000 tons on the contract., The North West
Coal and Transportation Co. received 320 acres of land per mile of road, and
an option to purchase a million acres at $10.00 per acre, for building the
railway from Lethbridge to Medicine Hat and Great Falls.</Metadata>
      <Metadata
name="Identifier">HASHd29cf4fb9d7943702637cf</Metadata>
      <Metadata name="gsdlassocfile">hline.gif:image/gif:</Metadata>
      <Metadata name="gsdlassocfile">rd_ball.gif:image/gif:</Metadata>
      <Metadata name="assocfilepath">HASHd29c.dir</Metadata>
    </Description>
    <Content>
      This is the text for the HTML document.
    </Content>
  </Section>
  </Archive>
```

Figure 3.5: Greenstone Archive Format file written in XML using Figure 3.3 DTD.

`<Content>`. `<Description>` contains metadata extracted from document text `<Content>` contains document text in the HTML format used for displaying it in the DL. Figure 3.5 shows an example of a Greenstone Archive Format document consisting of one section, and a variety of metadata. Some metadata is extracted automatically from documents, such as the name of the plugin that processed the file (`HTMLPlug`), and the source file's name (`birthcom.htm`). Additional metadata can be specified manually, embedded in documents, available in metadata files, or supplied via spreadsheets.

Each document is assigned an Object Identifier (OID) to identify documents, sections and sub-sections. Greenstone provides two methods of computing an OID (Bainbridge et al., 2003a):

1. Hash: computed by hashing the contents of the imported document[2]
2. Incremental: computed by sequentially numbering each document in the order they are imported

The hash method is slower than incremental, but is advantageous for collections planning to add new documents in the future. The OID is identified by the `"Identifier"` metadata; for example, the OID in Figure 3.4 is `HASHd29cf4fb9d7943702637cf`.

3.1.3 Writing the Summarizer as a BasPlug Option

Although plugins can contain their own exclusive options, options written for BasPlug are inherited by all the plugins. This makes writing the Summarizer as a BasPlug option advantageous--all plugins can adopt its functions. Presently the Summarizer has only been tested with .txt, .html, and .pdf documents. It would not be difficult to adjust the Summarizer to accommodate other document types (e.g. .ps, .doc, etc.). When a BasPlug option is defined in the collection configuration file, BasPlug passes this option to the applicable Perl module, in

[2] Gordon Paynter (June, 2000) described the algorithm for calculating the hash string from the file contents in Greenstone's *hashfile.cpp* source code: "the file is treated as a large base 256 number (each char is a digit), and the hash value is the remainder when this number is divided by a very large prime" (l. 148).

this case, the Summarizer module (*summary.pm*). A number of options have been written for the Summarizer module and will be discussed in detail in Section 3.3. The purpose of this section is to explain how BasPlug calls the Summarizer Perl module and adds metadata to the archive file.

To produce a summary, the configuration file must specify the `extract_summary` option in the plugin declaration:

```
  plugin HTMLPlug -extract_summary
```

BasPlug reads the `-extract_summary` option and performs these steps:

1. Call *summary.pm* to perform the initialization routine
2. Parse applicable summary options that have been declared in the configuration file
3. Obtain the source filename
4. Obtain the document text
5. Set the applicable summary options that have been declared in the configuration file
6. Call *summary.pm* to produce a summary of the document text
7. Set the summary as `TextSummary` metadata in the archive file
8. Set all other applicable metadata pertaining to the specified summary options
9. Call *summary.pm* to perform the finalization routine

3.1.4 Displaying Information for the User

Greenstone includes a Perl script that displays user information, and specific and general options available for the specified plugin:

```
pluginfo.pl <plugin name>
```

The options available for the Summarizer are:

1. `BasPlug.extract_summary`: extracts summary from the document text and sets it as the `TextSummary` metadata
2. `BasPlug.extract_summary_section`: extracts summary from the section-level document text and sets it as the section-level `TextSummary` metadata
3. `BasPlug.extract_nouns`: extracts nouns from the document text and sets it as the `Nouns` metadata
4. `BasPlug.extract_np`: extracts noun phrases from the document text and sets it as the `NP` metadata
5. `BasPlug.summary_length`: sets the Summarizer to produce summaries that are `<int>` bytes in length. If not set the default value is 500 bytes.
6. `BasPlug.summary_by_size`: sets the flag to produce summaries based on the first `<int>` bytes of the document text. If not set the summary is produced based on the entire document.
7. `BasPlug.summary_first_sentence`: sets the flag to use the first sentence of the text as the first sentence of the summary
8. `BasPlug.summary_no_unicode`: sets the flag to omit any sentences with Unicode in the summary

These options will be explored in detail in Section 3.3.

3.1.5 How to Install the Summarizer as a Package

Greenstone's *packages* directory contains code used by Greenstone but developed by other resources (Bainbridge et al., 2003a). Tools used by the Summarizer are included in Greenstone as a package. They are found in their own directory inside Greenstone's packages directory (e.g. `gsdl/etc/packages/summarizer`). This section reviews what these tools are, where to download them, and how to install them for use in the Summarizer.

3.1.6 Required Tools for the Summarizer

The Summarizer uses tools to create summaries: *WordNet-2.0*, *WordNet-QueryData-1.3.1*, *OAK*, *C99*, and *segsplitter*. Initially, these tools need to be downloaded and installed. Recall from Chapter 2 that *WordNet* (Miller et al., 2005) is a lexical database; it is available at *http://wordnet.princeton.edu*. Installation instructions are also available on this site. After downloading and unzipping WordNet, ensure the following environmental variables are set:

- `WNHOME`: contains the path to WordNet's directory (e.g. `C:/WordNet-2.0`).
- `WNSEARCHDIR`: contains the path to WordNet's dictionary (e.g. `C:/WordNet-2.0/dict`).

Ensure WordNet's bin directory is added to the environmental path (e.g. `C:/WordNet-2.0/bin`). WordNet's makefile requires editing to certify that `WNHOME` matches the same path as the environmental variable (above).

WordNet-QueryData (Rennie, 2002) provides a direct interface to WordNet's semantic lexicon. It is freely downloadable from *http://search.cpan.*

org/dist/WordNet-QueryData/QueryData.pm; installation instructions are also available at this site.

The *OAK* system (Sekine, 2002b) is a total English analyzer that was developed by Sekine at New York University. The software is available for download at *http://www.cs.nyu.edu/~sekine/PROJECT/OAK*. To use OAK in Greenstone, extract it to the Summarizer directory inside Greenstone's packages directory (e.g. `C:/gsdl/etc/packages/summarizer`). Edit the parameter file for the OAK system (`C:/gsdl/etc/packages/summarizer/OAK0.1/src/oak.prm`) to include the path to OAK's data directory. For example:

```
############################
#   Knowledge file name
############################

DICTIONARY_FILENAME     C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS021.dic
POSTAGGER_FILENAME      C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS002.pos
CHUNKERQUAD_FILENAME    C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS002.chq
CHUNKER_FILENAME        C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS002.chk
NEHIERARCHY_FILENAME    C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS021.neh
NEDICT_FILENAME         C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS021.ned
NE_FILENAME             C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS021.ner
DEPENDENCY_FILENAME     C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS003.dep
HEADTABLE_FILENAME      C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS002.htb
FUNCTAGS_FILENAME       C:/gsdl/etc/packages/summarizer/
                        OAK0.1/data/WS008.fnc
```

C99 (Choi, 2000b) is an algorithm for linear topic segmentation. Choi developed it at Manchester University and is available at http://www.freddychoi.me.uk/. The C99 script (e.g. `C:/gsdl/etc/packages/summarizer/`

`naacl00Exp/bin/C99`) requires editing to include the Summarizer path:

```
java-cp C:/gsdl/etc/packages/summarizer/naacl00Exp/
bin
```

Segsplitter splits the segments produced by C99 into separate files, which are used throughout by the Summarizer. Kolla developed the segsplitter at the University of Lethbridge and is included as part of the Summarizer package. The program should be copied to Greenstone's bin directory (e.g. `C:/gsdl/bin/linux/segsplitter`).

3.2 Issues When Merging the Summarizer with Greenstone

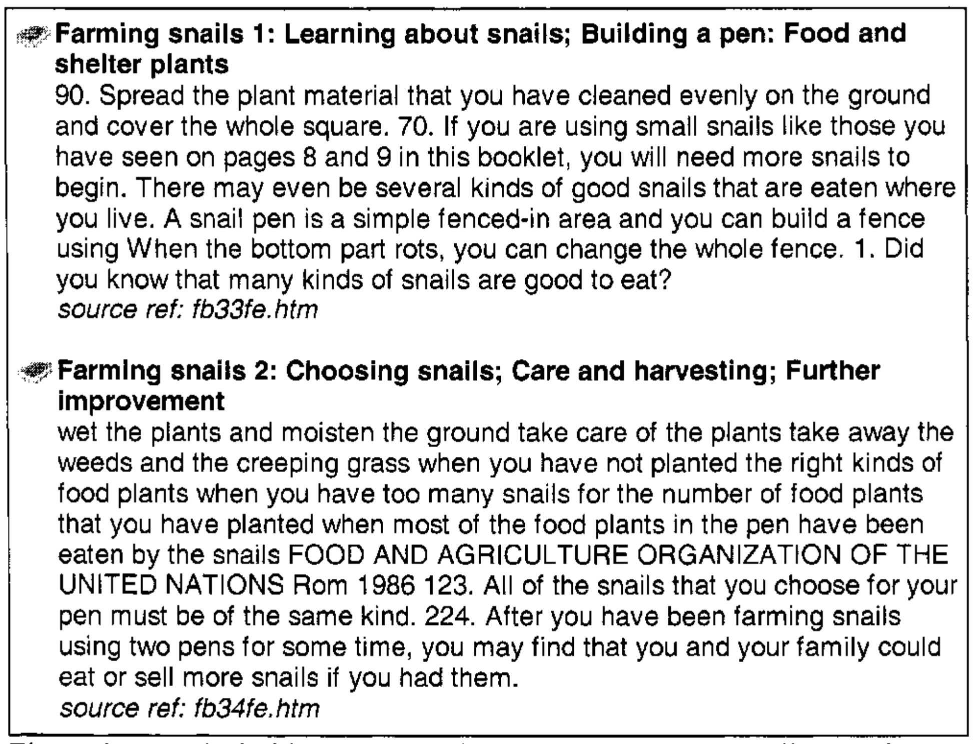
Originally, the Summarizer was to be treated as a black box. Unfortunately, issues arose when plugging the Summarizer into Greenstone and it was necessary to adjust the code:

1. Document formats: although the Summarizer was designed for processing plain text documents, additional document formats (.txt, .html, and .pdf) were required
2. Clean input: undesirable input such as typos, tags, headings, lists, footnotes, references, diagrams/figures, data, Unicode, and entities were not handled by the Summarizer
3. *Tokenizer* failure: extremely large strings and those with a combination of characters (tokens) more than fifteen cause the *tokenizer* to die
4. *Segmenter* failure: extremely large files result in the *segmenter* running out of memory
5. WordNet failure: uninitialized variables, extra spaces, and entities were

not handled, thus producing a "word area exhausted" error in WordNet

DL collections may consist of a number of text-based source document formats, including text, HTML, and PDF. Since the Summarizer was originally designed to process plain text, it was necessary to add additional processing functionalities for these formats. As mentioned in Chapter 2, Greenstone utilizes different plugins for converting documents of various file types into a standard format. Plain text can be extracted for use as input for the Summarizer.

The Summarizer was initially designed to work with clean input; specifically datasets consisting of newspaper articles in plain text format. Documents in a typical DL collection may not necessarily be as "clean" as in the datasets used to evaluate the Summarizer. Text sent to the Summarizer without first being cleaned may result in producing summaries that include objectionable text such as typos, headings, lists, or page numbers. The first summary in Figure 3.6 includes list numbers; the second summary includes a typo in which the first letter of the first sentence ("wet") is not capitalized. Although typos are not repairable at this time, other undesirable text such as headings and page numbers can be removed. For example, the summary in Figure 3.6 includes a heading ("FOOD AND AGRICULTURE ORGANIZATION OF THE UNITED NATIONS") and page numbers ("123", "224"). Ideally, clean text would include only the document content without spelling, grammar, or punctuation errors. Before entering the Summarizer, text can be cleaned by removing unwanted text, such as formatting and mark-up language tags, headings, footnotes, references, symbols, and lists.


Farming snails 1: Learning about snails; Building a pen: Food and shelter plants
90. Spread the plant material that you have cleaned evenly on the ground and cover the whole square. 70. If you are using small snails like those you have seen on pages 8 and 9 in this booklet, you will need more snails to begin. There may even be several kinds of good snails that are eaten where you live. A snail pen is a simple fenced-in area and you can build a fence using When the bottom part rots, you can change the whole fence. 1. Did you know that many kinds of snails are good to eat?
source ref: fb33fe.htm

Farming snails 2: Choosing snails; Care and harvesting; Further improvement
wet the plants and moisten the ground take care of the plants take away the weeds and the creeping grass when you have not planted the right kinds of food plants when you have too many snails for the number of food plants that you have planted when most of the food plants in the pen have been eaten by the snails FOOD AND AGRICULTURE ORGANIZATION OF THE UNITED NATIONS Rom 1986 123. All of the snails that you choose for your pen must be of the same kind. 224. After you have been farming snails using two pens for some time, you may find that you and your family could eat or sell more snails if you had them.
source ref: fb34fe.htm

Figure 3.6: Undesirable text appearing in summaries: typos, list numbers, headings, and page numbers.

HTML documents include numerous mark-up language tags. Content material appears between the `<body>...</body>` tags. Removing matter not defined within body tags significantly reduces objectionable text. In addition to defining formatting and section, tags help define headings (Figure 3.7). Removing the defined text further reduces non-content material. Unfortunately, not all text, such as headings, is defined by heading tags. For example, headings and bibliography information in Figure 3.8 are not defined by HTML, requiring additional procedures to find and eliminate possible headings and lists.

PDF documents are converted to HTML by specifying the PDFPlug plugin in Greenstone's configuration file:

```
plugin PDFPlug -convert_to html -complex
```

During this conversion, formatting tags are attached to the extracted text (Figure

```
<body BGCOLOR="d1fcfe" TEXT="#000000" LINK="#007afa"
VLINK="#00a3b0" ALINK="#de0000">
<h2><center><u><I><a name="beazcair.HTM">Cairn at Beazer, Alberta<br>
</a></I></h2></center></u>
<b><center>Taken from Cairn at Beazer, Alberta</center>
<p><center><I>Mark E. Beazer (Mark)</center></I><p>
```

Figure 3.7: HTML document with pre-defined heading tags.

```
<P ALIGN="JUSTIFY">FOOD AND AGRICULTURE ORGANIZATION OF THE
UNITED NATIONS</P>
<P ALIGN="JUSTIFY">Rome 1986</P>
<P ALIGN="JUSTIFY"></P>
<P ALIGN="JUSTIFY">P-69</P>
<P ALIGN="JUSTIFY">ISBN 92-5-102397</P>
<P ALIGN="JUSTIFY"></P>
<P ALIGN="JUSTIFY">© FAO 1986</P>
```

Figure 3.8: Headings appear in the HTML source without `<heading>` tags and bibliography text in an HTML document without pre-defined heading tags

3.9). PDFPlug assigns tags different from those in a typical human designed HTML document. Style classes and absolute positioning are used to define formats instead of high-level HTML structures such as lists, tables, or headings. This newly extracted text requires cleaning before the Summarizer can work on it. Unfortunately, simply removing formatting and mark-up tags is not sufficient; unwanted text such as headings and footnotes will remain (Figures 3.10a-c). An algorithm was devised that uses the style classes to calculate the most frequently used font size. The corresponding style formats are retained while text pertaining to all the other style formats is purged. Footnotes and headings typically use a different font size than that of the main body content; thus the algorithm works in eliminating them. Problems arise when the author chooses the identical font size for non-body content, such as headings and figures, as that of the body.

Unforeseen text, such as data from figures, may be also be extracted from PDF documents. Figure 3.11a shows a figure from (Choi, 2000a, p. 27) and

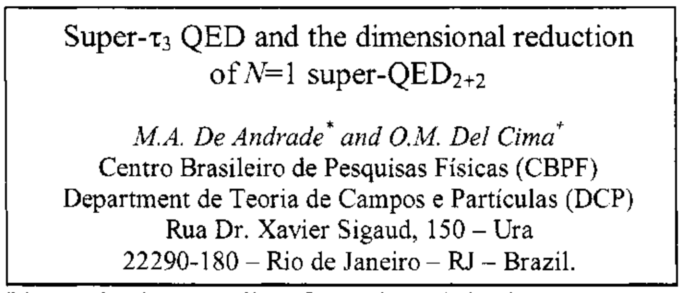
```
<DIV style="position:absolute;top:893;left:179"><nobr><span class="ft6">A
compact and very useful technique for working out representations of the super-
</span></nobr><DIV>
<DIV style="position:absolute;top:922;left:144"><nobr><span class="ft9">
symmetry algebra on fields was proposed by Salem and Strathdee [?, ?]:
superfields in<br>superspace. It is a particularly useful for N=1 theories, where
their superfield structure is <br>completely known. The well-known algebra
fulfilled by the generators of the supersymmetry<br> in D=2+2, P</span>
</nobr><DIV>
<DIV style="position:absolute;top:1018;left:281"><nobr><span class="ft4">μ
</span></nobr><DIV>
<DIV style="position:absolute;top:1009;left:292"><nobr><span class="ft6">, Q
</span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:325"><nobr><span class="ft4">α
</span></nobr><DIV>
<DIV style="position:absolute;top:1009;left:344"><nobr><span class="ft6">and
</span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:412"><nobr><span class="ft4">'
</span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:408"><nobr><span class="ft4">α
</span></nobr><DIV>
```

Figure 3.9: Formatting tags attached to text extracted from the PDF document *Super-τ_3 QED and the dimensional reduction of N=1 super=QED_{2+2}* (De Andrade & Del Cima, 1995).

Figure 3.11b is its extracted text. The question is, How to recognize and remove these undesirables? A simple algorithm to eliminate numbers, letters, and words appearing by themselves suffices. This also serves to omit erroneously extracted page numbers. Figure 3.11c shows how letters in (De Andrade & Del Cima, 1995, p. 3) can be mistakenly translated. The ASCII letters *f* and *i* are translated as the Latin small ligature *fi*, while the ASCII letters *f*, *f*, and *i* are translated as the Latin small ligature *ffi*. The part-of-speech tagger may incorrectly tag the word containing the offending symbol. For example, in the sentence:

> The idea of space-times with several time components and indefinite signature has been taken seriously into account since a self-dual Yang-Mills theory in 4-dimensions has been related to the Atiyah-Ward conjecture (De Andrade & Del Cima, 1995, p. 2).

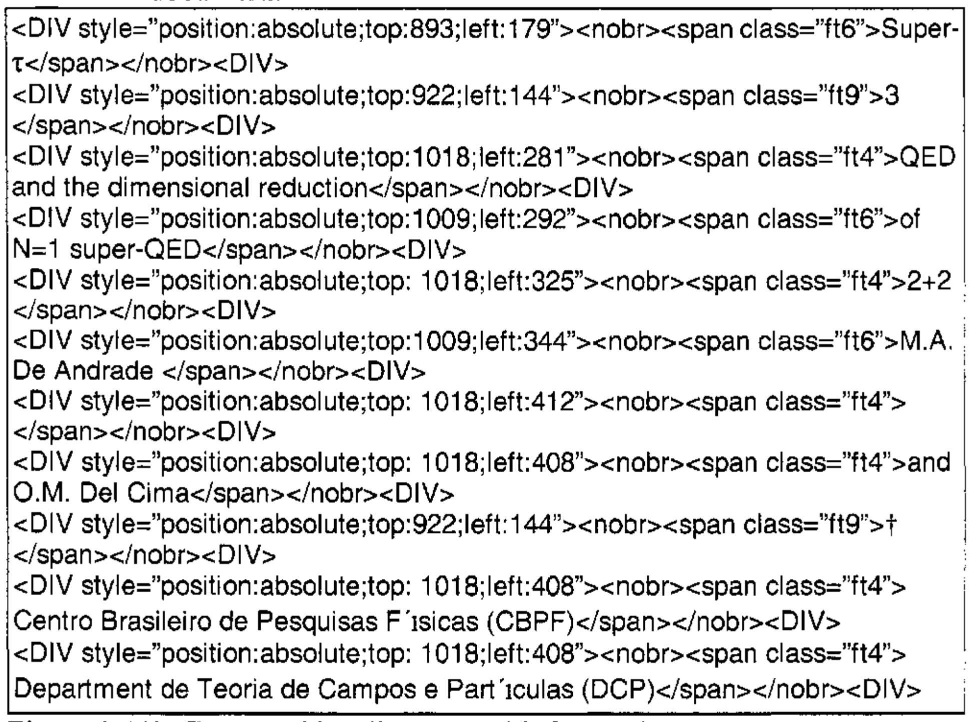
The word "indefinite", using the Latin small ligature *fi*, is tagged as a noun (Figure 3.12a). However, the same word, using the ASCII letters *fi*, is tagged as



Super-τ_3 QED and the dimensional reduction
of N=1 super-QED$_{2+2}$

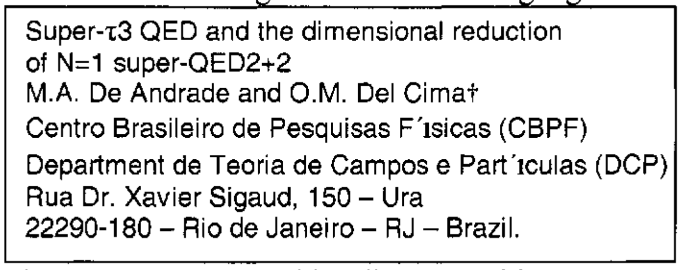
M.A. De Andrade[*] *and O.M. Del Cima*[†]
Centro Brasileiro de Pesquisas Físicas (CBPF)
Department de Teoria de Campos e Partículas (DCP)
Rua Dr. Xavier Sigaud, 150 – Ura
22290-180 – Rio de Janeiro – RJ – Brazil.

Figure 3.10a: Heading from the original PDF document.



```
<DIV style="position:absolute;top:893;left:179"><nobr><span class="ft6">Super-
τ</span></nobr><DIV>
<DIV style="position:absolute;top:922;left:144"><nobr><span class="ft9">3
</span></nobr><DIV>
<DIV style="position:absolute;top:1018;left:281"><nobr><span class="ft4">QED
and the dimensional reduction</span></nobr><DIV>
<DIV style="position:absolute;top:1009;left:292"><nobr><span class="ft6">of
N=1 super-QED</span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:325"><nobr><span class="ft4">2+2
</span></nobr><DIV>
<DIV style="position:absolute;top:1009;left:344"><nobr><span class="ft6">M.A.
De Andrade </span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:412"><nobr><span class="ft4">
</span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:408"><nobr><span class="ft4">and
O.M. Del Cima</span></nobr><DIV>
<DIV style="position:absolute;top:922;left:144"><nobr><span class="ft9">†
</span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:408"><nobr><span class="ft4">
Centro Brasileiro de Pesquisas F´ısicas (CBPF)</span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:408"><nobr><span class="ft4">
Department de Teoria de Campos e Part´ıculas (DCP)</span></nobr><DIV>
```

Figure 3.10b: Extracted heading text with formatting tags.



Super-τ3 QED and the dimensional reduction
of N=1 super-QED2+2
M.A. De Andrade and O.M. Del Cima†
Centro Brasileiro de Pesquisas F´ısicas (CBPF)
Department de Teoria de Campos e Part´ıculas (DCP)
Rua Dr. Xavier Sigaud, 150 – Ura
22290-180 – Rio de Janeiro – RJ – Brazil.

Figure 3.10c: Extracted heading text with formatting tags removed.

an adjective (Figure 3.12b). As mentioned in Chapter 2, the Summarizer creates lexical chains from nouns found within the document. Scores are assigned to segments and sentences based upon these chains. Falsely tagging an adjective as a

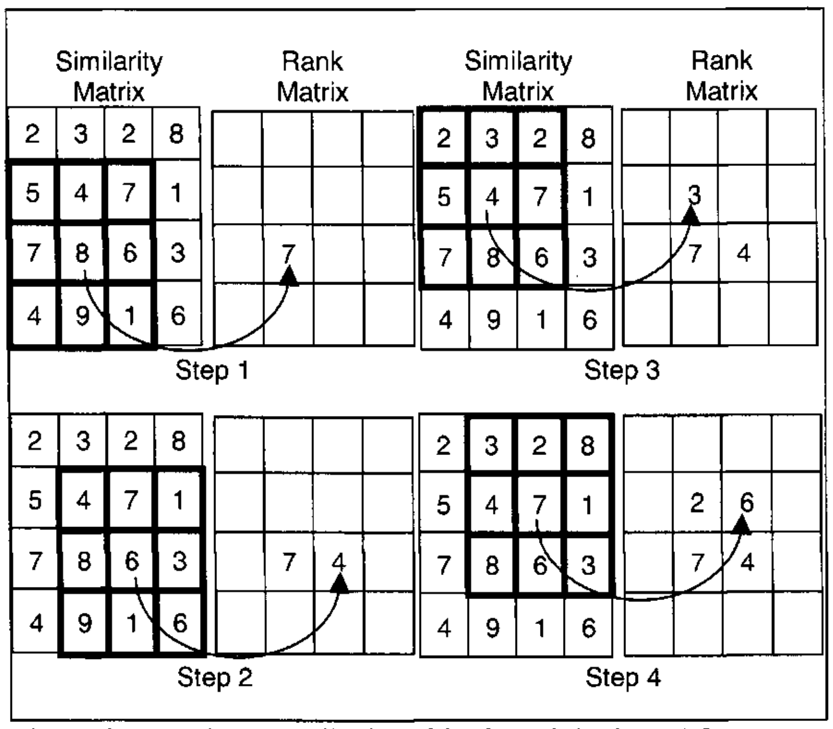


Figure 3.11a: Figure as displayed in the original PDF format.

```
<DIV style="position:absolute;top:893;left:179"><nobr><span
class="ft6">Similarity matrix</span></nobr><DIV>
<DIV style="position:absolute;top:922;left:144"><nobr><span class="ft9">3
</span></nobr><DIV>
<DIV style="position:absolute;top:1018;left:281"><nobr><span class="ft4">Rank
matrix</span></nobr><DIV>
<DIV style="position:absolute;top:1009;left:292"><nobr><span class="ft6">I 5
</span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:325"><nobr><span class="ft4">4 ~7
1 ~ </span></nobr><DIV>
<DIV style="position:absolute;top:1009;left:344"><nobr><span class="ft6">I 7 8
, 6 </span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:412"><nobr><span class="ft4">3 i
</span></nobr><DIV>
<DIV style="position:absolute;top: 1018;left:408"><nobr><span class="ft4">7
</span></nobr><DIV>
<DIV style="position:absolute;top:922;left:144"><nobr><span class="ft9">Step
1 </span></nobr><DIV>
```

Figure 3.11b: Extracted text from Figure 3.11a

> Superfields are analytic functions of superspace coordinates, which should be understood in terms of their power series expansions in θ and θ with coefficients which are themselves local fields over Minkowski [?, ?].

Figure 3.11c: ASCII letter combinations mistranslated as Latin ligatures: ASCII *fi* translated as the Latin small ligature *fi*; ASCII *ffi* translated as the Latin small ligature *ffi*.

```
[NP The/DT idea/NN] [PP of/IN] [NP space-times/NNS] [PP with/IN][NP
several/JJ time/NN components/NNS and/CC indeï¬nite/NN signature/NN]
[VP has/VBZ been/VBN taken/VBN] [ADVP seriously/RB][PP into/IN] [NP
account/NN] [PP since/IN] [NP a/DT self-dual/NNP Yang-Mills/NNP theory/NN]
[PP in/IN] [NP 4-dimensions/NNS][VP has/VBZ been/VBN related/VBN] [PP
to/TO] [NP the/DT Atiyah-Ward/NNP conjecture/NN] ./.
```

Figure 3.12a: Example of the word "indefinite" incorrectly tagged as an adjective.

```
[NP The/DT idea/NN] [PP of/IN] [NP space-times/NNS] [PP with/IN][NP
several/JJ time/NN components/NNS] and/CC [NP indefinite/JJ signature/NN]
[VP has/VBZ been/VBN taken/VBN] [ADVP seriously/RB][PP into/IN] [NP
account/NN] [PP since/IN] [NP a/DT self-dual/NNP Yang-Mills/NNP theory/NN]
[PP in/IN] [NP 4-dimensions/NNS][VP has/VBZ been/VBN related/VBN] [PP
to/TO] [NP the/DT Atiyah-Ward/NNP conjecture/NN] ./.
```

Figure 3.12b: Example of the word "indefinite" correctly tagged as a noun.

noun may skew the scoring process, resulting in sentences being extracted for use in the summary that may not otherwise have been selected. The solution is to replace the offending ligature symbols by their corresponding ASCII letter combinations.

Mentioned earlier was the problem of deleting headings and lists from the Summarizer's input. Ideally one could easily recognize headings and lists through mark-up language tags. What happens when the input does not contain tags, or the document's designer did not utilize these features? We have already seen how PDF documents converted to HTML include only basic formatting and style tags. This requires a procedure to recognize and erase headings and lists. Based on title capitalization rules (Shehata, 2000; *Writing Tips*, 1998), possible headings are identified and deleted. Similarly, grammar and punctuation rules (Straus, n.d.a, and Straus, n.d.b) are applied to identify and eliminate sentences containing long lists (e.g.: "At the store Mary bought apples, oranges, bread, bananas, eggs, and milk"). Numbered lists (e.g. Figure 3.6) are fixed by removing the offending numbers. Since the Tokenizer automatically places each sentence on its own line, the ideal place to perform these cleanings are on the tokenized file.

The Summarizer uses two programs developed outside of the University of Lethbridge; for more information, see Chapter 2:

1. OAK: developed and designed by Sekine as part of the Proteus Project at New York University. OAK is "a total English analyzer" (Sekine, 2002b, para. 2). The Summarizer utilizes OAK for tokenizing, chunking, and part-of-speech tagging.
2. C99 Segmenter: designed and developed by Choi, C99 "calculates the similarity between parts of a text using the cosine measure" (Dowman, Tablan, Cunningham, & Popov, 2005, p. 2).

Situations can occur to cause OAK's Tokenizer to fail. For instance, extremely large strings containing more than 4900 bytes cause segmentation faults. Splitting such strings into smaller pieces easily rectifies this. Sentences with more than fifteen tokens, and strings with three or more combined Unicode metasymbols causes the Tokenizer to die. The question arises: do we delete just the offending symbols or the entire sentence containing them? Solution: delete the entire sentence because only removing the offending symbols would alter the sentence's integrity. A drawback to eliminating entire sentences is that information could be lost. The advantage of removing the entire sentence is the Tokenizer will not fail, allowing the Summarizer to continue.

Occasionally, for unknown reasons, OAK will die. To ensure the system's robustness, an alarm was implemented to signal if OAK has not completed processing a document. In this event, the Summarizer moves onto the next document in the collection. Without the alarm the system will hang. The intent was to ensure that if problems arose, the system could simply abandon the offending document and move onto the next one.

The C99 Segmenter processes tokenized files. Extremely large files cause the C99 Segmenter to run out of memory. In these cases, the file is truncated to a pre-defined size or percentage. This truncation leads to information loss, which will be discussed in Chapter 4.

The Summarizer uses WordNet to define word senses (Chapter 2, p. 35). Initially the Summarizer did not ensure variables were initialized prior to sending them to WordNet, resulting in a "word area exhausted" error in WordNet. Forwarding entities and words containing extra spaces also produced problems. Additional areas in the Summarizer's code attempted using uninitialized variables. This required an extensive cleaning of the code itself to reduce system crashes and ensure its robustness.

3.3 How to Use the Summarizer

Since the Summarizer is implemented as a BasPlug option, it can be used in conjunction with any plugin specified. The available options include: `extract_summary`, `extract_summary_section`, `extract_nouns`, `extract_np`, `summary_length <int>`, `summary_by_size <int>`, `summary_first_sentence`, and `summary_no_unicode`. This section provides information on each of these.

3.3.1 Creating a Basic Summary

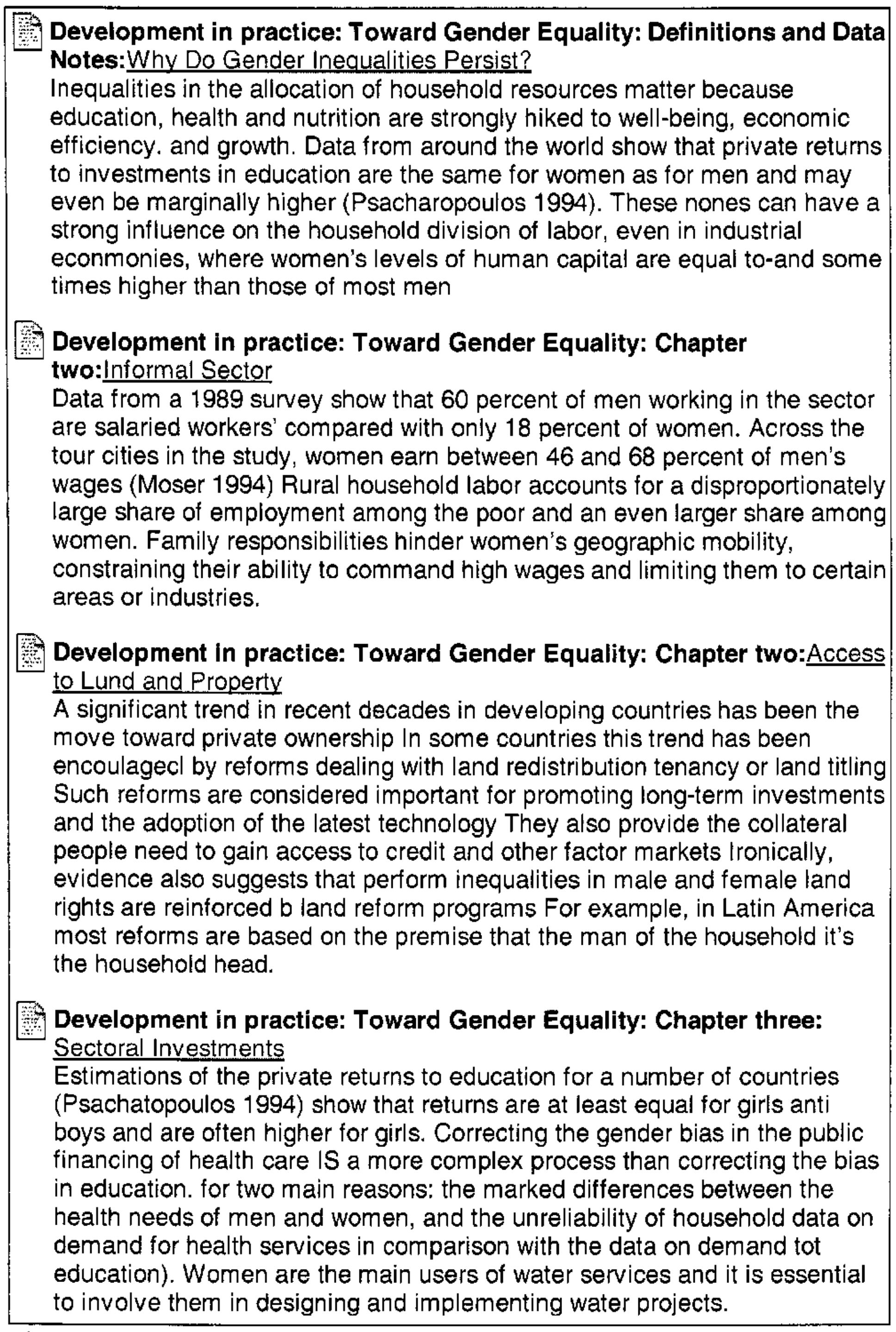
In order for a summary to be produced and used as metadata, the `extract_summary` option must be specified as an option to each applicable plugin declared in the collection's configuration file, for example:

```
plugin HTMLPlug    -extract_summary
plugin PDFPlug     -convert_to html -complex
-extract_summary
```

Use the `-convert_to html -complex` options (as shown above) when specifying the PDFPlug plugin. According to Boddie (2003, para. 15), using the default configuration, which does not include `-complex` option, will extract all the text from a PDF document, but it may not look like the original. Specifying the `-complex` option will extract all the text in a format similar to the original PDF document.

Additional options may be combined with `extract_summary`; they will be covered throughout this section. Specifying the `extract_summary` option without additional options produces a summary with the default values:

- `TextSummary` will be extracted based on the entire document text at the document-level
- `TextSummary` length is set to 500 bytes
- The first sentence of the `TextSummary` is calculated based on the ranking scheme described in Chapter 2; the document's first sentence will not automatically be set as the first sentence of the summary
- Any extracted sentences containing Unicode will be included in `TextSummary`



Development in practice: Toward Gender Equality: Definitions and Data Notes: Why Do Gender Inequalities Persist?
Inequalities in the allocation of household resources matter because education, health and nutrition are strongly hiked to well-being, economic efficiency. and growth. Data from around the world show that private returns to investments in education are the same for women as for men and may even be marginally higher (Psacharopoulos 1994). These nones can have a strong influence on the household division of labor, even in industrial econmonies, where women's levels of human capital are equal to-and some times higher than those of most men

Development in practice: Toward Gender Equality: Chapter two: Informal Sector
Data from a 1989 survey show that 60 percent of men working in the sector are salaried workers' compared with only 18 percent of women. Across the tour cities in the study, women earn between 46 and 68 percent of men's wages (Moser 1994) Rural household labor accounts for a disproportionately large share of employment among the poor and an even larger share among women. Family responsibilities hinder women's geographic mobility, constraining their ability to command high wages and limiting them to certain areas or industries.

Development in practice: Toward Gender Equality: Chapter two: Access to Lund and Property
A significant trend in recent decades in developing countries has been the move toward private ownership In some countries this trend has been encoulagecl by reforms dealing with land redistribution tenancy or land titling Such reforms are considered important for promoting long-term investments and the adoption of the latest technology They also provide the collateral people need to gain access to credit and other factor markets Ironically, evidence also suggests that perform inequalities in male and female land rights are reinforced b land reform programs For example, in Latin America most reforms are based on the premise that the man of the household it's the household head.

Development in practice: Toward Gender Equality: Chapter three: Sectoral Investments
Estimations of the private returns to education for a number of countries (Psachatopoulos 1994) show that returns are at least equal for girls anti boys and are often higher for girls. Correcting the gender bias in the public financing of health care IS a more complex process than correcting the bias in education. for two main reasons: the marked differences between the health needs of men and women, and the unreliability of household data on demand for health services in comparison with the data on demand tot education). Women are the main users of water services and it is essential to involve them in designing and implementing water projects.

Figure 3.13: A sample of summaries produced at the section-level.

3.3.2 Creating Summaries for each Section

In cases where a document is split into sections, summaries can be produced for each section by defining:

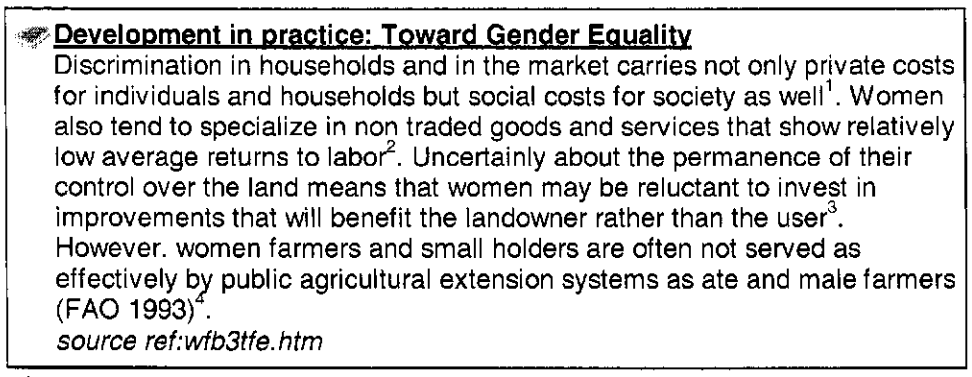

Development in practice: Toward Gender Equality
Discrimination in households and in the market carries not only private costs for individuals and households but social costs for society as well[1]. Women also tend to specialize in non traded goods and services that show relatively low average returns to labor[2]. Uncertainly about the permanence of their control over the land means that women may be reluctant to invest in improvements that will benefit the landowner rather than the user[3]. However. women farmers and small holders are often not served as effectively by public agricultural extension systems as ate and male farmers (FAO 1993)[4].
source ref:wfb3tfe.htm

Figure 3.14: Summary of a book produced at the document-level.

```
plugin HTMLPlug    -extract_summary
-extract_summary_section
```

in the collection's *collect.cfg* file. Declaring `-extract_summary_section` sets the flag for *summary.pm* to extract `TextSummary` metadata on each document's section in the collection. An example of a case where this option is useful is for a collection of books. Figure 3.13 contains a sample of section-level summaries created from Greenstone's demo collection.[3] As shown, each chapter is a section (e.g.: *Definitions and Data Notes*, *Chapter two*, and *Chapter three*). Chapters may be sub-sectioned (e.g.: *Chapter two*: *Informal Sector* and *Chapter two: Access to Lund and Property*). Text from each section is processed by the Summarizer to generate summary metadata, enabling each chapter to have its own summary.

Figure 3.14 illustrates the summary produced from the same book as the one seen in Figure 3.13. The difference is the summary in Figure 3.14 was produced at the document-level. Text from the entire book was processed by the

[3] Greenstone's demo collection is supplied with the Greenstone Digital Library software (http://sourceforge.net/projects/greenstone). The collection contains a variety of books. This section-level summary example is based on the *Development in Practice: Toward Gender Equality* book within the demo collection.

Summarizer to generate summary metadata for the entire book; this differs from the summaries in Figure 3.13 where each chapter of the book contains its own summary. Analyzing the document-level summary reveals sentences that are not seen in their corresponding section-level summaries:

- Sentence #1: "Discrimination in households and in the market carries not only private costs for individuals and households but social costs for society as well" is extracted from *Definitions and Data Notes: Why Do Gender Inequalities Persist?*
- Sentence #2: "Women also tend to specialize in non traded goods and services that show relatively low average returns to labour" is extracted from *Chapter two: Informal Sector*
- Sentence #3: "Uncertainly about the permanence of their control over the land means that women may be reluctant to invest in improvements that will benefit the landowner rather than the user" is extracted from *Chapter two: Access to Lund*[4] *and Property*
- Sentence #4: "However.[5] women farmers and small holders are often not served as effectively by public agricultural extension systems as ate[6] and male farmers (FAO 1993)" is extracted from *Chapter three: Sectoral Investments*

Recall from Chapter 2 (p. 31), the Summarizer splits text into segments,

[4] The Summarizer does not correct spelling mistakes that occur in the original documents. In *Chapter two: Access to Lund and Property*, the word "Land" is spelled as "Lund".

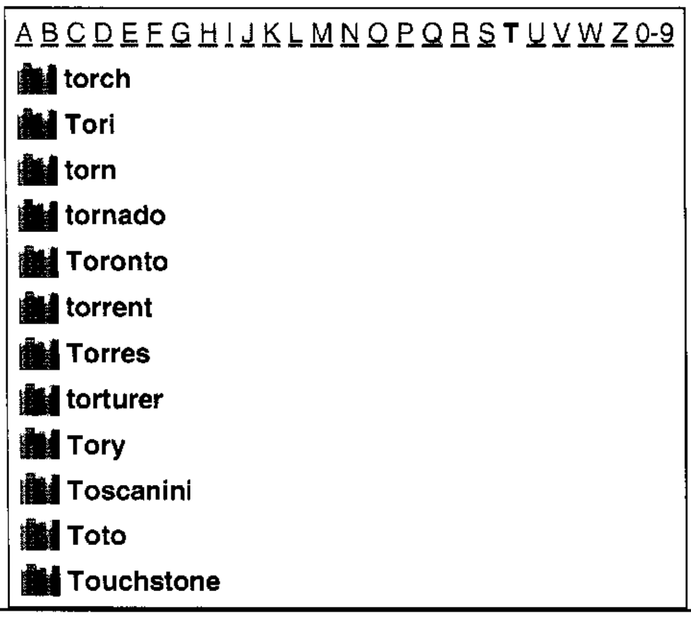
[5] The Summarizer does not correct punctuation mistakes that occur in the original documents. Sentence #4 reads "However. women" with a period between the words "However" and "women" where it should be a comma.

[6] The Summarizer does not correct spelling and grammar errors that occur in the original documents. In this instance, the word "ate" appears out of context.

ranks each of these segments, and assigns scores to each sentence within those segments. Sentences are ordered sequentially based on their scores, with the greatest scoring sentence at the top. The top sentences from the uppermost ranking segments are extracted for creating summaries. In the above example, when the book is processed at the document-level the entire text of the book is used to generate one summary. Based on the method the Summarizer uses to process text, as described above, sentences extracted for creating a document-level summary of the book come from multiple chapters. Conversely, when the same book is processed at the section-level, text from each chapter is individually used to generate multiple summaries, one summary per chapter. Sentences extracted for creating section-level summaries come from the chapter in which the text was extracted. This results in section-level summaries being more specific to their corresponding chapter. The document-level summary contains sentences from a variety of chapters, creating a more general summary of the book than the section-level ones.

Comparing the section-level summary for *Chapter two: Informal Sector* (Figure 3.13) with its corresponding document-level summary (Sentence #2 in Figure 3.14), confirms how a section-level summary can provide more detail than their document-level counterpart. Where the document-level summary presents a general comment on women, labour, and wages, the section-level summary supplies specific information on the same area:

- The document-level summary mentions that women specialize "in non traded goods and service" with "low average returns to labor"


A B C D E F G H I J K L M N O P Q R S **T** U V W Z 0-9

torch

Tori

torn

tornado

Toronto

torrent

Torres

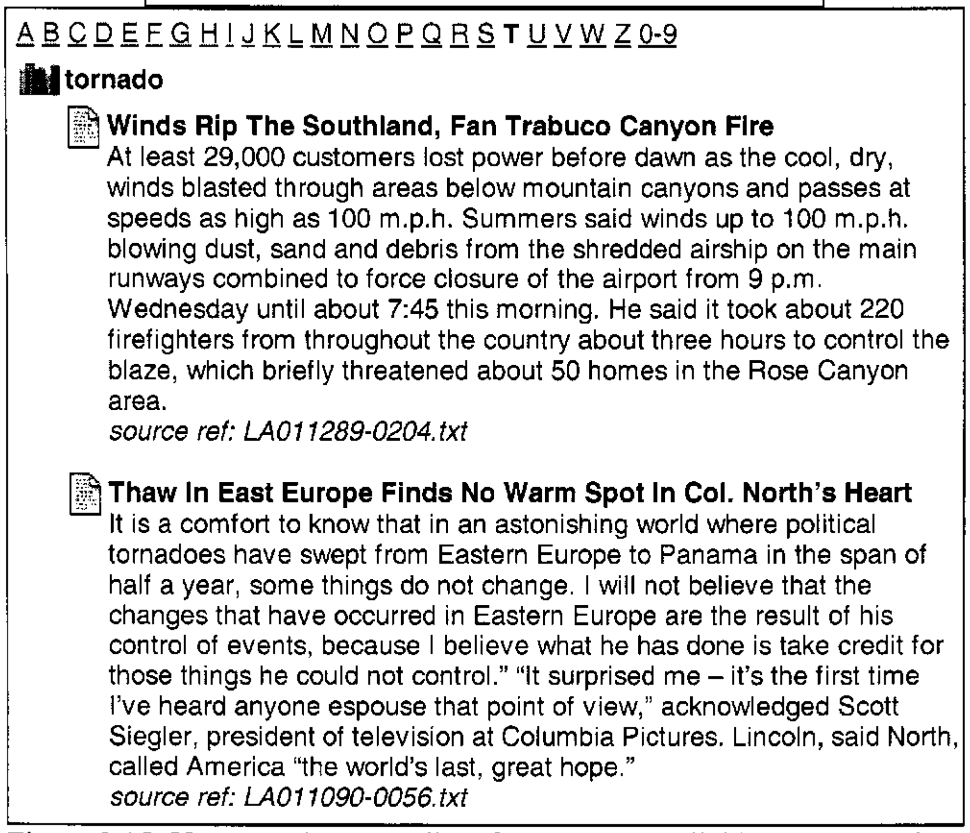
torturer

Tory

Toscanini

Toto

Touchstone


A B C D E F G H I J K L M N O P Q R S **T** U V W Z 0-9

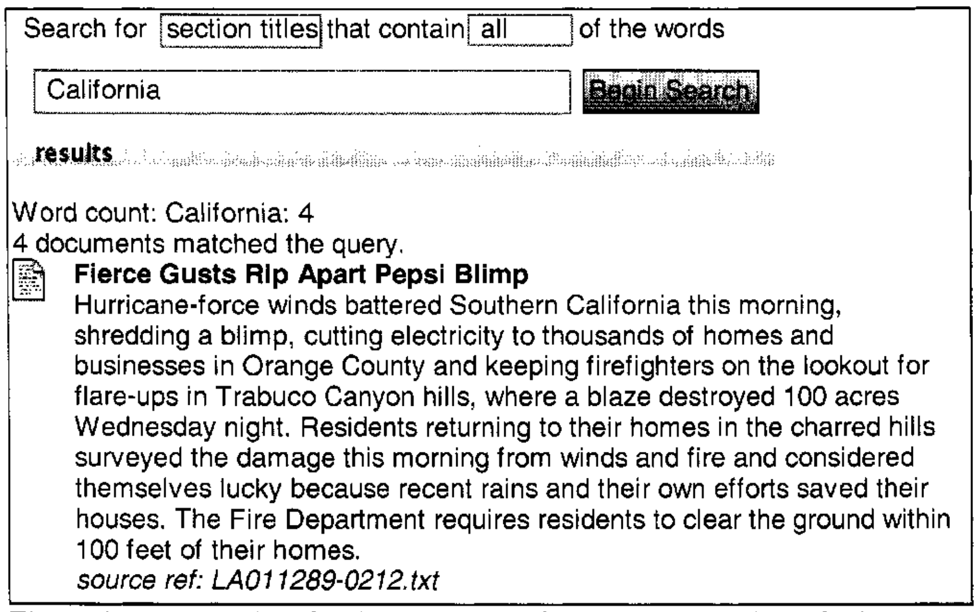
tornado

Winds Rip The Southland, Fan Trabuco Canyon Fire
At least 29,000 customers lost power before dawn as the cool, dry, winds blasted through areas below mountain canyons and passes at speeds as high as 100 m.p.h. Summers said winds up to 100 m.p.h. blowing dust, sand and debris from the shredded airship on the main runways combined to force closure of the airport from 9 p.m. Wednesday until about 7:45 this morning. He said it took about 220 firefighters from throughout the country about three hours to control the blaze, which briefly threatened about 50 homes in the Rose Canyon area.
source ref: LA011289-0204.txt

Thaw In East Europe Finds No Warm Spot In Col. North's Heart
It is a comfort to know that in an astonishing world where political tornadoes have swept from Eastern Europe to Panama in the span of half a year, some things do not change. I will not believe that the changes that have occurred in Eastern Europe are the result of his control of events, because I believe what he has done is take credit for those things he could not control." "It surprised me – it's the first time I've heard anyone espouse that point of view," acknowledged Scott Siegler, president of television at Columbia Pictures. Lincoln, said North, called America "the world's last, great hope."
source ref: LA011090-0056.txt

Figure 3.15: Users can browse a list of nouns (top). Clicking on "tornado" from it expands the list to reveal all documents in the digital collection containing "tornado" (bottom).

- The section-level summary presents particulars:
    - The percentage of salaried women workers is considerably lower


Search for section titles that contain all of the words

California

Begin Search

results

Word count: California: 4
4 documents matched the query.

Fierce Gusts Rip Apart Pepsi Blimp
Hurricane-force winds battered Southern California this morning, shredding a blimp, cutting electricity to thousands of homes and businesses in Orange County and keeping firefighters on the lookout for flare-ups in Trabuco Canyon hills, where a blaze destroyed 100 acres Wednesday night. Residents returning to their homes in the charred hills surveyed the damage this morning from winds and fire and considered themselves lucky because recent rains and their own efforts saved their houses. The Fire Department requires residents to clear the ground within 100 feet of their homes.
source ref: LA011289-0212.txt

Figure 3.16: Querying for the noun "California" returns a list of relevant documents.

(42%) than for men

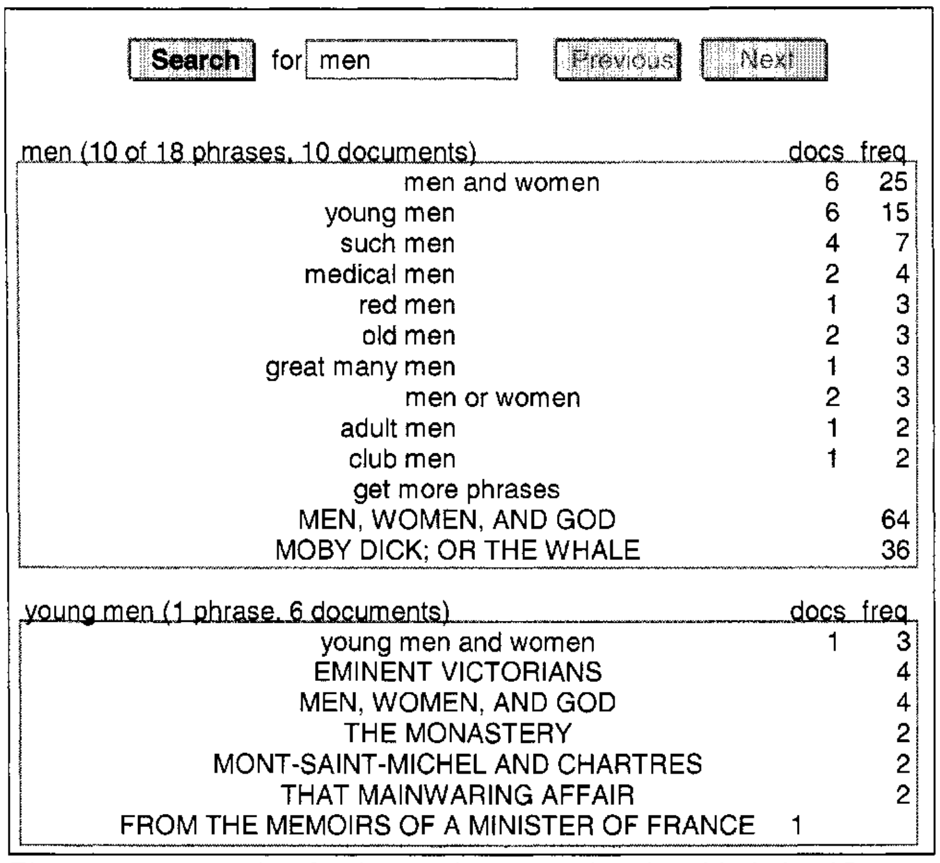
- Wages earned for women are 22% lower than for men
- The reason women are limited to certain areas and earned lower wages is because family responsibilities limit their ability to travel

3.3.3 Navigation with Nouns

Nouns help capture the content matter of a document. Enlisting noun metadata for searching and browsing is practical for retrieving documents containing the user's desired content. With this in mind, the `extract_nouns` option was developed to take advantage of the nouns the Summarizer extracts from documents (Chapter 2, p. 71). Figures 3.15 and 3.16 illustrate Greenstone's searching and browsing techniques (Chapter 2, p. 25) using nouns.

3.3.4 Navigating with Noun Phrases

According to Evans and Zhai (1996), "the ideal indexing terms would



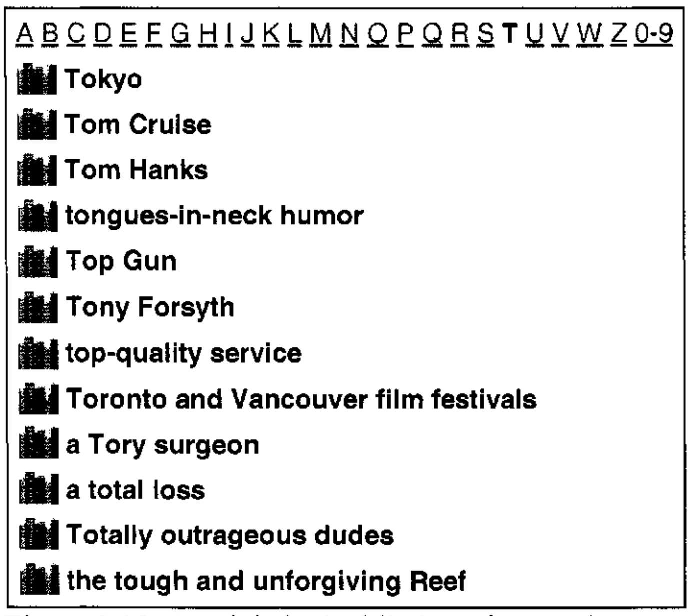
men (10 of 18 phrases, 10 documents)	docs	freq
men and women	6	25
young men	6	15
such men	4	7
medical men	2	4
red men	1	3
old men	2	3
great many men	1	3
men or women	2	3
adult men	1	2
club men	1	2
get more phrases		
MEN, WOMEN, AND GOD		64
MOBY DICK; OR THE WHALE		36

young men (1 phrase, 6 documents)	docs	freq
young men and women	1	3
EMINENT VICTORIANS		4
MEN, WOMEN, AND GOD		4
THE MONASTERY		2
MONT-SAINT-MICHEL AND CHARTRES		2
THAT MAINWARING AFFAIR		2
FROM THE MEMOIRS OF A MINISTER OF FRANCE	1	

Figure 3.17: Browsing for the word "men" using the Phind classifier.

directly represent the concepts in a document" (p. 18); and using phrase-based indexing is "a step toward the ideal of concept-based indexing" (p. 18). They explain that using phrases improves precision (Evans & Zhai, 1996, pp 18-19):

1. Phrases can replace individual indexing words. For example, if both "dog" and "hot" are used for indexing, they will match any query in which both words occur. But if only the phrase "hot dog" is used as an index term, then it will only match the same phrase, not any of the individual words.
2. Phrases can supplement word-level matches. For example, if only the individual words "junior" and "college" are used for indexing, both "junior college" and "college junior" will match a query with the phrase "junior college" equally well. But if we also use the phrase "junior college" for indexing, then "junior college" will match better than "college junior", even though the latter also will receive some credit as a match at the word level.


A B C D E F G H I J K L M N O P Q R S T U V W Z 0-9

Tokyo
Tom Cruise
Tom Hanks
tongues-in-neck humor
Top Gun
Tony Forsyth
top-quality service
Toronto and Vancouver film festivals
a Tory surgeon
a total loss
Totally outrageous dudes
the tough and unforgiving Reef

Figure 3.18: An alphabetical listing of noun phrases.

Greenstone's Phind (Chapter 2, p. 25) classifier facilitates phrase browsing. A query for the word "men" in Figure 3.17 populates the upper panel, which consists of three columns. The first column lists phrases containing the term "men". The second displays the number of documents containing the corresponding phrase. The third indicates the frequency the phrase appears in the collection. For example, the first phrase "men and women" appears in six documents 25 times. Clicking on the phrase "young men" in the upper panel populates the lower panel with additional phrases containing "young men", and the titles of the documents containing "young men". In addition to Phind, users can exercise the searching and browsing techniques described above in the noun metadata section. For example, a listing of noun phrases is shown in Figure 3.18. Clicking on a noun phrase retrieves documents pertaining to the phrase.

3.3.5 Setting the Summary Length

The 'ideal' summary length is subjective; what may be best for one person or situation may not be for another. Jing et al. (1998) found, when comparing evaluation results of the same summarization system, that the results can be significantly different if their summary lengths were different. Accuracy for "ideal" summary based evaluation decreases as summary lengths increases. For task-based evaluations, it is more beneficial for systems to set their own summary lengths (Jing et al., 1998, p. 68). Examples of situations where different summary/abstract lengths are employed include:

- Bainbridge, McKay, and Witten's (2003b) abstract for the conference paper, *Assembling and Enriching Digital Library Collections* is 989 bytes long
- Quigley's (2005) *5 Minute Herald*[7] for *Watergate tipster 'Deep Throat' revealed* is 242 bytes
- Magill's (Magill, 1976) Masterplot[8] for *Moby Dick* is 11,498 bytes
- McGowan's (McGowan, 2002) abstract for the thesis *Efficient Phrase Hierarchy Inference* is 1800 bytes long

Currently, the lengths of summaries are limited to a set number of bytes. The default length for `TextSummary` is 500 bytes. This length was chosen because the Document Understanding Conference (DUC) task guidelines state short summaries are to be ~100 words in length (*DUC 2003*, 2003). As explained

[7] *5 Minute Herald* is a selection of daily briefings of the day's top stories, business, editorials, sports, entertainment, and other news sections seen in *The Calgary Herald*.
[8] *Magill' Masterplots* is an encyclopedia set containing plot summaries and short essays on famous works of world literature (*MORE*, 2003).

in Chapter 2, Kolla entered the Summarizer into DUC 2003 and DUC 2004 and designed it to create summaries 500 bytes in length. This default length can be changed by specifying `-summary_length <int>` in the collections *collect.cfg* file. For example:

```
  plugin HTMLPlug -extract_summary -summary_length
  1000
```

sets *summary.pm* to extract `TextSummary` metadata `1000` bytes long from each document in the collection.

Although `summary_length` currently works with bytes, it can easily be altered to accept compression ratios (e.g. *5%, 10%, or 25%*). Lin and Hovy (2002, p.47) define the compression ratio as "the length of a summary (by words or sentences) divided by the length of its original document". Varying compression ratios and summary lengths affect summary performance. Lin and Hovy (2003, p. 79) found that high compression ratios resulted in lower system performance.

3.3.6 How to Deal with Long Documents

Situations may arise in which the user may not want to create a summary based on the entire document. For instance time: it is much more time consuming to produce summaries from extremely large documents. For example, the Project Gutenberg[9] eBook of George Eliot's (1860) *The Mill on the Floss* is 1,134,808 bytes (1108 KB or 1 MB) in length. It takes 2498 seconds (42 minutes) to extract a summary. To create summaries from a collection containing 10,000 similar documents would take 10 months. Contrarily, the Project Gutenberg eBook of

[9] Project Gutenberg was founded by Michael Hart in 1971. It "is the first and largest single collection of free electronic books, or eBooks" (*About Project Gutenberg*, 2004).

William Makepeace Thackeray's (1854) *A Little Dinner at Timmins's* is 59125 bytes (57 KB or 0.06 MB) in length and takes 55 seconds (1 minute) to extract a summary. A collection containing 10,000 similar documents would take 6 days to create summaries. Albeit, the above documents are not ideal for generating summaries, they are being cited for comparison reasons only. Chapter 4 uses large documents as a benchmark to compare document sizes and the time it takes to produce summaries from them. Specifying

```
plugin HTMLPlug -extract_summary -summary_by_size
60000
```

sets *summary.pm* to produce summaries based on the first `60000` bytes of the input file.

3.3.7 Dealing with Genre

One of the intentions of incorporating the Summarizer into Greenstone was to facilitate using the Summarizer with a variety of genres that could be seen in a typical DL collection. As explained in Chapter 2, the Summarizer was originally designed to work with newspaper articles as its input. The first sentence of a newspaper article provides a good short summary of the document (Jing et al., 1998, p.63). The Summarizer reflects this as it was designed to extract the first sentence of each document for use as the first sentence of the summary. Although this tactic works well for news articles, it does not necessarily work for all genres. For example, the first sentence of the summary in Figure 3.19 (*Crystal Spring Colony*, n.d.) is a reference and can safely be eliminated from the summary without altering its meaning. The *summary.pm* module was adjusted to

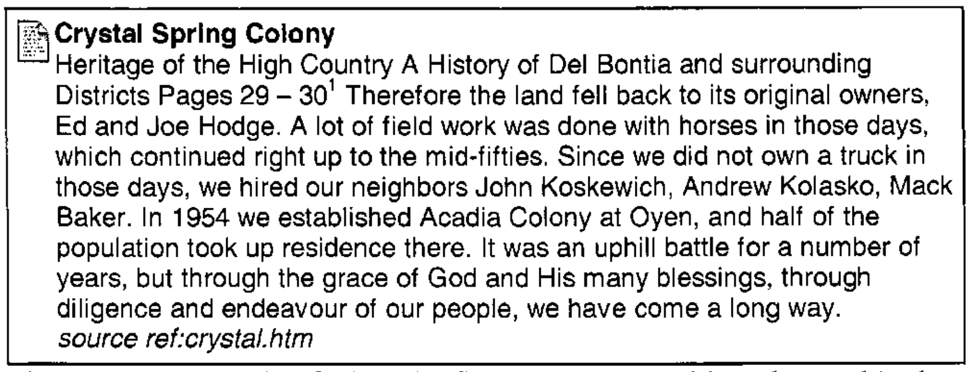

Crystal Spring Colony
Heritage of the High Country A History of Del Bontia and surrounding Districts Pages 29 – 30[1] Therefore the land fell back to its original owners, Ed and Joe Hodge. A lot of field work was done with horses in those days, which continued right up to the mid-fifties. Since we did not own a truck in those days, we hired our neighbors John Koskewich, Andrew Kolasko, Mack Baker. In 1954 we established Acadia Colony at Oyen, and half of the population took up residence there. It was an uphill battle for a number of years, but through the grace of God and His many blessings, through diligence and endeavour of our people, we have come a long way.
source ref:crystal.htm

Figure 3.19: Example of when the first sentence should not be used in the summary.

automatically use the top-ranking sentence from the chronologically ordered top-ranking segment as the summary's first sentence (Chapter 2, p. 39). In cases where users prefer the initial version, declaring `-summary_first_sentence` in the collection's *collect.cfg* file extracts the document's first sentence for use as the first sentence of the summary.

3.3.8 What to do When Unicode Characters Appear

Some documents contain special characters in the form of Unicode. There are numerous cases where these are encountered. For example, non-English documents regularly contain Unicode characters to represent accent marks or non-English characters, and technical papers present mathematical formulas as Unicode. Normally Unicode serves to enhance the document's readability; unfortunately, issues can arise when the Summarizer encounters it. The first issue was expounded upon earlier: the Tokenizer dies when it finds strings with three or more combined Unicode metasymbols. Fortunately, this issue was resolved (Section 3.2, p.63). The second issue concerns nouns: the Summarizer can incorrectly tag words containing Unicode characters as nouns. This poses a problem when it comes to selecting optimal sentences for use in the summary. For

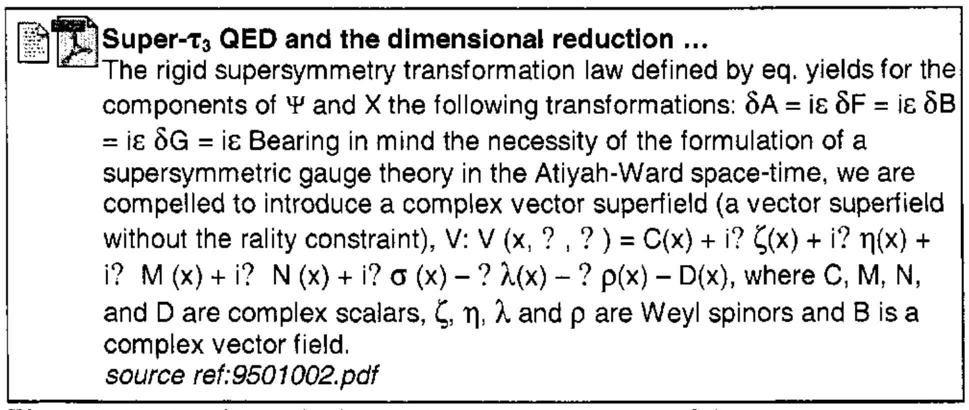

Super-τ_3 QED and the dimensional reduction ...
The rigid supersymmetry transformation law defined by eq. yields for the components of Ψ and X the following transformations: δA = iε δF = iε δB = iε δG = iε Bearing in mind the necessity of the formulation of a supersymmetric gauge theory in the Atiyah-Ward space-time, we are compelled to introduce a complex vector superfield (a vector superfield without the rality constraint), V: V (x, ? , ?) = C(x) + i? ζ(x) + i? η(x) + i? M (x) + i? N (x) + i? σ (x) – ? λ(x) – ? ρ(x) – D(x), where C, M, N, and D are complex scalars, ζ, η, λ and ρ are Weyl spinors and B is a complex vector field.
source ref:9501002.pdf

Figure 3.20: Mathematical equations appear as part of the summary.

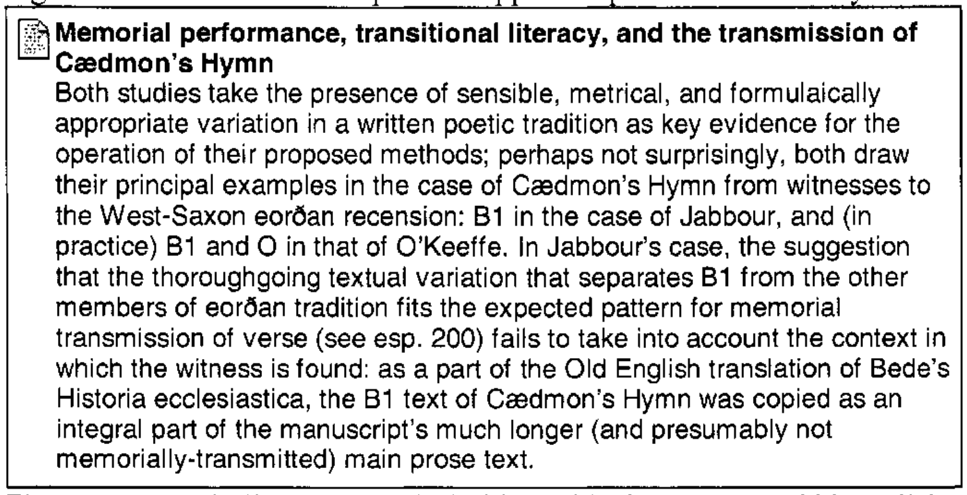

Memorial performance, transitional literacy, and the transmission of Cædmon's Hymn
Both studies take the presence of sensible, metrical, and formulaically appropriate variation in a written poetic tradition as key evidence for the operation of their proposed methods; perhaps not surprisingly, both draw their principal examples in the case of Cædmon's Hymn from witnesses to the West-Saxon eorðan recension: B1 in the case of Jabbour, and (in practice) B1 and O in that of O'Keeffe. In Jabbour's case, the suggestion that the thoroughgoing textual variation that separates B1 from the other members of eorðan tradition fits the expected pattern for memorial transmission of verse (see esp. 200) fails to take into account the context in which the witness is found: as a part of the Old English translation of Bede's Historia ecclesiastica, the B1 text of Cædmon's Hymn was copied as an integral part of the manuscript's much longer (and presumably not memorially-transmitted) main prose text.

Figure 3.21: Latin ligatures are desirable in this document on Old English text.

example, the summary in Figure 3.20 was generated from a physics paper that utilizes Unicode to represent mathematical formulas. Here, the Summarizer deemed mathematical equations salient and included them in the summary. By default, the Summarizer automatically includes Unicode. In situations where Unicode is not desirable, specifying the `summary_no_unicode` causes any sentences with Unicode to be excluded in the summary.

There are instances where users prefer to include Unicode in the summary. The summary in Figure 3.21 was generated from Cædmon's Hymn (O'Donnell et al., 2005). The book employs Unicode throughout to represent Old English characters. In this case, Unicode is desirable in the summary; not allowing Unicode would result in severe information loss in regards to the original meaning of the document. A brief discussion on information loss will be seen in Chapter 4.

3.4 Summary

This chapter detailed how the Summarizer was implemented in Greenstone. It began with the first section explaining how Greenstone can be extended through plugins, explained how plugins work, defined the different types of plugins, and informed that all plugins are derived from the BasPlug plugin. Next, the BasPlug was described as a specialized plugin that performs numerous universally required operations for each document. It went on to detail how the Summarizer was written as an option to the BasPlug to allow it to be inherited by all of Greenstone's plugins.

The second section told how to install the Summarizer as a package. The Summarizer utilizes tools to assist in generating summaries: WordNet-2.0, WordNet-QueryData-1.3.1, OAK, C99, and segsplitter. Code that is used by Greenstone but developed by other resources is stored in Greenstone's packages directory. The Summarizer's tools are bundled and stored in this directory.

The third section examined the issues that arose when the Summarizer was installed into Greenstone, requiring its code to be adjusted. The first issue explained how the Summarizer worked only on plain text files and needed to be adapted to include HTML, PDF and text documents. Second, input text needed to be cleaned and stripped of text not deemed appropriate for use in a summary, such as headings, lists, page numbers, figure data, etc. Third, the Summarizer needed to be robust and handle situations causing it to halt. Tokenizer failures were observed with files containing inoperative data, and extremely large files caused memory errors in the segmenter. Uninitialized variables and mishandled entities instigated WordNet errors. Errors caused by harmful data were solved by removing those sentences containing the offenders. The memory errors witnessed by the segmenter were solved by truncating copious files to a manageable size. Extensive cleaning of the Summarizer's code to initialize variables and fix entity issues solved the WordNet problems.

The final section elaborated on the specific options that were created for the Summarizer. This section described these options, explained how to use them, and gave examples to justify why/where they would be used. Along with summary metadata, noun and noun phrase metadata can be extracted for searching and browsing. Extracted metadata is available for both document and section-level indexing. Summaries can be customized to take advantage of certain genres. For example, using the first sentence of the document is most desirable with the newspaper genre. Additionally, summaries containing Unicode may be suitable for genres containing extensive mathematical formulas, while it may be suitable for other genres. Summaries can be tailored to a specific length. Summaries can also be created based on a portion of the text instead of the entire document. The following chapter analyzes how the Summarizer works in terms of time and information loss.

Chapter 4

Technical Evaluation

The purpose of this chapter is to analyze the altered version of The Summarizer, as it is described in Chapter 3, as a part of Greenstone. The focus is to see if it is feasible to produce summaries for digital collections. Two aspects of feasibility include time and information loss. In terms of time: how long does it take to run the Summarizer on digital collections? The Summarizer functions during Greenstone's Import Phase. For the purpose of this thesis, we are only concerned with the amount of time it takes to process documents during Greenstone's Import Phase and not its Build Phase (Chapter 2, p. 23). We will look at what factors play a role with regards to time, and make recommendations on how to resolve possible time issues. The last section discusses information loss seen in non-English documents and those containing Unicode.

4.1 Description of the Test Collections Used

Three digital collections were created for testing the time it takes to process documents using the Summarizer during Greenstone's Import Phase. As described in Chapters 2 and 3, the Summarizer was designed for use on newspaper documents. With this in mind, the first collection was created to include 10,000 newspaper documents[10] in text form. Researchers commonly use DLs for finding papers applicable to their work; the second collection includes 10,000 physics papers[11] in PDF form. Numerous DLs are dedicated to preserving books in digital format, known as eBooks. The third collection includes 2,740 eBooks from the Project Gutenberg Library.[12] The Gutenberg collection serves two purposes: first to see how the Summarizer handles extremely large documents; second to physically look at the summaries produced from eBooks. The latter will be examined in Chapter 5. Originally, it was intended to compare three different collections containing 10,000 documents, but as we will see, the Gutenberg collection is large and time consuming. For time purposes, the Gutenberg collection was limited to 2,740 documents.

Each collection uses identical Summarizer parameters: summary size is set to 500 bytes; the summary is based on the entire document; and the metadata extracted for indexing include summary, noun phrases, and nouns. As a reminder, this computer was not dedicated solely for our test purposes and other users had access to it.

4.1.1 Time Tests to Process Documents

Table 4.1 provides size and time data results for each test collection. Size is broken down into three parts: the number of input documents in each collection,

---

[10] Newspaper documents were obtained from the Text REtrieval Conference (TREC). The Summarizer was originally evaluated with newspaper documents from TREC.

[11] PDF documents were obtained from ArXiv (www.arxiv.org), an e-print service operated by Cornell University. Physics, mathematics, non-linear science, computer science, and quantitative biology documents are available.

[12] The Project Gutenberg eBook Library (www.gutenberg.org) provides free electronic versions of books that fall under the public domain and whose authors gave permission for Project Gutenberg to distribute them.

Table 4.1: Test collection dataset. Total Document Size refers to the original size of the input documents, and Total Text Size refers to the size of the documents after text has been extracted from them.

Actual Dataset						
Collection	Number of Documents	Total Document Size (MB)	Total Text Size (MB)	Total Number of Segments	Total Import Time (hours)	Total Import Time (days)
Newspaper	10,000	52	52	41,352	38	1.6
PDF	10,000	810	278	115,225	150	6.3
Gutenberg	2,740	904	787	102,845	883	36.8

the total size of those input documents in megabytes (MB), the total size of text extracted from the documents,[13] and the total number of segments produced by the Summarizer. Time refers to how long it takes in hours and days for documents to be processed during Greenstone's Import Phase.

In Table 4.1, each test collection has different file sizes: the Newspaper collection is smallest with 52 MB, the PDF collection has 1,810 MB, and Gutenberg has 904 MB. Although the first two collections contain 10,000 documents, the Gutenberg collection only has 2,740. It is smaller because it is more time consuming to process compared to the other two. For instance, it takes 883 hours for 2,740 Gutenberg documents to pass through the Import Phase, while 10,000 newspaper documents take only 38 hours. Based on the dataset, with four times fewer documents in Gutenberg than the other two collections, we see that Gutenberg takes six times longer than PDF documents, and 23 times longer

[13] Recall from Chapters 2 and 3, the Summarizer works with text as its input. Thus, the size of the text extracted from the input documents could be different from the size of the actual documents themselves.

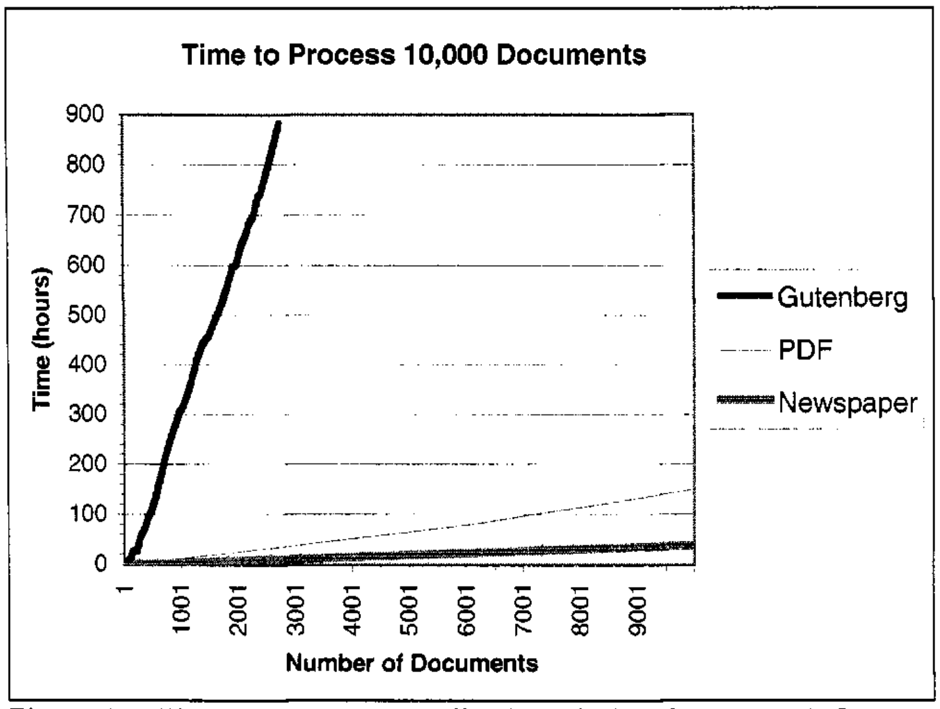


Figure 4.1: Time to process test collections during Greenstone's Import Phase

than Newspaper. The time difference is more noticeable when Excel's trend[14] function is applied to the Gutenberg collection's data results. A collection of 10,000 Gutenberg eBooks would take 3,230 hours to process, or 135 days.

Figure 4.1 compares the time it takes, in hours, for documents in each collection to be processed. Interestingly, documents in the Newspaper collection appear to be processed quicker than PDF or Gutenberg. This is deceiving because speed is influenced by the collection's file size, not its document count. Based on the size of the extracted text, it is revealed that the PDF collection is actually faster. The Newspaper collection processes 1.4 MB per hour, the PDF collection processes 1.9 MB per hour, and the Gutenberg collection is the slowest at 0.9 MB

[14] Based on known x's and known y's, Excel's Trend function calculates a linear trend fitting a straight line. The data in Figure 4.1 is linear, which allows the Trend function to be applied to calculate the time to process 10,000 Gutenberg eBooks.

Table 4.2: Processing times per hour

Collection	Total Processing Time (hr)	Summarizer Processing Time (hr)	Greenstone Processing Time (hr)
Newspaper	41	31	10
PDF	152	102	50
Gutenberg	884	814	70

per hour. Recall from Table 4.1, Gutenberg's collection contains 787 MB of input text and takes 37 days to process. Applying Excel's Trend function to the Newspaper and PDF collections (based on 787 MB) reveals the Newspaper collection would take 21 days to process and the PDF 17 days. This section explores what factors play a role in time, on both the Summarizer and the Greenstone side.

4.1.2 Factors Contributing to the Summarizer's Processing Time

Initially, each document is stripped of all mark-up tags, leaving only the actual text to be processed by the Summarizer. Table 4.2 shows the time to process plain text for each test collection. Three time measurements are shown: Total Processing, Summarizer Processing, and Greenstone Processing. The Total Processing time measures the entire time to process the plain input text during Greenstone's Import Phase, the Summarizer time measures the actual amount of time the Summarizer spends on each MB of text, and the Greenstone time is calculated as the difference between the other two time measurements. As shown, the Summarizer dominates the total processing time by about 67%-92%. Greenstone's processing time will be examined later, but first, we will look at the Summarizer's time.

Table 4.3: Processing speeds MB per hour

Collection	Total Processing Speed (MB/hr)	Summarizer Processing Speed (MB/hr)	Greenstone Processing Speed (MB/hr)
Newspaper	1.3	1.7	5.2
PDF	1.8	2.7	5.5
Gutenberg	0.9	1.0	11.2

Table 4.4: Percentage of time spent on each Summarizer stage

Summarizer Processing Times (percentage)	Newspaper	PDF	Gutenberg
Text Cleaning	0.2%	2.2.%	0.2%
Text Tokenization and Segmenting	12.2%	6.5%	1.7%
Text Chunking	62.4%	55.6%	7.5%
Noun Extraction	0.7%	1.2%	0.2%
Lexical Chaining	24.5%	34.5%	90.3%
Sentence Extraction	0.1%	0.1%	0.0%

Processing speeds in Table 4.3 are calculated based on the processing time (Table 4,2) divided by the total text size (Table 4.1). One hypothesis is that with time based on the number of MBs processed, each test collection's speed would be identical; as Table 4.3 reveals this is not the case. Contributing factors include time spent on cleaning text, tokenization failure, file truncation, and segmentation failure. Additional factors include the quantity of nouns in each collection, input/output time to write data to the hard-drive, and the fact a dedicated machine was not used for testing purposes. To assist in understanding how these factors affect time, Table 4.4 examines the percentage of time the Summarizer spends on each stage.

Initially, text requires cleaning before beginning the Summarization process. This involves stripping any unnecessary text such as mark-up tags that may be included. While stripping tags from HTML text is straightforward, tags added to documents during their conversion from PDF to HTML are more complicated (Chapter 3, p. 57). This leads to extra cleaning time for PDF files compared to HTML, and less for text files. Table 4.4 mirrors this: for the PDF collection (consisting of PDF files) 2% of the total time is spent cleaning input text while for the other two collections (consisting of plain text files) 0.2% of the time is spent cleaning input text. In addition to removing mark-up tags, text needs to be clean of undesirable text such as headings, lists, footnotes, diagram/figure data, etc. The Newspaper collection contains a lower amount of undesirable text than the other two collections, leading to a smaller percentage of time spent cleaning. The Gutenberg collection consists of eBooks, which contain more headings than the Newspaper collection (e.g. chapter titles). In addition to headings, the PDF collection contains a vast amount of diagram/figure data that require cleaning.

As we discovered in Chapter 3, the Summarizer encounters problems with some input text. In the first stage, text undergoes tokenization. A file containing extremely large strings and tokens causes the Tokenizer to die, requiring removal of the offending sentences, followed by tokenization again. Fixing problem text and re-processing adds time. Segmenter failure causes similar results. Extremely large files result in the segmenter running out of memory. To allow summarization to proceed, the offending file must be truncated and resubmitted to the segmenter. Occasionally, files are so large that truncating the file once is not enough. Waiting for the segmenter to finish, having it run out of memory, truncating the file, and restarting increases time.

Table 4.4 reveals that the majority of time spent in all three collections

Table 4.5: Greenstone's processing speed

Collection	With Summary (MB/hr)	Without Summary (MB/hr)
Newspaper	5.1	35.8
PDF	5.4	7.6
Gutenberg	2.7	130.1

occurs during the Text Chunking and Lexical Chaining stages combined. As described in Chapter 2, each document's segment undergoes text chunking by Oak's Chunker tool. Afterwards, nouns are extracted and written to a corresponding file for each segment. During the Lexical Chaining stage, each noun file is read into memory; at which time WordNet is used to create lexical chains from each noun (Chapter 2, p. 34). Generated chains are then written to files for each document. Large documents and those containing large quantities of nouns utilize more time.

As shown, the Noun Extraction and Sentence Extraction stages take the least amount of time. The reason for this is that these two stages are linear and do not require much time.

4.1.3 Factors Contributing to Greenstone's Processing Times

Each test collection was run through Greenstone independently of the Summarizer in order to examine the effects Greenstone's processing time has on the text MB. Table 4.5 compares Greenstone's processing speeds when applied to collections utilizing the Summarizer and those not using it. Two speeds for each collection are shown: "With Summary" and "Without Summary". The prior is based on collections utilizing the Summarizer where the speed is calculated using the time difference between the Import and Summarizer processing times seen in

Table 4.6: Greenstone time and size differences between collections created with additional Summarizer metadata and without.

Summarizer Processing Time: Import with Summary vs Import without Summary Metadata (hrs)	Newspaper	PDF	Gutenberg
With Summary Metadata	10	50	70
Without Summary Metadata	1	36	6
Percent More Time Added by Metadata	15%	72%	9%
Archive Text Size (MB)	Newspaper	PDF	Gutenberg
With Summary	247	4655	1574
No Summary	62	4089	917
Percent More Text Added by Metadata	25%	88%	58%

Table 4.2, while the later is the speed for Summarizer-free collections.

Greenstone's processing times for collections using the Summarizer took longer than those independent of the Summarizer. The major factor contributing to this time difference is the amount of additional metadata generated by the Summarizer requires additional processing to be performed by Greenstone. For example, the Summarizer adds 88% more text to the PDF collection's archive files (Table 4.6). This results in 72% more time for Greenstone to process the PDF collection when the additional metadata is added. As mentioned, the test machine was not dedicated for our test purposes, thereby resulting in unknown factors contributing to the time difference.

4.2 Using the summary_by_size Option to Increase Speed

The `summary_by_size` option was written for the Summarizer to generate summaries based on the first `n` bytes of input text (Chapter 3, p. 75). The theory behind this option is that the less input text the Summarizer has to process, the faster it will operate. Each test collection was generated using the

Table 4.7: Processing speeds for collections using the entire input text and those using the first 5000 bytes of input text

Summarizer Processing Speed: Full Text vs First 5000 bytes*	Newspaper	PDF	Gutenberg
Full Text (MB/hr)	1.3	1.6	1.9
First 5000 bytes (MB/hr)	1.6	1.7	9.0
Speed Increase (percentage)	23%	6%	374%
*Based on 52 MB input text			

`summary_by_size` option based on the first 5000 bytes of input text and compared against those created using the full input text. Table 4.7 reveals using the option increased each collection's speed: the PDF collection increased by 6%, followed by the Newspaper collection at 23%. The most increase was noticed in the Gutenberg collection, where using the `summary_by_size` option increased processing time from 2 MB per hour to 9 MB per hour, 374% faster.

4.3 A Discussion on Information Loss

Chapter 3 described three situations where data is deleted from the original input. The first occurs when the tokens are too large for the Tokenizer to handle. These tokens need to be eliminated in order for the Summarizer to continue. The second arises when documents are too large and cause the Segmenter to run out of memory, requiring documents to be truncated. The third materializes when users do not want Unicode characters to appear in the summary, resulting in sentences containing Unicode being removed from the summary. Information from the original document is lost due to these factors. Table 4.8 shows information loss as it pertains to the datasets in Table 4.1. Although the Newspaper collection does not suffer from information loss due to

Table 4.8: Information loss

Information Loss due to Tokenization and Unicode	Newspaper	PDF	Gutenberg
Original Text Size (MB)	52	270	787
Text lost due to Tokenization (MB)	5	25	384
Text lost due to Unicode* (MB)	0	30	11
Loss due to Tokenization (percentage)	9%	9%	49%
Loss due to Unicode (percentage)	0%	11%	1%
*Entire sentences removed if they contained Unicode			

Unicode, loss due to large tokens and truncated documents is 9%. Information loss due to Unicode is seen most frequently in the PDF collection, while loss due to large tokens and truncated documents appears in the Gutenberg collection. The PDF collection contains a large number of Unicode characters, resulting in deletion of 11% of the collection. Gutenberg contains full eBooks, which are extremely large; 49% of the Gutenberg collection is lost due to large tokens and truncated documents.

As discussed earlier, Unicode is not dismissed in every instance, only in those collections for which users deem necessary (e.g.: documents containing large amounts of mathematical formulas). Information loss due to undesirable Unicode is not necessarily bad.

Information loss due to large tokens and truncated documents only occurs in order for the Summarizer to succeed. The Tokenizer dies from tokens that are too large and the Segmenter runs out of memory on immense documents. The question arises: is it better to produce a summary from a document with deleted data, or not to produce a summary at all?

Table 4.9: Time expensive Summarizer stages

Collection	Topmost Summarizer Stage	Time Spent	Secondmost Summarizer Stage	Time Spent
Newspaper	Text Chunking	62.4%	Lexical Chaining	24.5%
PDF	Text Chunking	55.6%	Lexical Chaining	34.5%
Gutenberg	Lexical Chaining	90.3%	Text Chunking	7.5%

4.4 Summary

The focus of this chapter was to determine if it is feasible to produce summaries for digital collections. Three collections were tested: Newspaper, PDF, and Gutenberg. The Newspaper collection consists of small newspaper articles in plain text, PDF contains physics papers in PDF form, and Gutenberg includes large eBooks in plain text. Tests showed the PDF collection was processed the fastest at 1.8 MB/hr, followed by the Newspaper collection at 1.3 MB/hr, with the Gutenberg collection being the slowest at 0.9 MB/hr. These tests were split to reveal the processing speeds executed by both Greenstone and the Summarizer. Greenstone processed the Gutenberg collection faster at 11.2 MB/hr, while the Summarizer processed the Gutenberg collection at 1.0 MB/hr. Thus, the Summarizer works slowest on collections containing large files.

The Summarizer undergoes seven stages. The majority of time was consumed during the text chunking and lexical chaining stages (Table 4.9). The ultimate solution to increasing speed would be to optimize these stages. Since this thesis is focused on optimizing the Summarizer without altering its code, a secondary solution was implemented to process a portion of the document. Operating on the first 5000 bytes of input text resulted in speed increases in each collection. The Gutenberg collection witnessed dramatic improvement from 1.9

MB/hr to 9.0 MB/hr.

Three situations cause information loss. The first is when data causes tokenizer errors resulting in the offending strings being removed. The second occurs when the segmenter does not have enough memory to execute large files, thereby requiring the files to be truncated to a manageable size. The third is seen when the user specifies those sentences containing Unicode characters be omitted from the summary. With the Newspaper collection there was little information loss with only 9% due to tokenization. With the PDF collection there was a similar level of information loss to tokenization, 11% caused by deleting Unicode characters. With the Gutenberg collection 49% of information was lost due to tokenization.

This chapter has determined it is feasible to produce summaries for digital collections containing newspaper articles and PDF documents. Processing speeds for these collections are adequate and information loss is low. Digital collections containing large files, such as those seen in the Gutenberg collection, are not feasible. Processing speeds are too slow and significant information loss appears due to tokenization. The next chapter examines how the Summarizer works qualitatively with regards to genre.

Chapter 5

Genre Analysis

This chapter is aimed at discovering how the Summarizer works qualitatively with regards to genre. Our hypothesis is that the Summarizer works better on short, narrative style genres, such as narrative newspaper articles. Summaries were produced on four different genres: newspaper, M.Sc. theses, novels, and poetry. Documents were selected on the basis of the availability of professionally written summaries or abstracts. This chapter is split into three parts. The first describes the genres tested and the professional method of abstraction used. The second section compares each summary against the document's corresponding abstract. The third section discusses when the Summarizer should or should not be used.

5.1 Description of the Genre Documents Used

The four genres that were selected to test the Summarizer include newspapers, M.Sc. theses, novels, and poetry. Although poetry is not normally a genre one would summarize, it is included in this section because it is interesting to see what happens when the Summarizer generates a summary of it.

The Summarizer was designed to work with newspaper articles; it seems appropriate to test the Summarizer with them. Not all newspaper articles are the same; there exist sub-genres within the newspaper genre. For example, an article can be written in a narrative-style, while a different one could be written in a roundup-style.[15] The Summarizer may not work equally well with all newspaper sub-genres. It has become common practice for newspapers to provide a summary of few of its articles in a section of its paper. The *Calgary Herald* is among these; their section is referred to as the *5 Minute Herald.* Two articles were selected from the *Calgary Herald*: *'I'm the guy they used to call Deep Throat': World's most famous anonymous source revealed* (Poole, 2005) and *Wild weather spawns tornadoes* (Gauntlett, 2005). Summaries generated by the Summarizer are compared against their corresponding *5 Minute Herald* summaries.

Many universities find it beneficial to possess a DL comprised of their students' theses. It would be a practical feature to store, summarize, and retrieve them. For that reason, the M.Sc. thesis genre has been included for analysis. M.Sc. theses include abstracts written by the author. Scientific theses commonly include mathematical equations, which are represented by Unicode. Chapters 3 and 4 discuss how the Summarizer tags Unicode characters as nouns and scores the sentences accordingly. In an effort to minimize the number of equations appearing in the finished summary, the Summarizer was set with the `summary_no_unicode` option. Generated summaries from two M.Sc. theses, *Efficient Phrase Hierarchy Inference* (McGowan, 2002) and *Automatic Text Summarization Using Lexical Chains: Algorithms and Experiments* (Kolla, 2004) are compared against their corresponding abstracts.

[15] Stewart defines a roundup as "a story that rounds up a variety of developments in a broad, ongoing story" (Stewart, 2003, p. 235).

Many scholars have dedicated their work to analyzing books and writing plot summaries about them. Frank Magill is an expert in this field. He has written volumes of books entitled Masterplots that contain plot summaries and short essays on famous works of world literature. His masterplots for the novel *Moby Dick* and artistic non-fiction work, *Eminent Victorians* are used to compare the adjacent summaries produced by the Summarizer.

5.2 Comparing Summaries against Professional Abstracts

Two commonly used methods of producing summaries include abstraction and extraction. Recall from Chapter 2 that the abstraction method creates summaries by simplifying and condensing the original text. This method requires abstracts to be generated manually. On the other hand, the extraction method can be performed automatically. This procedure extracts sentences from the original text to include in the summary. The Summarizer utilizes the latter method. The professional abstracts described above use both methods. Abstraction is seen in summaries of M.Sc. theses, novels, and poetry, while extraction is used in newspapers. With this in mind, analysis is performed by targeting key topics[16] within the professional abstract and checking if they exist in the Summarizer version. The criterion used for comparing the automated summaries against the professional abstracts was developed in conjunction with Dr. Dan O'Donnell. Dr. O'Donnell is the chair for the University of Lethbridge English Department and participated as an external valuator. For analysis, the summaries generated by the

[16] The Summarizer creates lexical chains from nouns, and uses them to determine which salient sentences to extract for use in the summary. Therefore, it was logical to manually extract key topics based on nouns and noun phrases.

5 Min Herald Title: Watergate tipster 'Deep Throat' revealed

The Washington Post said Tuesday that a former FBI official, Mark Felt, was the confidential source known as "Deep Throat" who provided the newspaper information that let to then-president Richard Nixon's impeachment and eventual resignation.

Automatically Generated Summary:

The mystery surrounding one of America's most durable journalistic secrets was solved Tuesday with the revelation that a retired FBI official was the anonymous Watergate-era source known as "Deep Throat." But is was Felt, at the time the FBI's second in command, who had been the highly placed government contact who helped guide Woodward and Bernstein's path breaking coverage of the FBI investigation into the June 1972 break-in and bugging of Democratic Party offices in Washington's Watergate complex.

Figure 5.1: The top summary is taken from the *5 Minute Herald* (Quigley, 2005, June 1); the bottom summary is an automatically generated version created by the Summarizer as part of Greenstone

Summarizer were set to the same length as their corresponding, professional abstracts.

5.2.1 Analyzing the Newspaper Genre

On Wednesday, June 1, 2005, the *Calgary Herald* ran a story revealing the identity of the infamous "Deep Throat", (Poole, 2005, June1). Quigley (2005, June 1) compiled a summary for this story in the *Calgary Herald's 5 Minute Herald* section. A summary of a similar length was automatically generated by the Summarizer for the same newspaper article. Both summaries are shown in Figure 5.1. In order to measure how the Summarizer compares to the professional summary produced by Quigley, key words and topics are highlighted in the professional version and compared against the automated version.

In the figure, the automated version successfully identifies "Deep Throat" as the former FBI official, Felt. Although the professional version explicitly says

5 Min Herald Title: Twisters and hailstorms pound flood-worn Alberta

Parts of flood-ravaged Alberta were lashed by tornadoes and hail stones as big as golf balls Tuesday night, only hours after insurers estimated flood losses would exceed million and residents believed the worst of their weather woes were over. In a new weather warning issued Tuesday evening, Environment Canada said golf ball-sized hailstones were reported west of Claresholm and at least one tornado was spotted near Lethbridge.

Automatically Generated Summary:

Parts of flood-ravaged Alberta were lashed by tornadoes and golf ball-sized hailstones Tuesday night, only hours after insurers estimated flood losses would exceed $200 million. "This is a warming that thunderstorms are imminent," said Environment Canada at 6:30 p.m. In Calgary, more than 150 flood victims sought assistance at the city's disaster relief centre, while two people were charged with failing to comply with outdoor water restrictions.

Figure 5.2: The top summary is taken from the *5 Minute Herald* (Quigley, 2005, June 22); the bottom summary is an automatically generated version created by the Summarizer as part of Greenstone

Felt "provided the newspaper information", the automated version implies it, "(he) helped guide Woodward and Bernstein's path breaking coverage". Unfortunately, the automatic version completely omits any references to President Nixon's impeachment and resignation. On another note, the automatically generated summary is coherent. Overall, the Summarizer's summary does fare well both compared to the professionally written summary, and in terms of readability.

In Spring 2005, the *Calgary Herald* covered a story on tornadoes hitting Alberta (Gauntlett, 2005, June 22). A corresponding summary was included in their *5 Minute Herald* section (Quigley, 2005, June 22). As in the previous example, the newspaper article was processed by the Summarizer as part of Greenstone. Both summaries are shown in Figure 5.2. It is interesting that the first sentence of each summary is almost identical; the difference being the professional version refers to human feelings and the automated version does not. The automated version states the estimated dollar amount of flood losses would exceed $200 million, whereas the professional version simply states the losses would exceed millions (without an actual dollar amount included). Although references to Environment Canada are incorporated in both summaries, the Summarizer's summary fails to inform readers that "hailstorms were reported west of Claresholm and at least one tornado was spotted near Lethbridge". Similar to the previous example, the Summarizer's version of the *Herald's* tornado summary reads easily and does cover the majority of issues seen in the professionally written summary.

5.2.2 Analyzing the M.Sc. Thesis Genre

In the first M.Sc. thesis, *Efficient Phrase Hierarchy Inference* (McGowan, 2002), McGowan improves upon Phind, Greenstone's phrase browser. As mentioned, McGowan's thesis uses the abstraction method to manually generate her abstract. When analyzing abstracts produced by this method, one has to keep in mind that the abstract is influenced by the author's understanding of the topic, and his/her own personal writing style. With this in mind, eighteen key terms and topics were extracted from the thesis abstract[17]. Figure 5.3 sorts them into two columns. The first includes terms and topics covered in the Summarizer's automatically generated summary; the second lists those not mentioned. The second column hosts the majority of the issues, most notably being the terms

[17] For each sentence of the thesis abstract, key terms and topics were manually extracted, interpreted, and compared against the Summarizer's version. Terms and topics in the Summarizer version that were similar to those in the thesis version were counted as a match.

Information Successfully Included:	Missing Information:
• expansion relation • hierarchy • sequence of words occur multiple times • suffix trees • improvement in phrase extraction time • suffix arrays • implementation time and memory for original suffix array	• hierarchical phrase browsing system • Phind • Greenstone • phrase hierarchy inference • linear-time construction algorithms • linked list tree representations • hash table representations • nonlinear suffix tree construction algorithm • alphabet dependence • specialized tree representations • subphrase checking algorithms

Figure 5.3: Comparison between McGowan's (2002) M.Sc. thesis abstract and the Summarizer's version.

"Phind", "phrase browsing", and "Greenstone". Since the thesis is based on improving Greenstone's phrase browsing function, called Phind, it is imperative that the summary includes information pertaining to it.

In addition to examining key topics, it is interesting to explore how the Summarizer's automatically generated summary stands in regards to readability. Figure 5.4 shows the summary generated by the Summarizer implemented as part of Greenstone. Chapters 2, 3, and 4 explain how the Summarizer uses nouns and lexical chains to determine which salient sentence to extract for use in generating summaries. Theses commonly use the nouns "Figure", "Table" and cardinal numbers. Accordingly, these nouns appear frequently in the summary, triggering an increased number of sentences in the summary containing the applicable nouns. For example, the summary in Figure 5.4 contains five occurrences of the word "Table" and four of the word "Figure". Constant referrals to tables and figures surmount to a reduction in the summary's readability.

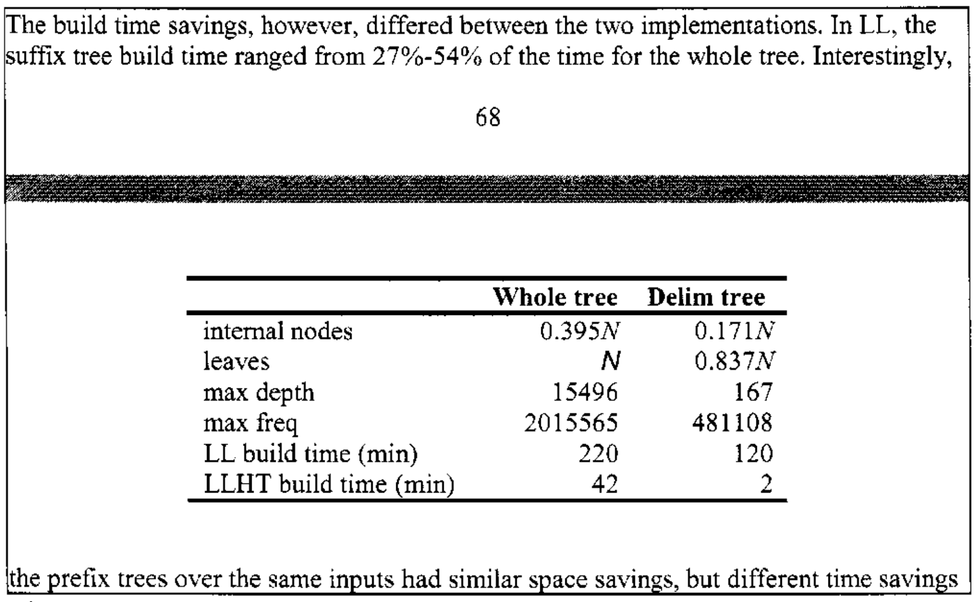
Automatically Generated Summary:

The expansion relation is the key to the hierarchy: The basic problem addressed is, given a word or sequence of words w, find all phrases in a given text corpus that contain w and 1. occur multiple times, 2. are maximal in the sense that they have no unique expansion in either direction, 3. are minimal in the sense that no shorter expansion of w satisfies these conditions. There are three cases when extending a suffix j, in phase i + 1. Interestingly, Table 3.4: Comparison of whole and delimited suffix trees: on FAO. internal nodes max depth max freq LL build time (min) LLHT build time (min) the prefix trees over the same inputs had similar space savings, but different time savings. Figure 4.1: Memory profile during tree construction: treesuffix and treesuffix Table 4.3: Space savings for 2HT over LLHT. suffixTree: TEdges prefixTree: TEdges prefixTree: TLeaf prefixTree: TSuffLink The static space requirement of treesuffix In treesuffix, reducing the size of TEdges and losing TLeaf and TSuffLink from the prefix tree together save 21.6N + 4Q bytes, as shown in Table 4.3. Omitting the prefix tree TLeaf array and frequency data, and compacting the suffix tree TLeaf array together Figure 4.9: Improvement in tree construction time: treesuffix Figure 4.10: Improvement in phrase extraction time: treesuffix Figure 4.11: Improvement in overall run time: treesuffix Table 4.6: Space savings for A2T over LL. HT would Table 6.1: Memory requirement for suffix arrays and suffix trees. input symbols suffix array have about the same build time, but any tree traversals or iterations through all children of a node would be much slower. The original suffix program used suffix arrays to extract the hierarchy, and required 16 hours and 179 MB of memory (with a peak of 191 MB during array construction) to process a moderate-sized collection of 12 million symbols.

Figure 5.4: Automatically generated summary for the thesis entitled *Efficient Phrase Hierarchy Inference* (McGowan, 2002).

Additional readability problems are addressed as a direct result of the original document's file format: PDF. Chapters 3 and 4 discuss issues which arise when converting PDF files to text, for instance, trying to eliminate undesirable text such as those seen in figures. Although the algorithm used successfully eradicates objectionable text, there are instances where it remains in place. For example, Figure 5.5 is a screenshot of a figure in the thesis that falls in the middle of a sentence. In this example, only a small amount of figure data is successfully deleted, while others, including the figure's label, remain. The offending data instigates errors in the sentence. The original sentence states,

> Interestingly, the prefix trees over the same inputs had similar space savings, but different time savings.


The build time savings, however, differed between the two implementations. In LL, the suffix tree build time ranged from 27%-54% of the time for the whole tree. Interestingly,

68

	Whole tree	Delim tree
internal nodes	0.395*N*	0.171*N*
leaves	*N*	0.837*N*
max depth	15496	167
max freq	2015565	481108
LL build time (min)	220	120
LLHT build time (min)	42	2

the prefix trees over the same inputs had similar space savings, but different time savings

Figure 5.5: An example of a figure in the middle of a sentence.

The addition of figure data has dramatically altered the sentence to read,

> Interestingly, Table 3.4: Comparison of whole and delimited suffix trees: on FAO. internal nodes max depth max freq LL build time (min) LLHT build time (min) the prefix trees over the same inputs had similar space savings, but different time savings.

In the summary we find four substantially large sentences suffering from this affliction. The addition of these erroneous sentences in the summary causes the entire summary to be muddled and incoherent.

In the second M.Sc. thesis, Automatic Text Summarization Using Lexical Chains: Algorithms and Experiments (Kolla, 2004), Kolla details lexical chaining and automatic text summarization. As with McGowan's thesis, Kolla also uses the abstraction method to manually generate his abstract. Fifteen key terms and topics were extracted from his abstract (Figure 5.6). This example fares better than the previous one in the ratio of topics successfully included in the summary compared to those missing: McGowan's generated summary had a success:missing ratio of 7:11, while Kolla's ratio is 8:7. 39% of the selected key terms and topics appear

Information Successfully Included:	**Missing Information:**
• lexical cohesion	• Natural Language Processing
• sequence of related words in the text	• Information Retrieval
• using lexical chains to generate summaries by extracting sentences	• identifying topically related concepts
• saliency	• filtering portions of extracted sentences
• selecting sentences from clusters	• multi-document summarization
• headline generation	• identifying themes
• query based summarization	• filtering portions of extracted sentences
• using gloss relations to compute lexical chains	

Figure 5.6: Comparison between Kolla's (2004) M.Sc. thesis abstract and the Summarizer's version.

in McGowan's automatically generated summary, while 53% of Kolla's appear. One hypothesis for McGowan's summary containing fewer terms and topics than Kolla's is that McGowan's summary contains a greater number of erroneous sentences, resulting from inserted figure/table, than Kolla's summary.

Recall that in McGowan's summary (Figure 5.4) there existed four substantially large, objectionably altered sentences, while in Kolla's summary (Figure 5.7), we find only one sentence in which data has erroneously altered the sentence. In Kolla's summary, eliminating data has malformed the sentence to exclude required information. For example, Figure 5.8 is a screenshot of a sentence containing mathematical equations. The algorithm used to filter figure data also filtered the equations and mathematical symbols; consequently, the sentence was transformed to read,

> This can be computed as shown below: where LCS is the LCS score of the union of the longest common subsequence between the reference sentence and the candidate summary C. u,m refer to the number of sentences and number of words in reference summary, and v and n refer to the number of sentences and words in candidate summary. (current evaluations.).

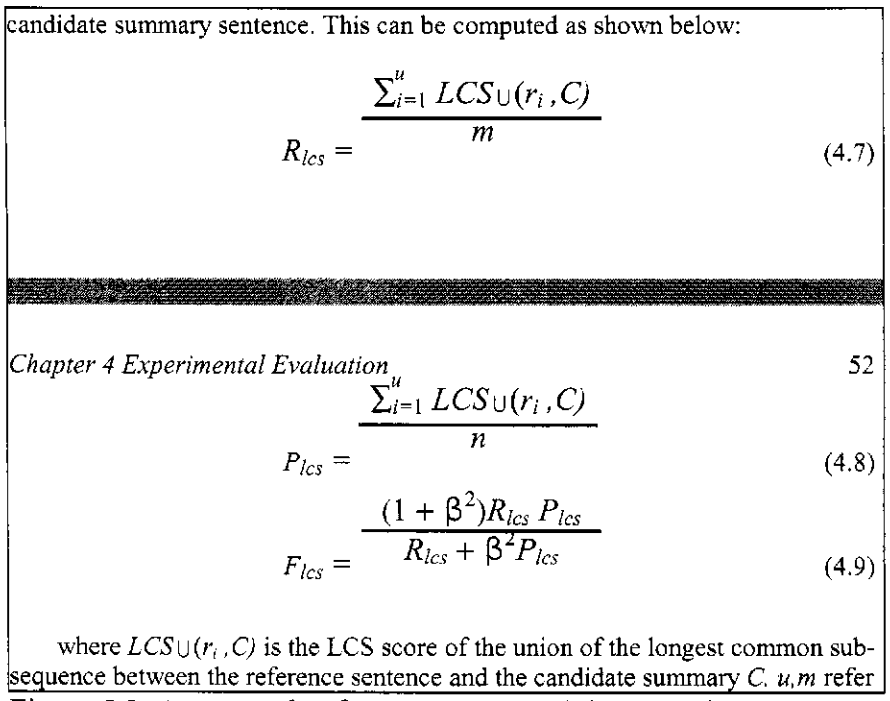
Automatically Generated Summary:

The process of summarization, as shown in Figure 1.1, can be sub-divided into three stages (Mani and Maybury, 1999) (Mani, 2001) (Sparck-Jones, 1999): Analysis: This phase builds an internal representation of the source. Lexical cohesion does not occur just between two words but a sequence of related words spanning the entire text, lexical chains (Morris and Hirst, 1991). Lexical cohesion is the device to hold the text together based on the semantic or identical relations between the words of the text (Morris and Hirst, 1991). In order to extract the noun concepts present in the gloss of a synset, we need to query the noun.xml file with the synsetId. is collection of programs, used to create and query full text inverted index of a document collection. Florida is having a really bad weather. - Synonym relation (words belonging to the same synset in WordNet): eg:- Not all criminals are outlaws. - Hypernym/Hyponym relation: eg:- Peter bought a computer. We describe a system to compute lexical chains as an intermediate representation for the source and extract sentences as summaries to satisfy certain criteria. Barzilay and Elhadad (1997) shows the disadvantage of not considering the compound form of words in formation of lexical chains. Using these principles, saliency of a segment can be determined by the number of lexical chains it has in common with various segments. Sentences from each cluster are selected based on the following principles: - Sentences that do not begin with a pronoun. - Sentences that do not have some quotations; - Sentences that have the entity name. This method is known as Jackknifing procedure and is applied to all ROUGE measures in the ROUGE evaluation package. This can be computed as shown below: where LCS is the LCS score of the union of the longest common subsequence between the reference sentence and the candidate summary C. u,m refer to the number of sentences and number of words in reference summary, and v and n refer to the number of sentences and words in candidate summary. (current evaluations.) Table 4.3 shows the results of ROUGE evaluation. We found that our system achieved better results in Chapter 5 Conclusion and future work headline generation and in query based summarization in context of DUC (Over, We wish to pursue further research in the following directions: Our method to compute lexical chains includes the gloss relations.

Figure 5.7: Automatically generated summary for the thesis entitled *Automatic Text Summarization Using Lexical Chains: Algorithms and Experiments* (Kolla, 2004).

The equations, which reside between "shown below:" and "where LCS" in Figure 5.8, have been removed; additionally, mathematical symbols are missing inside "(current evaluations.)".

Earlier we mentioned the ratio of key terms/topics successfully appearing in the generated summaries compared to those missing. One of the reasons for the lack of terms is the malformed sentence issue seen above. It stands to reason that


candidate summary sentence. This can be computed as shown below:

$$R_{lcs} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{m} \qquad (4.7)$$

Chapter 4 Experimental Evaluation 52

$$P_{lcs} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{n} \qquad (4.8)$$

$$F_{lcs} = \frac{(1 + \beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \qquad (4.9)$$

where $LCS_{\cup}(r_i, C)$ is the LCS score of the union of the longest common subsequence between the reference sentence and the candidate summary C. u, m refer

Figure 5.8: An example of a sentence containing equations.

a thesis containing more tables and figures would cause additional flawed sentences to appear in the summary. The first thesis contains 80 tables and figures with a 39% key term success rate. The second thesis contains 25 tables and figure with a 53% key term success rate. Although it is tempting to find a solution to this problem (such as refining the objectionable data extraction algorithm), this chapter is dedicated to analyzing genres, not technical issues. Even with this issue, the M.Sc. thesis genre does not perform well in automatically generating summaries with the Summarizer.

5.2.3 Analyzing the Novel Genre

The novel genre is vastly different from those seen above in terms of document size. For this reason, the Summarizer has substantially greater text to examine in order to determine which salient sentences will be included in their generated summaries. Ergo, the novel genre's summaries may not contain as high a ratio of key terms/topics as seen earlier. Two works, a work of non-fiction showing novel-like qualities, and a novel were selected for analysis, *Eminent Victorians* (Strachey, 1918) and *Moby Dick* (Melville, 1851).

Eminent Victorians is a biased, and controversial, biographical style work looking at the lives of four prominent people of the Victorian era: Henry Edward Manning, Florence Nightingale, Dr. Thomas Arnold, and Charles George Gordon. Magill's masterplot (Magill, 1979a) for *Eminent Victorians* begins with Strachey's description of the first Victorian's life, Henry Edward Manning. A quick overview of Strachey's view of Manning's life, as described in the masterplot, begins with his aspirations of leading a life in politics. His political plans did not work out, so he became an archdeacon in the Church of England. Manning decided to convert to Roman Catholicism, which, unfortunately, caused a break in his friendship with Gladstone. Manning experienced a series of career triumphs and advances, and made influential friends at the Vatican. He was appointed the supreme commander of the Roman Catholic Church in England, and later a cardinal. At his death, crowds of working people swarmed his funeral procession route.

The size of the summary generated by the Summarizer is based on the size of Magill's masterplot and is significantly long. For this reason, Figure 5.9 contains the subset of the generated summary pertaining to Henry Edward Manning. The beginning of the summary mentions young politicians, Ministers, and Gladstone,

> In those days the Union was the recruiting-ground for young politicians; Ministers came down from London to listen to the

Eminent Victorians Automatically Generated Summary:

His life was extraordinary in many ways, but its interest for the modern inquirer depends mainly upon two considerations--the light which his career throws upon the spirit of his age, and the psychological problems suggested by his inner history. In those days the Union was the recruiting-ground for young politicians; Ministers came down from London to listen to the debates; and a few years later the Duke of Newcastle gave Gladstone a pocket borough on the strength of his speech at the Union against the Reform Bill. The fervours of piety, the zeal of Apostolic charity, the enthusiasm of self-renunciation-- these things were all very well in their way and in their place; but their place was certainly not the Church of England. The question seems difficult to answer, but Keble had, as a matter of fact, forestalled the argument in the following passage, which had apparently escaped the notice of the Rev. Mr. Maitland: 'Now whether the facts were really so or not (if it were, it was surely by special providence), that Abraham's household at the time of the circumcision was exactly the same number as before; still the argument of St. Barnabas will stand. Manning was obviously marked out as his successor, but the new bishop happened to be a low churchman, an aggressive low churchman, who went so far as to parody the Tractarian fashion of using Saints' days for the dating of letters by writing 'The Palace, washing- day', at the beginning of his. The series has subsequently been continued by a more modern writer whose relation of the history of the blessed St. Mael contains, perhaps, even more matter for edification than Newman's biographies. Hurrell Froude had died before Newman had read the fatal article on St. Augustine; but his brother, James Anthony, together with Arthur Clough, the poet, went through an experience which was more distressing in those days than it has since become; they lost their faith. Already, before his illness, these doubts had begun to take possession of his mind. 'I am conscious to myself,' he wrote in his Diary, 'of an extensively changed feeling towards the Church of Rome ... Such was the position of affairs when, two years after Errington's appointment, Manning became head of the Oblates of St. Charles and Provost of the Chapter of Westminster. The difficulties before him were very great; not only had a new University to be called up out of the void, but the position was complicated by the presence of a rival institution--the undenominational Queen's Colleges, founded by Peel a few years earlier with the object of giving Irish Catholics facilities for University education on the same terms as their fellow-countrymen. Such were Newman's thoughts when an unexpected event occurred which produced a profound effect upon his life: Charles Kingsley attacked his good faith, and the good faith of Catholics in general, in a magazine article. Newman smiled grimly at this; he declared to a friend that the letter was 'insolent'; and he could not resist the temptation of using his sharp pen. Once more, for a moment, the eyes of all Christendom were fixed upon Rome. Manning always believed that this was the direct result of Mr. Russell's dispatches, which had acted as an antidote to the poison of Lord Acton's letters, and thus carried the day. The definition itself was perhaps somewhat less extreme than might have been expected. And when the fatal paragraph was read in Rome, might it not actually lead to the offer of the Cardinalate being finally withheld? Towards those who gathered about him, the dying man was still able to show some signs of recognition, and even, perhaps, of affection; yet it seemed that his chief preoccupation, up to the very end, was with his obedience to the rules prescribed by the Divine Authority. 'I am glad to have been able to do everything in due order', were among his last words. 'Si fort qu'on soit,' says one of the profoundest of the observers of the human heart, 'on peut eprouver le besoin de s'incliner devant quelqu'un ou quelque chose.

Figure 5.9: Henry Edward Manning's portion of the automatically generated summary of *Eminent Victorians* (Strachey, 1918).

> debates; and a few years later the Duke of Newcastle gave Gladstone a pocket borough on the strength of his speech at the Union against the Reform Bill.

This sentence implies Gladstone was one of the young politicians and gives no reference to Manning. In actuality, Manning was ambitious and wanted a political career. Unfortunately, the summary lacks any mention of the sort. In addition, this sentence is the only reference to Gladstone; nowhere else does the summary come close to revealing their friendship. In regards to Manning's career in the Church, the summary does enlighten us with his succession as bishop, followed by his appointment as the "head of the Oblates of St. Charles and Provost of the Chapter of Westminster" (Strachey, 1918), and eventually a cardinal. These career advancements are indicative of the "career triumphs and advances" stated in Magill's (Magill, 1979a) masterplot. On one hand, the summary describes Manning's state of mind upon his deathbed as recognizing people, showing affection, but still showing "obedience to the rules prescribed by the Divine Authority" (Strachey, 1918). On the other hand, the masterplot refers to actions after his death with the crowds of people surrounding his funeral procession.

The summary does fairly well in mirroring topics shown in the masterplot, but at a price. The sentences in the summary are structurally sound in that we find very few, if any, lines that do not read well on their own. Unfortunately, when these very sentences are combined to form the entire summary, the reader easily becomes lost in the details. In actuality, this is what one could expect from extracting sentences from a vast document. There are immense sections of text between the extracted lines. These sections include details that could be used to

Eminent Victorians Automatically Generated Summary:

Florence announced an extreme desire to go to Salisbury Hospital for several months as a nurse; and she confessed to some visionary plan of eventually setting up in a house of her own in a neighbouring village, and there founding 'something like a Protestant Sisterhood, without vows, for women of educated feelings'. For dark as had been the picture of the state of affairs at Scutari, revealed to the English public in the dispatches of "The Times Correspondent", and in a multitude of private letters, yet the reality turned out to be darker still. Vast quantities of valuable stores sent from England lay, she found, engulfed in the bottomless abyss of the Turkish Customs House. The situation had been embittered by rumours of religious dissensions, while the Crimean nurses were Roman Catholics, many of those at Scutari were suspected of a regrettable propensity towards the tenets of Dr. Pusey. For instance, I was called up last night, a dragoon having come express with a telegraphic message in these words, "Lord Panmure to General Simpson-- Captain Jarvis has been bitten by a centipede. poised above the Bison's head, and more than once she was actually on the point of really dropping it-- for his recalcitrancy grew and grew. Miss Nightingale was surprised and mortified; she had thought better of Mr. Mill; for surely her proof of the existence of God could hardly be improved upon. 'A law,' she had pointed out, 'implies a lawgiver.' He discussed with her in a series of enormous letters the problems of religion and philosophy; he criticised her writings on those subjects with the tactful sympathy of a cleric who was also a man of the world; and he even ventured to attempt at times to instil into her rebellious nature some of his own peculiar suavity. 'I sometimes think,' he told her, 'that you ought seriously to consider how your work may be carried on, not with less energy, but in a calmer spirit.

Figure 5.10: Florence Nightingale's portion of the automatically generated summary of *Eminent Victorians* (Strachey, 1918).

link the extracted sentences together and convey additional information for the reader, thereby increasing the summary's readability and coherence. It is common for the summary to include references to people or events only once, such that the reader does not know their background, nor why they are mentioned.

Magill's masterplot (Magill, 1979a) continues with the second Victorian, Florence Nightingale. The quick overview begins with a reference to Florence as the "Lady with a Lamp". Strachey characterized Florence as being "extremely capable but she was also tiresomely demanding and disagreeable" (Magill, 1979a). He divided her accomplishments into two phases: her "contribution to the welfare of British wounded during the course of the Crimean campaign"; and her efforts after the war to "transform the Army Medical Department, revolutionize hospital services" and reform the War Office. Florence Nightingale became a legend at the age of ninety. At the end of her life, she received the Order of Merit.

As with Figure 5.9, Figure 5.10 contains the subset of the Summarizer's version of Eminent Victorians that contains information pertaining to Florence Nightingale. The summary begins by informing the reader of Florence's desire to be a nurse at the Salisbury Hospital. It does not refer to her title as the "Lady with a Lamp". Although it mentions Crimean nurses, it does not explicitly state that Florence was one of them. In the masterplot, Strachey portrays Florence as "having a demonic fury"; the last line of the summary implies this,

> he even ventured to attempt at times to instil into her rebellious nature some of his own peculiar suavity. 'I sometimes think,' he told her, 'that you ought seriously to consider how your work may be carried on, not with less energy, but in a calmer spirit (Strachey, 1918).

There are quite a few key points listed in Magill's masterplot that are overlooked in the summary, such as her contributions to the wounded during the Crimean campaign, her efforts to transform the Army Medical Department, revolutionize hospital services, and reform the War Office. Furthermore, it neglects to inform readers about Florence becoming a legend and receiving the Order of Merit.

The generated summary fails to include many of the topics seen in Magill's masterplot. It too bears the same readability issues seen in Manning's summary. Florence's summary has some interesting lines that draws the reader into wanting to read more to discover what it is happening. For example,

> With supreme skill, she kept this sword of Damocles poised above the Bison's head, and more than once she was actually on the point

Eminent Victorians Automatically Generated Summary:

All who knew him during these years were profoundly impressed by the earnestness of his religious convictions and feelings, which, as one observer said, 'were ever bursting forth'. That the classics should form the basis of all teaching was an axiom with Dr. Arnold. 'The study of language,' he said, 'seems to me as if it was given for the very purpose of forming the human mind in youth; and the Greek and Latin languages seem the very instruments by which this is to be effected.' Dr. Arnold himself occasionally visited them, in Rugby; and the condescension with which he shook hands with old men and women of the working classes was long remembered in the neighbourhood. Once or twice he found time to visit the Continent, and the letters and journals recording in minute detail his reflections and impressions in France or Italy show us that Dr. Arnold preserved, in spite of the distractions of foreign scenes and foreign manners, his accustomed habits of mind.

Figure 5.11: Dr. Thomas Arnold's portion of the automatically generated summary of *Eminent Victorians* (Strachey, 1918).

> of really dropping it-- for his recalcitrancy grew and grew (Strachey, 1918)

Engages the reader into wondering why she has a sword and what will she do with it. If the purpose of writing a summary is to hook the reader into reading the entire document, then perhaps this summary is successful.

Dr. Thomas Arnold is Strachey's fourth Victorian. In the masterplot, he is considered the father of the British public school system. According to Magill (Magill, 1979a), Strachey tributes Dr. Arnold with the "worship of athletics and the worship of good form." He goes on to say Dr. Arnold is "the apostle of ideas obviously harmful and absurd" (Magill, 1979a).

In the Summarizer's version (Figure 5.11), Dr. Arnold believed the classics should shape the foundations of teaching. It also mentions his statement that the study of language "was given for the very purpose of forming the human mind in youth" (Strachey, 1918). This sentence in the summary comes the closest to Magill's statement regarding Dr. Arnold's, "worship of good form". In addition, the closest the summary comes to Magill's reference to Dr. Arnold's, "worship of athletics", is in its statement about Dr. Arnold occasionally visiting students playing rugby. Interestingly, while the masterplot depicts Strachey's view of Dr. Arnold as being "the apostle of ideas obviously harmful and absurd", the summary has high regards for Dr. Arnold. For example, the line

> All who knew him during these years were profoundly impressed by the earnestness of his religious convictions and feelings, as one observer said, 'were ever bursting forth'

shows how Magill's masterplot and the generated summary contradict each other.

According to Magill's masterplot (Magill, 1979a), Charles George Gordon started out as a "mischievous, unpredictable boy, he developed into an undisciplined, unpredictable man". He was known for his heroic exploits, and his passion for religion. Gordon is characterized as being unsociable, icy, and possessing a bad temper that he took out on servants and subordinates. When he was in his fifties, the English government sent Gordon on a diplomatic mission to Africa where he was assigned to evacuate the Sudan by British forces. Unfortunately, the assignment ended tragically for Gordon at Khartoum.

In the Summarizer's version (Figure 5.12), the only reference to General Gordon's exploits in Africa, as mentioned in the masterplot, was the British military's movements in the Eastern Sudan. The masterplot discusses General Gordon's diplomatic mission in Africa with a mention of its tragic ending at Khartoum. The generated summary suggests that General Gordon did not appear to be in any serious danger in Khartoum. In fact, it was expected that it was Gordon's duty to survive the events occurring there. Although it is difficult to follow events in the summary, it alludes to a major tragedy occurring in Khartoum

Eminent Victorians Automatically Generated Summary:

He is a glorious fellow!' The followers of the Mahdi, dressed, in token of a new austerity of living, in the security; they were ready, if necessary, to take the field against the Mahdi with English troops. While these messages were flashing to and fro, Gordon himself was the reconquest, but the abandonment of the Sudan; Gordon, indeed, Not only did he find himself deprived, by the decision of the Government, both of the hope of Zobeir's assistance and of the prospect of smashing up the Mahdi with the aid of British troops; the military movements in the Eastern Sudan produced, at the very same moment, a yet more fatal consequence. If, indeed, it should turn out to be the fact that General Gordon was in serious danger, then, no doubt, it would be necessary to send a relief expedition to Khartoum. For him, Sir Evelyn Baring was the embodiment of England-- or rather the embodiment of the English official classes, of English diplomacy, of the English Government with its hesitations, its insincerities, its double-faced schemes. In his opinion it was General Gordon's plain duty to have come away from Khartoum. The fate of General Gordon, so intricately interwoven with such a mass of complicated circumstance with the policies of England and of Egypt, with the fanaticism of the Mahdi, with the irreproachability of Sir Evelyn Baring, with Mr. Gladstone's mysterious passions-- was finally determined by the fact that Lord Hartington was slow. The treacherous Sheikh was an adherent of the Mahdi, and to the Mahdi all Colonel Stewart's papers, filled with information as to the condition of Khartoum, were immediately sent. Owing to a misunderstanding, he believed that Sir Evelyn Baring was accompanying the expedition from Egypt, and some of his latest and most successful satirical fancies played around the vision of the distressed Consul-General perched for days upon the painful eminence of a camel's hump. 'There was a slight laugh when Khartoum heard Baring was bumping his way up here-- a regular Nemesis.' It was not until December 30th--more than a fortnight after the last entry in Gordon's Journal--that Sir Herbert Stewart, at the head of 1,100 British troops, was able to leave Korti on his march towards Metemmah, 170 miles across the desert. Thirteen years later the Mahdi's empire was abolished forever in the gigantic hecatomb of Omdurman; after which it was thought proper that a religious ceremony in honour of General Gordon should be held at the palace at Khartoum.

Figure 5.12: Charles George Gordon's portion of the automatically generated summary of *Eminent Victorians* (Strachey, 1918).

with Gordon dying there.

In general, the Summarizer does successfully cover many issues seen in Magill's masterplot of *Eminent Victorians*. Unfortunately, there remain those that are missed. Readability is affected by the way the Summarizer works in selecting which salient sentences to include in the summary (Chapter 2, p. 39), so it is not surprising that when they are combined the reader is left with unanswered questions. For instance, references to people, places, and events may occur only once, leaving the reader wondering who they are, why they are mentioned, and how they pertain to the overall story. Furthermore, the combination of these sentences appears awkward in some instances. Because of the shear size of *Eminent Victorians* as input text for the Summarizer, there occur vast chunks of text residing between selected sentences that, if included, could answer those questions for the reader, as well as help the summary flow more smoothly. Flow value can be extremely misleading. At the end of this chapter, we will look at a summary (*The Adventures of Tom Sawyer*) that flows well, but is otherwise misleading.

The second novel-like work selected for analysis is Herman Melville's *Moby Dick*. The story is a symbolic allegory of good versus evil. According to Magill (Magill, 1979b),

> Melville intended to indicate in this work the disaster which must result when man constitutes himself a god and sets out to eliminate a force established by God throughout the universe. The whale symbolizes evil, but Ahab, in believing that alone he could hope to destroy it, was also evil.

His masterplot for *Moby Dick* is 2,928 words long; the summary generated by the Summarizer is 3,095 words. As it is so immense, we will look at a sample of the summary (the first 1,660 bytes) in Figure 5.13 to get a feel for how it flows.

The summary begins with many of the extracts Melville included in the novel. In view of the fact that the Summarizer extracts sentences for use in the summary, it is not surprising that one of these extracts is incomplete and taken out of context, "Among the former, one was of a most monstrous size" is the second sentence within an extract from Tooke's Lucian, *The True History* (Figure 5.14).

Moby Dick Automatically Generated Summary:

Among the former, one was of a most monstrous size. "Silly Mansoul swallowed it without chewing, as if it had been a sprat in the mouth of a whale." --PILGRIM'S PROGRESS. "The Whale-ship Globe, on board of which vessel occurred the horrid transactions we are about to relate, belonged to the island of Nantucket." --"NARRATIVE OF THE GLOBE," BY LAY AND HUSSEY SURVIVORS. "It is generally well known that out of the crews of Whaling vessels (American) few ever return in the ships on board of which they departed." --CRUISE IN A WHALE BOAT. What do you see?--Posted like silent sentinels all around the town, stand thousands upon thousands of mortal men fixed in ocean reveries. Finally, I always go to sea as a sailor, because of the wholesome exercise and pure air of the fore-castle deck. For my mind was made up to sail in no other than a Nantucket craft, because there was a fine, boisterous something about everything connected with that famous old island, which amazingly pleased me. The picture represents a Cape-Horner in a great hurricane; the half-foundered ship weltering there with its three dismantled masts alone visible; and an exasperated whale, purposing to spring clean over the craft, is in the enormous act of impaling himself upon the three mast-heads. A still duskier place is this, with such low ponderous beams above, and such old wrinkled planks beneath, that you would almost fancy you trod some old craft's cockpits, especially of such a howling night, when this corner-anchored old ark rocked so furiously. But the fare was of the most substantial kind--not only meat and potatoes, but dumplings; good heavens! dumplings for supper!

Figure 5.13: The first 1,660 bytes of the generated summary for *Moby Dick* (Melville, 1851).

Although this extract was taken out of context, the other extracts were not: Pilgrim's Progress; Narrative of the Globe, by Lay and Hussey Survivors; and Cruise in a Whale Boat. Even with extracts included in their full context, they do not serve in assisting the summary's readability.

We find poor readability throughout the summary. The sentences jump from one topic to the next without any intermediate connecting them; in short, the summary does not make any sense. For example, the second last sentence in the summary sample (Figure 5.13) paints the picture of a public room in an inn; the last sentence describes what the sailor had for dinner.

We discovered in Chapter 3 that the Summarizer occasionally dies upon extremely large text. The remedy was to truncate its input text to a more

"Scarcely had we proceeded two days on the sea, when about sunrise a great many Whales and other monsters of the sea, appeared. **Among the former, one was of a most monstrous size.** ... This came towards us, open-mouthed, raising the waves on all sides, and beating the sea before him into a foam." –TOOKE'S LUCIAN. "THE TRUE HISTORY."

Figure 5.14: Complete extract taken from *Moby Dick*. The bolded sentence was extracted for use in the summary.

Moby Dick Automatically Generated Summary:

"Pull, pull, my good boys," said Starbuck, in the lowest possible but intensest concentrated whisper to his men; while the sharp fixed glance from his eyes darted straight ahead of the bow, almost seemed as two visible needles in two unerring binnacle compasses. The repeated specific allusions of Flask to "that whale," as he called the fictitious monster which he declared to be incessantly tantalizing his boat's bow with its tail--these allusions of his were at times so vivid and life-like, that they would cause some one or two of his men to snatch a fearful look over the shoulder.

Figure 5.15: The last portion of the generated summary for *Moby Dick* (Melville, 1851).

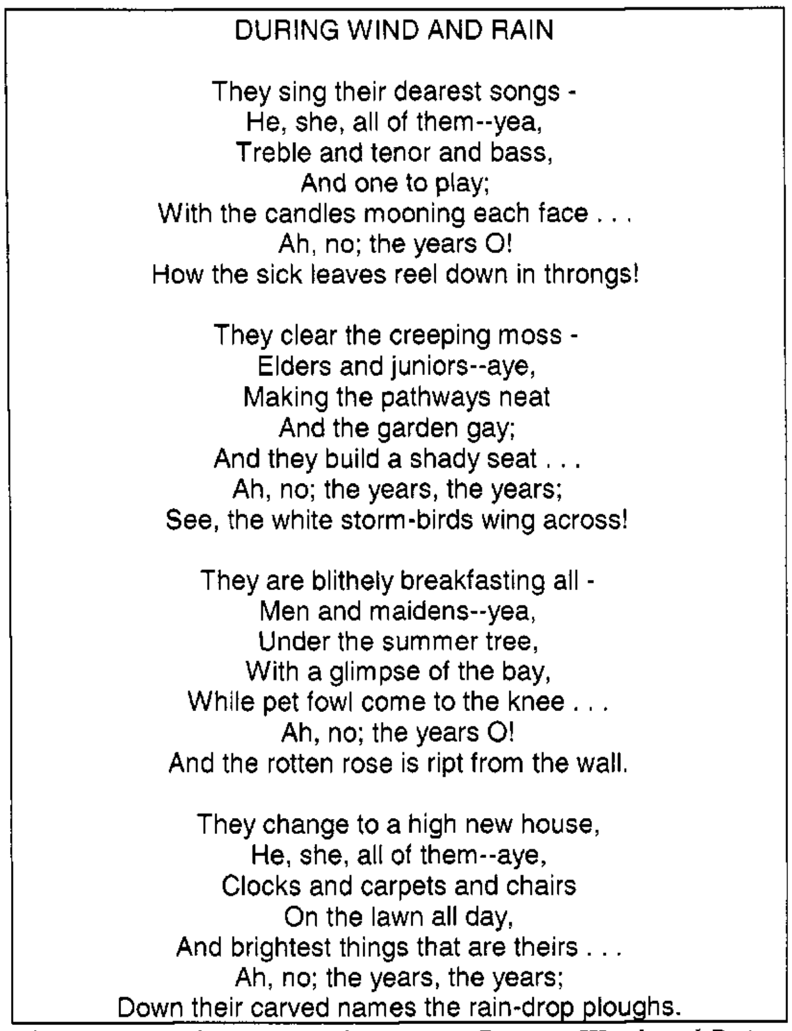
manageable size. Moby Dick was one such document. Because of this truncation, the entire text could not be processed, thereby generating a summary with a different ending than the novel. Figure 5.15 shows the generated summary's ending. The final sentence of the summary actually appears one quarter of the way into the novel. The poor readability and misleading ending in *Moby Dick* makes it a fine example of a novel summary gone amiss.

5.2.4 Analyzing the Poetry Genre

Webster (www.webster.com) defines poetry as,

> writing that formulates a concentrated imaginative awareness of experience in language chosen and arranged to create a specific emotional response through meaning, sound, and rhythm.

Thomas Hardy's poem, *During Wind and Rain* (Figure 5.16) is a beautiful example of poetry that follows this definition. In it, Hardy depicts the happy moments of his dead wife's childhood, contrasted with the inevitability of death. Each stanza begins with a happy recollection from her childhood, and ends with


DURING WIND AND RAIN

They sing their dearest songs -
He, she, all of them--yea,
Treble and tenor and bass,
And one to play;
With the candles mooning each face . . .
Ah, no; the years O!
How the sick leaves reel down in throngs!

They clear the creeping moss -
Elders and juniors--aye,
Making the pathways neat
And the garden gay;
And they build a shady seat . . .
Ah, no; the years, the years;
See, the white storm-birds wing across!

They are blithely breakfasting all -
Men and maidens--yea,
Under the summer tree,
With a glimpse of the bay,
While pet fowl come to the knee . . .
Ah, no; the years O!
And the rotten rose is ript from the wall.

They change to a high new house,
He, she, all of them--aye,
Clocks and carpets and chairs
On the lawn all day,
And brightest things that are theirs . . .
Ah, no; the years, the years;
Down their carved names the rain-drop ploughs.

Figure 5.16: Thomas Hardy's poem *During Wind and Rain* (Hardy, 1917).

death and decay. For example, the first stanza tells the tale of the family sitting around the piano, singing. It ends by describing leaves drying out and falling from the trees. Each stanza contains a repetitious line, "Ah, no; the years O!" that refers to time passing. The rhythm of the poem flows like a song.

While it is not customary to summarize poetry, it is interesting to see how it fares when run through the Summarizer. The length of the original poem is 976

DURING WIND AND RAIN

They sing their dearest songs -
He, she, all of them--yea,
Treble and tenor and bass,
And one to play;
With the candles mooning each face . . .

They are blithely breakfasting all -
Men and maidens--yea,
Under the summer tree,
With a glimpse of the bay,
While pet fowl come to the knee . . .

Figure 5.17: Automatically generated summary for the Hardy's poem *During Wind and Rain* (Hardy, 1917).

bytes; the generated summary was set to be 250 bytes in length.[18] As we can see in Figure 5.17, processing Hardy's poetry through the Summarizer results in a summary omitting the repetition, "Ah, no; the years O!" The reason for this is as follows.

The first stage of the Summarizer is to separate (tokenize) each sentence of text to appear on a separate line. In poetry, sentences do not follow the conventional rules. For example, the first five lines of each stanza begin with a capital and ends in punctuation. In conventional rules, proper nouns and words at the beginning of each sentence are capitalized. When the Summarizer tokenized the poem, the first five lines of the stanza were combined as a sentence and placed on its own line.[19] For example, the tokenized version of the first sentence of the poem reads,

[18] The original version of the Summarizer was written with summary length set to 500 bytes. The algorithm used to generate the length extracts sentences until it reaches 500 bytes. Consequently, summaries commonly reach lengths greater than 500 bytes because the length of the final extracted sentence is normally greater than the difference needed to reach its goal. Therefore, setting the summary length is an approximation. The result of setting the summary length of the poem to 250 bytes actually produced a summary 327 bytes long.

[19] Lines that were combined by the Tokenizer were manually separated in Figure 5.17 to mirror those seen in the original poem (Figure 5.16).

> They sing their dear songs – He, she, all of them—yea, Treble and tenor and bass, And one to play; With the candles mooning each face . . .

After tokenizing, the poem is transformed from twenty-eight lines (seven lines per stanza, four stanzas in total) to ten lines (ten sentences, one per line). The first sentence of each stanza contains numerous nouns; the repetitious sentences do not contain as many. Since the Summarizer selects salient sentences based on scoring equations described in Chapter 2, noun-rich sentences are more apt to appear in the summary; hence, the omission of Hardy's repetition in the generated summary. In conclusion, the lack of repetition in the summary takes away from the feeling of the poem; for this reason the poetry genre does not fare well to automatic summarization.

5.3 When Should the Summarizer be Used and Not Used?

There are certain genres that the Summarizer works better than others. Earlier we saw how newspaper articles are an ideal genre for automatically generating summaries. Figures 5.18 and 5.19 contain two summaries generated by the Summarizer from this genre. Although the first summary provides plenty of information, it does not tell a story. The second summary both provides information and is easily readable. Both summaries arise from newspaper articles, and both articles contain the same topic: basketball. The question remains: why does one summary read better than the other does?

Looking at the first summary we see it is dominated by proper nouns and numbers, with each sentence filled with names, basketball scores, and ages. In fact, its corresponding article is a roundup; it is consumed by basketball scores

Boys' Basketball Roundup; Parks Scores 28 As Marina Upsets Fountain Valley, 71-68

Center Cherokee Parks scored 28 points, including a pair of free throws with 22 seconds remaining, as Marina High School upset Fountain Valley, 71-68, Wednesday in a Sunset League boys' basketball game at Marina. Fountain Valley (17-6 overall, 5-4 in league play) had moved to within a point, 66-65, with 59 seconds remaining when guard Doug Weaver hit a jump shot, but Marina (10-14, 3-6) scored the upset with some accurate free throw shooting down the stretch. Cypress 56, Katella 46 -- Mark Johnson scored 21 points to lead Cypress (11-12, 6-3) at Katella. Mission Viejo 51, San Clemente 49 -- Matt Turner scored 17 points to lead host Mission Viejo (10-12, 5-4).

Figure 5.18: Example of a noun-rich summary.

The Center Of Attention; Basketball: Ingrid Dixson Has Averaged 25.1 Points And 18.6 Rebounds To Carry West Covina To A Perfect 12-0 Record In The Sierra League.

Even before she played a minute of varsity basketball in high school, Ingrid Dixson was receiving letters of inquiry from college recruiters. At 6-foot-3 and 180 pounds, Dixson's potential appears to be approaching its fruition as a senior at West Covina High -- which merged with Edgewood after her sophomore year. Dixson has averaged a valley-leading 25.1 points and 18.6 rebounds to carry West Covina to a perfect 12-0 record in the Sierra League and 17-4 overall. Torricelli, for one, doesn't think the adjustment from high school to college will be that difficult for Dixson.

Figure 5.19: Example of a narrative summary.

and statistics. Although the summary is informative, it is not that interesting to read.

In the second summary, we also see proper names and numbers, but it differs from the first in that it tells a story. At the onset, the reader is drawn into the tale, "Even before she played a minute". Later the reader learns that Ingrid is a basketball player and is highly regarded by college recruiters. It continues by reporting her playing stats and concludes with a prediction as to Ingrid's adjustment from high school to college. This summary is interesting to read as well as informative.

According to Chapter 2, the Summarizer selects salient sentences based on nouns. The summary in Figure 5.18 is based on a document that is rich in nouns, specifically names and basketball stats. The summary in Figure 5.19 is similar to the first in that it provides statistical information. The main difference is the second summary is narrative; its originating article has a narrative nature. This demonstrates that non-narrative genres do not summarize as well as narrative ones.

Earlier we looked at summaries from two artistic prose works, *Eminent Victorians* and *Moby Dick*. The prior was written in a biographic form. The Summarizer performed surprisingly well in including many topics as those seen in the professionally written masterplot. Readability problems arose in both summaries caused by large amounts of text remaining between extracted sentences. Such works are not suitable for automatic summarization. With that in mind, it is entertaining to look at a summary produced from a well-known novel, *The Adventures of Tom Sawyer* by Mark Twain.

In 1876, Mark Twain wrote *The Adventures of Tom Sawyer.* This novel tells exciting tales of an energetic, mischievous, and daring adolescent boy and his escapades along the banks of the Mississippi River during the 1840s. Along with his friend Huck Finn, Tom engages in exploits such as witnessing the murder of Dr. Robinson by Injun Joe, finding buried treasure, and spending three days lost in a cave with his sweetheart Becky Thatcher.

The entertaining summary for this novel is shown in Figure 5.20. The summary gives the impression the novel begins with two people who are making up after a spat over fishing. One of the two is a female, "She began to soften; she felt sorry." The next sentence reveals the other is a male, with a second male

The Adventures of Tom Sawyer Generated Summary:

"What'll you give?" "I don't care whose tick he is--he's on my side of the line, and you sha'n't touch him." She began to soften; she felt sorry. Just at this point he met his soul's sworn comrade, Joe Harper --hard-eyed, and with evidently a great and dismal purpose in his heart. About two o'clock in the morning the raft grounded on the bar two hundred yards above the head of the island, and they waded back and forth until they had landed their freight. While Joe was slicing bacon for breakfast, Tom and Huck asked him to hold on a minute; they stepped to a promising nook in the river-bank and threw in their lines; almost immediately they had reward. Then with a mutual impulse the two bereaved women flung themselves into each other's arms and had a good, consoling cry, and then parted. I'm glad to see him, poor motherless thing!" At last he spied her, but there was a sudden falling of his mercury. The school stared in perplexity at this incredible folly. The first of all the negro minstrel shows came to town, and made a sensation.

Figure 5.20: Automatically generated summary of *The Adventures of Tom Sawyer* (Twain, 1876).

entering the tale, "Just at this point he met his soul's sworn comrade, Joe Harper." The three are on a raft, and dock on a sand bar by an island. As Joe slices bacon, Tom and Huck (one is the female in the story) catch fish for their breakfast. The next line reveals both Tom and Huck are actually females,

> Then with a mutual impulse the two bereaved women flung themselves into each other's arms and had a good, consoling cry, and then parted.

A male (Joe?) spots one of the females and his temperature drops, "there was a sudden falling of his mercury." There must be a school on the island because "the school stared in perplexity at this incredible folly." The tale ends with entertainers coming to town. Obviously, this story is completely off base from the novel. The purpose is to show the comical side of extracting sentences of a well-known novel for generating a summary.

5.4 Summary

This chapter has examined four genres: newspaper, M.Sc. thesis, novel, and poetry. The newspaper genre performed the best, especially when articles are written in a narrative-style; roundup styles did not cope as well. The M.Sc. thesis genre experienced two problems: the first problem is that only some of the topics listed in the author's abstracts appeared in the summary; the second is poor readability. The Summarizer was not useful in the novel genre because many of the summaries were incoherent. Analyzing sub-genres showed interesting results. For instance, it worked better on biographic styles than story type. Topics appeared in the bibliographic style summaries that were also seen in the professional masterplot. Story type novels lacked readability and are inaccurate. With that being said, they can be humorous in certain scenarios, such as in the above example of *The Adventures of Tom Sawyer*. Overall, the Summarizer worked well on the newspaper genre, and poorly with the M.Sc. thesis and novel genres.

Chapter 6

Conclusion and Future Work

This thesis explored the use of automatic text summarization in a practical digital library. The purpose was to supply DL users with automated tools to allow them to examine, at a glance, the contents of documents. The project reported in this thesis provides Greenstone with this tool by implementing it with the Summarizer. This thesis describes the options created specifically for this tool, analyzes how the Summarizer runs inside Greenstone, and analyzes the summaries produced for different genres.

The Summarizer provides automatically generated summaries, extracts nouns and noun phrases for use as metadata for searching and browsing digital collections. Options created for it allow users to tailor summaries to suit their individual needs by:

- Adjusting summary length
- Generating summaries based on the first *n* bytes of input text
- Producing section-level summaries
- Using the first sentence of the document as the first sentence of the summary
- Excluding Unicode in the summary

In this thesis, we concluded that processing speeds were fastest with PDF documents and slowest with large plain text files. The Summarizer spends the majority of its time during the text chunking and lexical chaining stages. Processing a portion of the document significantly increases the Summarizer's speed. Time requirements suggested that it is feasible to produce summaries for collections containing newspaper articles and PDF documents. For collections containing full-length novels, it can be feasible provided the total size of the input text is taken into consideration. For example, a 10,000 MB collection of novels would take 11,236 hours or 468 days, whereas a 100 MB collection would take 112 hours or 5 days.

This thesis project ensures the Summarizer tool robustness. This required altering the Summarizer to handle situations that would otherwise have caused the system to freeze or die. Ensuring that the system will continue running when it encounters problems caused information loss. Information loss occurs in three situations. The first arises when the Tokenizer comes across tokens too large for it to handle, requiring them to be removed for the Summarizer to continue. The second is seen when large documents cause the Segmenter to run out of memory, resulting in the files being truncated. The third appears when users request sentences containing Unicode to be removed from the summary.

The thesis studied how genre affects summaries. Four genres were examined: newspaper, M.Sc. thesis, novel, and poetry. Analysis showed that not all genres perform well; narrative-style genres achieved the best results. Although at first it may appear obvious that newspaper articles are ideal, analysis showed that not all newspaper articles performed the same. Within the newspaper genre, there exist sub-genres, for example roundup and narrative. The roundup genre did not cope well, while the narrative-style functioned the best. M.Sc. thesis and poetry suffered from incoherency and did not fare well to automatic summarization. Although some novels appeared coherent, it was shown that the easy flowing sentences could be misleading. A summary can easily imply different meanings from that of the original novel. In conclusion, as the Summarizer stands, it is only useful for the newspaper genre and proves inoperative for all other genres.

6.1 Statement of Limitations

This project originated as a technical issue when adding the University of Lethbridge Summarizer as a tool in the Greenstone Digital Library Software Suite. In doing so, a number of unanticipated issues presented themselves that required solutions. These issues are significant for future research on integrating automatic text summarization and digital libraries. Had these issues not arisen and remedies implemented out of practicality, I would have designed my research differently.

In the course of the project, it was discovered that the Summarizer did not perform well with genres outside the newspaper scope. Had this been known at the beginning, testing would have been performed using different automatic summarization systems for different genres.

Greenstone is used internationally in a variety of situations, from hosting digital libraries on university websites, to providing information and knowledge for developing countries (Witten, 2003). For example, Human Info NGO and UNESCO used Greenstone for creating development and basic needs CD-ROM libraries for developing countries (*Human Info NGO – we care and share*, 2003). Computers in developing countries are often low-end with minimal software and memory installed on them (Witten, Loots, Trujillo, & Bainbridge, 2001, p. 83). To accommodate for a variety of hardware/software configurations, Greenstone is available for all versions of Windows 3.1 and up, Linux, Unix, BSD, and Mac OSX. The Summarizer was written for, and tested on, Linux. In the course of the project, it was attempted to install the Summarizer on the Windows XP operating system. Unfortunately, problems were encountered when attempting to run the Summarizer. Had the Summarizer run optimally, tests would have been performed on a variety of operating systems, including Windows 3.1.

The computer used for testing was a networked server employed by many people. This resulted in processing speed tests not being as accurate as if they were administered on a dedicated machine. Ideally, given the appropriate hardware, tests would have been administered on a computer dedicated solely to it. It would also have been beneficial to conduct tests on a diversified collection of hardware profiles. Doing so could help resolve processing bottlenecking issues.

Capturing qualitative data would establish whether providing DL users with summaries assisted them in searching and browsing. It would be favourable to extend the study by performing usability evaluations.

6.2 Future Work

This thesis showed that there are areas where the Summarization tool can be improved. When Greenstone converts PDF documents to HTML, it attaches HTML tags in a fashion that make it difficult to extract only meaningful body content. Analysis showed instances where undesirable text remains when extracting content text. In some cases, figure data is included in the middle of sentences, making them erroneous. Improving the text extraction algorithm could increase the readability of PDF documents, thus increasing the feasibility of generating summaries for genres such as M.Sc. theses and scientific papers. Improving the PDF conversion algorithm would be convenient for other research areas that require text extracted from PDF documents.

It would be valuable to include an additional option to the Summarizer tool that takes a list of nouns. Words in this list would be ignored by the Summarizer's noun extraction stage. Because the Summarizer chooses salient sentences based on lexical chains created from extracted nouns, ignoring the words in the list would prevent sentences being selected based on these nouns. For example, M.Sc. theses routinely use the nouns "Figure", "Table" and numbers. As a result, sentences referring to them appear more frequently in the summary, causing poor readability. Supplying an option to ignore these nouns would prevent summaries from referring to figures, thereby increasing their readability.

When work was being performed on the thesis project, the Summarizer that was available only worked with single documents. Since then, a new version has been created to include summaries for multi-documents and query-based. Future work could take advantage of these additional features by incorporating them into the Summarizer tool.

Despite the fact this thesis focuses on the work with the Summarizer and Greenstone, this research can be extended globally. Results disclosed a problem with producing high quality summaries for a variety of genres using a single automatic text summarization system. This brings to the forefront two areas of research, genre classification and summarizing non-newspaper genre.

The Columbia Summarizer (McKeown et al., 2002) performs multi-document summarization for newspaper documents. Documents containing similar topics are clustered together. Columbia uses different summarizer strategies depending on the type of documents in each cluster. A router decides which strategy to use on each cluster. A similar concept would be useful for detecting genre in a digital library. Each file would be routed to the applicable summarization system designed for that document's genre. Kessler, Nunberg, and Schütze (1997) pointed out that genre classification can be extremely important for problems in computational linguistics; such as parsing, POS-tagging, and word-sense disambiguation. This thought can also be extended to include automatic text summarization. Genre classification is not an easy task. When Lee and Myaeng (2002) classified genre based on word statistics they discovered some genre classes share subject-specific terms, thereby confusing the classifier.

The topic of summarization was presented in a workshop on IR research challenges. This workshop found that one of the problems associated with automatic summarization was a lack of research on producing summaries for genre other than newspaper documents (Allan et al., 2003, p.40). A few systems are addressing this issue by producing summaries for specialized documents. For example, the SUM project summarizes legal documents (Grover, Hachey, Hughson, & Korycinski, 2003) and SumUM works with raw technical text (Saggion & Lapalme, 2002). Additional research on automatically summarizing non-newspaper genre needs to be performed.

References

About project gutenberg. (2004). Retrieved June 14, 2005 from http://www .Gutenberg.org/about.

Adams, W.J., Jansen, B.J., & Howard, R. (1998). Distributed digital library architecture: The key to success for distance learning [Electronic version]. *Computers and technology in education.* Cancun, Mexico.

Allan, J., Aslam, J., Belkin, N., Buckley, C., Callan, J., Croft, B., et. al. (2003). Challenges in information retrieval and language modeling: report of a workshop held at the intelligent information retrieval, 37(1) 31-47. NewYork, NY: ACM Press.

Anick, P. G., & Vaithyanathan, S. (1997). Exploiting clustering and phrases for context-based information retrieval [Electronic version]. *Proceedings of the 20th annual international ACM SIGIR conference on research and development in information retrieval,* 314-323. Philadelphia, Pennsylvania: ACM Press.

Bainbridge, D., McKay, D., & Witten I.H. (2003a). *Greenstone digital library - developer's guide.* Manual, University of Waikato, New Zealand. http://prdownloads.sourceforge.net/greenstone/Develop-en.pdf.

Bainbridge, D., Thompson, J., & Witten, I. H. (2003b). Assembling and enriching digital library collections [Electronic version]. *Proceedings of the 3rd*

ACM/IEEE-CS joint conference on digital libraries, 323-334. Houston, Texas: IEEE Computer Society.

Boddie, S. (2003, January 28). *Greenstone word/pdf/ps import problems*. Message posted to Greenstone mailing list, archived at http://www.sadl.uleth.ca/gsdl/cgi-bin/library?e=d-000-00---0gsarch--00-0-0--0prompt-10---4------0-11--1-en-50---20-about---00031-001-1-0utfZz-8-00&a=d&c=gsarch&cl=CL1.7.102&d=002101c2c687$308655f0$7bf4d982-scms.waikato.ac.nz.

Brill, E. (1992). A simple rule-based part of speech tagger [Electronic version]. *Proceedings of the third conference on applied natural language processing*, 152-155. Trento, Italy: Association for Computational Linguistics.

Brunn, M., Chali, Y., Dufour, B. (2002). The University of Lethbridge text summarizer at DUC 2003. *Proceedings of the document understanding conference*: 148-152.

Buckland. M., & Gey, F. (1999). The relationship between recall and precision [Electronic version]. *Journal of the American society for information science*. 45(1) (1999), 12-19. John Wiley & Sons, Inc.

Bush, V. (1996). As we may think. *Interactions*. 3(2) (1996), 35-46. ACM Press: New York, NY, USA. (Original work published in 1945).

Buyukkokten, O., Garcia-Molina, H., & Paepcke, A. (2001). Seeing the whole in parts: text summarization for web browsing on handheld devices [Electronic version]. *Proceedings of the 10th international conference on World Wide Web*, 652-662. New York, NY, USA: ACM Press.

Chali, Y., Kolla, M. (2004). Summarization techniques at DUC 2004. *Proceedings of the document understanding conference*, Boston, May 2004, pp 105-111, NIST.

Chali, Y., Kolla, M., Singh, N., Zhang, Z. (2003). The University of Lethbridge text summarizer at DUC 2003. *Proceedings of the document understanding conference*, Edmonton, June 2003, pp 148-152, NIST.

Chen, C. (2004). Information visualization: Beyond the horizon. Springer.

Choi, F. (2000a). Advances in domain independent linear text segmentation [Electronic version]. *Proceedings of the first conference on North American chapter of the association for computational linguistics*, 26-33. Seattle, Washington: Morgan Kaufmann Publishers Inc.

Choi, F. (2000b). *C99 A domain independent algorithm for linear text segmentation* (Version 1.0) [Computer software]. http://www.lingware.co .uk/homepage/freddy.choi/software/software.htm.

Choi, F. (2000c). Improving the efficiency of speech interfaces for text navigation [Electronic version]. *Proceedings of ICCHP'00*, Karlsruhe, Germany. Retrieved September 10, 2005 from http://www.lingware.co.uk/homepage/ freddy.choi/publication/docs/icchp00.pdf

Crystal spring colony. (n.d.). Retrieved July 5, 2003, from http://www.telusplanet .net/public/mtoll/crystal.htm

De Andrade, M. A., & Del Cima, O. M. (1995). Super-T_3 QED and the dimensional reduction of N=1 super-QED_{2+2} [Electronic version]. *hep-th 9501002*, submitted for publication.

DiStefano, A., Rudestam, K. E., Silverman, R. (2003). *Encyclopedia of distributed learning*. Sage Publications Inc.

Don, K. (n.d.). *MGPP: A search engine for XML documents user guide*. Retrieved February 2, 2005 from http://www.greenstone.org/docs/mgpp_user.pdf.

Dowman, M., Tablan, V., Cunningham, H., & Popov, B. (2005). Content augmentation for mixed-mode news broadcasts [Electronic version]. *3rd European conference on interactive television: User centered ITV systems, programmes and applications*. Aalborg University, Denmark. Retrieved May 19 from http://gate.ac.uk/sale/euro-itv-2005/content-augmentation-for-mixed-mode-news-broadcast-consumption.pdf.

DUC 2003 documents for summarization, tasks, and measures. (2003). Retrieved May 15, 2005 from http://duc.nist.gov/duc2003/tasks.html.

Evans, D. A., & Zhai, C. (1996). Noun-phrase analysis in unrestricted text for information retrieval [Electronic version]. *Proceedings of the 34th annual meeting on Association for computational linguistics*. 17-24. Santa Cruz, California: Association for Computational Linguistics.

Eyambe, L., Suleman, H. (2004). A digital library component assembly environment [Electronic version]. *ACM international conference proceeding* *series*. (75) (2004), 15-22. Republic of South Africa: South African Institute for Computer Scientists and Information Technologists.

Fox, E. A., Akscyn, R. M., Furuta, R. K., & Leggett J. J. (1995). Digital libraries [Electronic version]. *Communications of the ACM*. 38(4) (1995), 22-28. New York, NY, USA: ACM Press.

Gauntlett, K. (2005, June 22). *Wild weather spawns tornadoes*. Calgary Herald, pp. A1,A4.

Gladney, H. M., Belkin, N. J., Ahmed, Z., Fox, E. A., Ashany, R., & Zemankova, M. (1994). Digital library: gross structure and requirements (Report from a workshop) [Electronic version]. *Proceedings workshop on on-line access to digital libraries*, 101-107.

Goncalves, M. A., Fox, E. A., Watson, L. T., & Kipp, N. A. (2004). Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries [Electronic version]. *ACM transactions on information systems (TOIS)*, 270-312. New York, NY: ACM Press.

Grover, C., Hachey, B., Hughson, I., & Korycinski, C. (2003). Automatic summarisation of legal documents [Electronic version]. *Proceedings of the 9th international conference on artificial intelligence and law*, 243-251. New York, NY: ACM Press.

Hernandez, N., & Grau, B. (2003). What is this text about? [Electronic version]. *Proceedings of the 21st annual international conference on documentation*, 117-124. New York, NY: ACM Press.

Human Info NGO – we care and share. (2003). Retrieved March 30, 2006, from http://www.humaninfo.org/main_projects.html.

Jing, H., Barzilay, R., McKeown, K., & Elhadad. M. (1998). Summarization evaluation methods: Experiments and analysis [Electronic version]. *Working notes of the AAAI-98 spring symposium on intelligent text summarization*, 60-68.

Kan, M.-Y. & Klavans, J. L. (2002). Using librarian techniques in automatic text summarization for information retrieval [Electronic version]. *Proceedings of the 2nd ACM/IEEE-CS joint conference on digital libraries*, 36-45. New York, USA: ACM Press.

Kessler, B., Nunberg, G., Schütze, H. (1997). Automatic detection of text genre [Electronic version]. Proceedings of the 35th annual meeting of the association for computational linguistic and the 8th meeting of the European chapter of the association for computational linguistics, 32-38. San Francisco CA: Morgan Kaufmann Publishers.

Kolla, M. (2004). *Automatic text summarization using lexical chains: Algorithms and experiments* (Master of Science Thesis, University of Lethbridge, 2004).

Kruk, M. (2002). *pdftohtml* (Version 0.36) [Computer software]. http://sourcforge .net/projects/pdftohtml.

Lawrence, S. L. (2001). Online or invisible? [Electronic version]. *Nature.* 411(6837) 521.

Lin, C.-Y., & Hovy, E. (2002). Manual and automatic evaluation of summaries [Electronic version]. *Proceedings of the workshop on automatic Summarization post conference workshop of ACL-2002*, 45-51. Philadelphia, PA.

Lin, C.-Y., & Hovy, E. (2003). The potential and limitations of automatic sentence extraction for summarization [Electronic version]. *Proceedings of the HLT-NAACL 2003 workshop on automatic summarization*, 73-80.

Loots, M., Camarzan, D., & Witten, I.H. (n.d.). *From paper to collection*. Retrieved October 5, 2004, from The Greenstone Digital Library Software Web site: http://www.greenstone.org/cgi-bin/library?a=p&p=docs.

Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts [Electronic version]. *IBM journal*. (April, 1958), 159-165. Retrieved September 5, 2005 from http://researchweb.watson.ibm.com/journal/rd/022/luhn.pdf

Magill, F. N. (1979a). *Masterplots* (Vol. III, revised 1st ed.). New Jersey: Salem Press.

Magill, F. N. (1979b). *Masterplots* (Vol. VII, revised 1st ed.). New Jersey: Salem Press.

Mani, I. (1999). Intelligent scalable text summarization. *Encyclopedia of computer science and technology*, (41), 83-98. Marcel Dekker.

Marchionini, G., & Maurer, H. (1995). The roles of digital libraries in teaching and learning [Electronic version]. *Communications of the ACM*. 38(4) (1995), 67-75. New York, NY, USA: ACM Press.

Marshall, C. C., & Bly, S. (2004). Sharing encountered information: Digital libraries get a social life [Electronic version]. *Proceedings of the 4th ACM/IEEE-CS joint conference on digital libraries*, 218-227. New York, NY, USA: ACM Press.

McGowan, K.J. (2002). *Efficient phrase hierarchy inference* (Master of Science Thesis, University of Waikato, 2002).

McKeown, K., Barzilay, R.,Evans,D., Hatzivassiloglou, V., Kan,M., Schiffman, B., & Teufel, S. (2001). Columbia multi-document summarization: Approach and evaluation [Electronic version]. *Proceedings of the document understanding workshop (DUC)*, 2001.

McKeown, K. R., Regina, B., Evans, D., Hatzivassiloglou, V., Klavans, J. L., Nenkova, A., Sable, C., Schiffman, B., Sigelman, S. (2002). Tracking and summarizing news on a daily basis with Columbia's newsblaster [Electron version]. *Proceedings of the human language technology conference (HLT)*, San Diego, CA, USA.

Miller, G. A., Fellbaum. C., Tengi, R., Wolff, S., Wakefield, P., Langone, H., & Haskell, B. (2005). *WordNet* (Version 2.0) [Computer software]. http:// wordnet.Princeton.edu. Princeton University Cognitive Science Laboratory.

MORE: Minnesota opportunities for reference excellence—literature—book reviews, plots, literary criticism. (2003). Retrieved April 6, 2005 from http:// www.arrowhead.lib.mn.us/more/bkreviews.htm.

Pan American Health Organization / Regional Office of the World Health Organization [PAHO/WHO] (1999). *Virtual disaster library* [Digital library]. Retrieved August 11, 2005 from http://www.sadl.uleth.ca.

Parmanto, B., Ferrydiansyah, R., Saptono, A., Song, L., Sugiantara, I. W., Hackett, S. (2005). AcceSS: accessibility through simplification & summarization [Electronic version]. *Proceedings of the 2005 international cross-disciplinary workshop on web accessibility (W4A)*, 18-25. New York, NY: ACM Press.

Pasquinelli, A. (2002). *Digital library technology trends* [White paper electronic version]. Retrieved July 28, 2005, from Sun Microsystems, Inc. website: http:// www.sun.com/products-n-solutions/edu/whitepapers/pdf/digital_library_trends .pdf

Paynter, G. (2000, June). hashfile.cpp. *Greenstone digital library* (Version 2.60). [Comment from computer software source code]. http://prdownloads .sourceforge.net/greenstone/gsdl-2.60-src.tar.gz?download.

Poole, E. (2005, June 1). 'I'm the guy they used to call Deep Throat': World's most famous anonymous source revealed. Calgary Herald, pp. A1,A3.

Quiqley, D. (2005, June 1). *Watergate tipster 'Deep Throat' revealed.* Calgary Herald, p. B12.

Quiqley, D. (2005, June 22). *Twisters and hailstorms pound flood-worn Alberta.* Calgary Herald, p. B8.

Rennie, J. (2002). *WordNet::QueryData* (Version 1.37) [Computer software]. http://search.cpan.org/dist/WordNet-QueryData/QueryData.pm.

Saggion, H., Lapalme, G. (2002). Generating indicative-informative summaries with sumUM [Electronic version]. *Computational linguistics*, 28(4), 497-526. Cambridge, MA: MIT Press.

Sakai, T., & Sparck-Jones, K. (2001). Generic summaries for indexing in information retrieval [Electronic version]. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 190-198. New Orleans, Louisiana: ACM Press.

Sang, E. F. T. K., & Sabine, B. (2000). Introduction to the CoNLL-2000 shared task: Chunking. *Proceedings of CoNLL-2000 and LLL-2000*, pp 127-132, Lisbon, Portugal.

Sekine, S. (2002a). *Manual of OAK system* (Version 0.1). Retrieved July 21, 2004 from http://www.cs.nyu.edu/~sekine/PROJECT/OAK/manual.ps.

Sekine, S. (2002b). *Proteus project OAK system english sentence analyzer* (Version 0.1) [Computer software]. Retrieved http://www.cs.nyu.edu/~sekine/ PROJECT/OAK.

Shehata, C. (2000). *Documentation workflow*. Retrieved April 15, 2005, from http://www.hodgemony.com/pages/HCI/manual_workflow.tps.pdf.

Stewart, S. (2003). Media training 101: A guide to meeting the press. John Wiley and Sons.

Straus, J. (n.d.a). *The blue book of grammar and punctuation*. Retrieved April 6, 2005, from http://www.grammarbook.com/punctuation/commas.asp.

Straus, J. (n.d.b). *The blue book of grammar and punctuation*. Retrieved April 6, 2005, from http://www.grammarbook.com/punctuation/semicolons.asp.

Tolle, K. M., & Chen, H. (2000). Comparing noun phrasing techniques for use with medical digital library tools [Electronic version]. *Journal of the American society for information science*. 51(4) (2000), 352-370. John Wiley & Sons, Inc.

Witten, I.H. (2003). Examples of practical digital libraries: collections built internationally using Greenstone [Electronic version]. *D-Lib magazine*. 9(3).

Witten, I.H., & Bainbridge, D. (2003) *How to build a digital library*. Morgan Kaufmann, San Francisco.

Witten I.H., Bainbridge, D., & Boddie, S.J. (2001). Greenstone: Open-source digital library software with end-user collection building [Electronic version]. *Online information review*, 288-298. Emerald Group Publishing Limited.

Witten, I. H., Bainbridge, D., Paynter G., & Boddie S. (2002a). Importing documents and metadata into digital libraries: Requirements analysis and an extensible architecture [Electronic version]. *Proceedings of the 6th European conference on research and advanced digital libraries*, 390-405. Rome, Italy: Springer-Verlag.

Witten, I. H., Bainbridge, D., Paynter G., & Boddie S. (2002b). The greenstone plugin architecture [Electronic version]. *Proceedings of the 2nd ACM/IEEE-CS joint conference on digital libraries*, 285-286. Portland, Oregon: ACM Press.

Witten, I. H., Boddie, S.J., & Thompson, J. (n.d.). *Greenstone digital library - user's guide.* Manual, University of Waikato, New Zealand. http:// prdownloads.sourceforge.net/greenstone/User-en.pdf.

Witten, I. H., Loots, M,. Trujillo, M., & Bainbridge, D. (2001). The promise of digital libraries in developing countries [Electronic version]. *Communications of the ACM.* 44(5). 82-85. New York, NY: ACM Press.

Witten, I. H., McNab, R. J., Boddie, S. J., & Bainbridge, D. (2000). Greenstone: a comprehensive open-source digital library software system [Electronic version]. *Proceedings of the fifth ACM conference on digital libraries*, 113-121. San Antonio, Texas: ACM Press.

Witten, I. H., Moffat, A., Bell, T. C. (1999). *Managing gigabytes.* San Francisco: Morgan Kaufmann.

Writing tips: Capitalization in titles. (1998). Retrieved April 6, 2005, from http:// www.writersblock.ca/tips/monthtip/tipmar98.htm.