New Jersey Institute of Technology

Digital Commons @ NJIT

Fall 10-31-1995

Automatic office document classification and information extraction

Xiaolong Hao
New Jersey Institute of Technology

Follow this and additional works at: https://digitalcommons.njit.edu/dissertations

Part of the Computer Sciences Commons

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

UMI Number: 9605744

UMI

300 North Zeeb Road
Ann Arbor, MI 48103

ABSTRACT

AUTOMATIC OFFICE DOCUMENT CLASSIFICATION AND INFORMATION EXTRACTION

by
Xiaolong Hao

TEXPROS (TEXt PROcessing System) is a document processing system
(DPS) to support and assist office workers in their daily work in dealing with
information and document management. In this thesis, document classification
and information extraction, which are two of the major functional capabilities in
TEXPROS, are investigated.

Based on the nature of its content, a document is divided into structured and
unstructured (i.e., of free text) parts. The conceptual and content structures are
introduced to capture the semantics of the structured and unstructured part of the
document respectively. The document is classified and information is extracted based
on the analyses of conceptual and content structures. In our approach, the layout
structure of a document is used to assist the analyses of the conceptual and content
structures of the document. By nested segmentation of a document, the layout
structure of the document is represented by an ordered labeled tree structure, called
Layout Structure Tree (L-S-Tree). Sample-based classification mechanism is adopted
in our approach for classifying the documents. A set of pre-classified documents are
stored in a document sample base in the form of sample trees. In the layout analysis,
an approximate tree matching is used to match the L-S-Tree of a document to be
classified against the sample trees. The layout similarities between the document
and the sample documents are evaluated based on the "edit distance" between the
L-S-Tree of the document and the sample trees. The document samples which have

the similar layout structure to the document are chosen to be used for the conceptual analysis of the document.

In the conceptual analysis of the document, based on the mapping between the document and document samples, which was found during the layout analysis, the conceptual similarities between the document and the sample documents are evaluated based on the degree of "conceptual closeness degree". The document sample which has the similar conceptual structure to the document is chosen to be used for extracting information. Extracting the information of the structured part of the document is based on the layout locations of key terms appearing in the document and string pattern matching. Based on the information extracted from the structured part of the document the type of the document is identified. In the content analysis of the document, the bottom-up and top-down analyses on the free text are combined to extract information from the unstructured part of the document. In the bottom-up analysis, the sentences of the free text are classified into those which are relevant or irrelevant to the extraction. The sentence classification is based on the semantical relationship between the phrases in the sentences and the attribute names in the corresponding content structure by consulting the thesaurus. Then the thematic roles of the phrases in each relevant sentence are identified based on the syntactic analysis and heuristic thematic analysis. In the top-down analysis, the appropriate content structure is identified based on the document type identified in the conceptual analysis. Then the information is extracted from the unstructured part of the document by evaluating the restrictions specified in the corresponding content structure based on the result of bottom-up analysis.

The information extracted from the structured and unstructured parts of the document are stored in the form of a frame like structure (frame instance) in the data base for information retrieval in TEXPROS.

AUTOMATIC OFFICE DOCUMENT CLASSIFICATION AND INFORMATION EXTRACTION

by
Xiaolong Hao

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Computer and Information Science

October 1995

APPROVAL PAGE

AUTOMATIC OFFICE DOCUMENT CLASSIFICATION AND INFORMATION EXTRACTION

Xiaolong Hao

Dr. Peter A. Ng, Dissertation Advisor   Date
Chairperson of Department of Computer and Information Science
Professor of Computer Science, NJIT

Dr. Jason T.L. Wang, Dissertation Co-advisor, Committee Member   Date
Assistant Professor of Computer Science, NJIT

Dr. H. T. Yeh, Committee Member   Date
Supervisor of FNMS Development, AT&T Bell Laboratories

Dr. James A. McHugh, Committee Member   Date
Director of Ph.D. Program in Computer Science
Professor of Computer Science, NJIT

Dr. Daochuan Hung, Committee Member   Date
Assistant Professor of Computer Science, NJIT

Dr. Machel Bieber, Committee Member   Date
Assistant Professor of Computer Science, NJIT

Dr. Edward Sarian, Committee Member   Date
Associate Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author: Xiaolong Hao

Degree: Doctor of Philosophy

Date: October 1995

Undergraduate and Graduate Education:

- Doctor of Philosophy,
  New Jersey Institute of Technology, Newark, NJ, U.S.A., 1995

- Master of Computer Science,
  Xian Jiaotong University, Xian, Shaanxi, P.R. China, 1989

- Bachelor of Computer Science,
  Xian Jiaotong University, Xian, Shaanxi, P.R. China, 1986

Major: Computer Science

Presentations and Publications:

1. X. Hao, J.T.L Wang, and P. A. Ng,, "Nested Segmentation: An Approach for Layout Analysis in Document Classification," *Proc. of Second IAPR Conference on Document Analysis and Recognition*, Tshkuba Science City, Japan, October. 1993.

2. X. Hao, J.T.L Wang, M. P. Bieber and P. A. Ng, "A Tool for Classifying Office Documents," *Proc. of the Fifth Int'l Conf. on Tools with Artificial Intelligence*, Boston, Massachusetts, USA, November. 1993.

3. X. Hao, J.T.L Wang, M. P. Bieber and P. A. Ng, "Heuristic Classification of Office Documents," *Int'l Journal on Artificial Intelligence Tools, Vol. 3, No. 2 1994.*

4. X. Hao, J.T.L Wang, M. P. Bieber and P. A. Ng, "A New Approach for Office Document Classifications," *Research Report, CIS-94-47, New Jersey Institute of Technology, 1994*

5. C. Wei, J.T.L. Wang, X. Hao, and P.A. Ng, "Inductive Learning and Knowledge Representation for Document Classification: The TEXPROS Approach" *Proc. of 3rd International Conference on Systems Integration*, Sao Paulo, SP, Brazil, August 1994.

This work is dedicated to my parents, my lovely wife and son.

ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his advisor, Professor Peter A. Ng, for his guidance, friendship, and moral support throughout this research.

Special thanks to Professor Jason T.L. Wang for being the co-advisor of this dissertation.

The author appreciates the consistent help from the colleagues of systems integration research group.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 TEXPROS

TEXPROS (TEXt PROcessing System) [47] is a personal, customized system for processing office documents. The system has functional capabilities of automating (or semi-automating) common office activities, such as document classification and filing, information extraction, browsing, synthesizing, reproduction, and retrieval. To accomplish these goals, the system includes the following components:

- A state-of-the-art data model capable of capturing the behavior of the various office activities[29, 46].

- A customized document classification handler that exploits both layout and textual analysis to identify the type of a document[13, 10, 11, 51, 44].

- Extracting a synopsis or the most significant information from a document[12].

- An agent-based architecture supporting document filing and file reorganization[50, 58].

- A retrieval system that can handle incomplete and vague queries[24, 22, 23].

This thesis presents the document classification and information extraction components of TEXPROS.

1.2 Document, Document Classification and Information Extraction

In an office environment, a very large amount of information is manipulated in the form of documents. A *document* consists of units of *text* (such as paragraphs, tables, figures, etc.) that can be interchanged between an originator and a recipient. *Text*

1

used in this thesis refers to a representation of any visual information for human perception that can be reproduced in two-dimensional form. Text and possibly additional control information constitute the *content* of a document [14, 26, 30]. A document can be interchanged as intended by the originator. It also can be interchanged in a *processable form*, which permits document editing and layout revision by the recipient.

One of the most important functionalities of TEXPROS provided by a retrieval system [22] is to allow users to browse, retrieve and synthesize information from the documents. In TEXPROS, both the original documents and the structured information extracted from the contents of the documents are stored in the document base. The structured information contains the synopsis of the content of the document and is called the *frame instance* of the document. The structure of the frame instance is described by the *frame template* associated with the document [29]. That is, we can view the frame instance as the instantiation of the frame template.

We give the formal definitions of frame template and frame instance as follows:

The TEXPROS document model uses the concepts of *type*, and *instance* to define the frame template and frame instance. The *primitive types* are **integer**, **real**, **string**, **text**, and **boolean**. An *enumeration type* is an ordered tuple of finite strings from $\mathcal{A}$, where $\mathcal{A}$ is an alphabet, that is, a finite set of symbols. The primitive and enumeration types are called *basic types*. An *attribute name* (or *attribute*) is a finite string of symbols. An attribute has a corresponding *type*.

Definition 1 *(Type)* Types are defined recursively as follows:

1. A basic type is a type.

2. Let A_i be an attribute with its corresponding type T_i, $1 \leq i \leq m$. $T = [\langle A_1 : T_1 \rangle,$ ..., $\langle A_m : T_m \rangle]$ is a type, called a *tuple type*. T_1, ..., and T_m are called the *underlying types* of T.

3. $T = \{T_1, ..., T_n\}$ is a type, called a *set type*. T_i, $1 \leq i \leq n$, is an underlying type of T. ☐

Definition 2 *(Instance)* Instances are defined recursively as follows:

1. An instance of a basic type is called a *basic instance*.

2. If A_1, ..., and A_m are distinct attributes of types T_1, ..., T_m and I_1, ..., and I_m are instances of T_1, ..., and T_m, then $I = [\langle A_1 : I_1 \rangle, ..., \langle A_m : I_m \rangle]$, $m \geq 1$, is an instance, called a *tuple instance*, of the type $[\langle A_1 : T_1 \rangle, ..., \langle A_m : T_m \rangle]$.

3. For $T = \{T_1, ..., T_n\}$, let I_i be an instance of an underlying type T_i. Then, a *set instance* I of the type T is a set of instances of the types T_i. ☐

Definition 3 *(Frame Template)* A frame template $\mathbf{F}$ is a tuple type $\mathbf{F} = [\langle A_1 : T_1 \rangle, ..., \langle A_m : T_m \rangle]$ - where A_i ($1 \leq i \leq m$) is an attribute over the attribute type T_i - which describes the structure of a document class in $\mathcal{O}$. ☐

Definition 4 *(Frame Instance)* A frame instance fi of a document $o \in \mathcal{O}$ is a tuple instance of a frame template $\mathbf{F}$, $fi = [\langle A_1 : I_1 \rangle, ..., \langle A_m : I_m \rangle]$, where $\mathbf{F} = [\langle A_1 : T_1 \rangle, ..., \langle A_m : T_m \rangle]$, A_i is an attribute, T_i is an attribute type and I_i is an instance of attribute type T_i extracted from the document o. ☐

For example, Figure 1.1 shows the frame template and its frame instance of a memorandum about CIS qualifying examination (QE memo).

In TEXPROS, the documents whose corresponding frame instances share the same frame template are grouped into classes. The documents which belong to the same class are said to be of the same *document type*. The concept of document type plays an important role in document processing systems such as TEXPROS [1, 7, 26]. By identifying the defined type of the documents, it is possible to implement

NJIT
New Jersey Institute of Technology   Ph.D Program Committee

MEMORANDUM

CONFIDENTIAL

TO: John Smith

FROM: Dr. Mike Thomas, Chairman
        Director of Ph.D Program in Computer Science

SUBJ: CIS Qualifying Examination

DATE: May 21, 1991

I would like to inform you that the CIS Qualifying Examination Committee has recommended to me that you conditionally pass the qualifying examination. However, upon the Committee's recommendation, you must take a written re-examination on Formal Language and Programming Language within a year.

In preparation for the partial re-examination on the above named areas, you are advised to repeat relevant courses in the topic area.

Cc: Members of the Ph.D Program Committee in Computer Science. Full Professors, Associate Chairs.

Receiver	Name	John Smith
	Title	
Sender	Name	Mike Thomas
	Title	Chairman, Director of Ph.D Program in Computer Science
Subject	CIS Qualify Examination	
Date	Year	1991
	Month	May
	Day	21
Cc	Members of the Ph.D Program Commitee in Computer Science. Full Professors, Associate Chairs.	
QE result	contionally pass	
Courses retaken	Formal Language and Programming Language	

Receiver	Name	String[10]
	Title	String[20]
Sender	Name	String[10]
	Title	String[20]
Subject	String[20]	
Date	Year	integer
	Month	String[10]
	Day	integer
Cc	Text	
QE result	Text	
Courses retaken	Text	

Figure 1.1 The frame template and its frame instance for a QE memo

efficient storage and access methods to enhance the performance of retrieval. Since the documents of the same type share the same frame template, each document type is associated with a frame template.

In TEXPROS, the task of *document classification* is identifying the document types of the office documents. That is, given an office document, document classification subsystem identifies the corresponding frame template of the document. The task of *information extraction* is extracting from the contents of the document the most relevant information pertinent to the user. That is, given an office document, the information extraction subsystem obtains the frame instance of the document by instantiating the corresponding frame template. The document classification and information extraction can be achieved in aid of analyzing the document structures.

1.3 Document Structures

The layout organization and the content of a document can be described by the document structures [33]. In our approach, the document structures include the layout, conceptual and content structures. The *layout structure* of a document is the description about where the text units of the document are positioned in the physical media such as paper or electronic media [14, 33]. A tree structure (L-S-Tree) is proposed [13] to represent the layout structure of a document in order to capture accurately the layout characteristics of the document. (The detail of the layout structure will be discussed in Chapter 3.)

The content of a document can be divided into *structured* and *unstructured* parts. The structured part specifies, more or less, the intentions of the document. Usually, the structured part of the documents of the same type share the similar layout organization and semantical functionality. The unstructured part refers to the major content of the document which are written in free-text. It usually plays the only role of specifying the intension of the document.
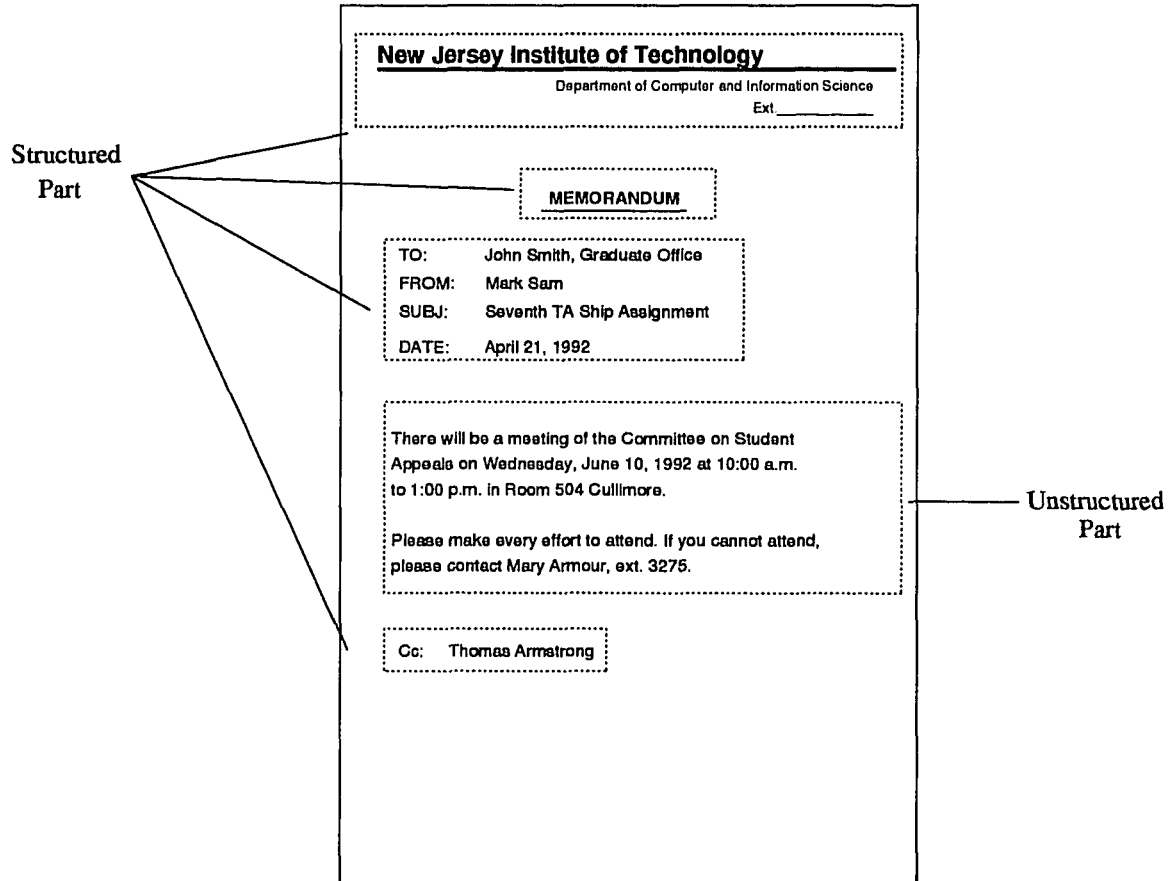
Figure 1.2 The illustrations of structured and unstructured parts of a memorandum

Figure 1.2, 1.3 and 1.4 illustrate the structured and unstructured parts of three different documents, i.e., a memorandum, a business letter and a research paper.

As mentioned in the previous section, each document type is defined by a frame template. The *conceptual structure* is introduced to facilitate the instantiation of the attributes of the frame template from the *structured part* of a document. The conceptual structure of the document is represented by a set of attributes descriptors which specify the properties of the attributes' values. In other words, the conceptual structure describes the semantical functionalities of the structured part of the document and it is used in document conceptual analysis for the structured part

Figure 1.3 The illustrations of structured and unstructured parts of a letter

Structured Part

Unstructured Part

Within the figure:

Telos: Representing Knowledge About Information System

JHON MYLOPOULOS, ALEX BORGIDA, MATTHIAS JARKE, and MANOLIS KOUBARAKIS
University of Toronto

We descibe Telos, a language intented to support the development of infomation systems, The design principles for the language are based on the premise that information system development is knowledge intensive and that the primary responsibility of any language intedted for the task is to be able to formally represent the relevant knowledge. Accordingly, the proposed language is founded on concepts from knowledge representation. Indeed, the language is approapriate for representing knowledge about a variety of worlds related to an information system, such as the subject world (applicaton domain), the usage world (user models, environments), the system world (software requirmentt, design), and the development world (terms, methodologies).

Categories and Subject Descriptors: D.2.1 [**Software Engineering**] Requirements/Specifications -language;

Gernerl Terms: Design, Languages

1. INTRODUCTION
Language facilities have been a key vehicle for advances in software productivity since the introduction of assembler in the early 1950s, the first high level programming language in the mid-1950s, and the languages supporting encapsulatioon/modu-

This work was supported by the University of Toronto; the National Science and Engineering Research Council of Canada; the Institute of Computer Science, Iraklion, Crete, Greece; and the Commission of European Communities through ESPRIT projects LOKI and DAIDA.

© 1990 ACM 1046-8188/90/0100-0325 $1.50

ACM Transactions on Information Systems, Vol. 8, No. 4, October 1990, Pages 325-362

Figure 1.4 The illustrations of structured and unstructured parts of an ACM Transaction Paper

of the document. (The detail of the conceptual structure will be discussed in Chapter 4.)

The *content structure* is introduced to facilitate the instantiation of the attributes of the frame template from the *unstructured part* of a document. The content structure of the document is represented by an activation condition and a set of attribute descriptors. The activation condition specifies under what condition the content structure is used as the knowledge to extract information from the unstructured part of the document. And each attribute descriptor specifies the properties of the attributes' values. The content structure describes the semantical functionalities of the unstructured part of the document and it is used in document content analysis for the unstructured part of the document. (The detail of content structure will be discussed in Chapter 5.)

1.4 Organization of the Document Classification and Information Extraction Component

The overall organization of the proposed document classification and information extraction subsystem is shown in Figure 1.5. The system is composed of two components: the document processing component and system customization component. The processing component classify the incoming document and extracts information from the document. The customization component acquires the knowledge needed for the document classification and the information extraction from the user. These knowledge include document samples, conceptual structure, content structure, etc..

A system control diagram is shown in Figure 1.6 to illustrate the relationship between these two components.

The office documents to be processed are first digitized and thresholded into binary images by a scanner or a facsimile. Then the document is encoded through the recognition system. After the recognition process, the content of its textual part
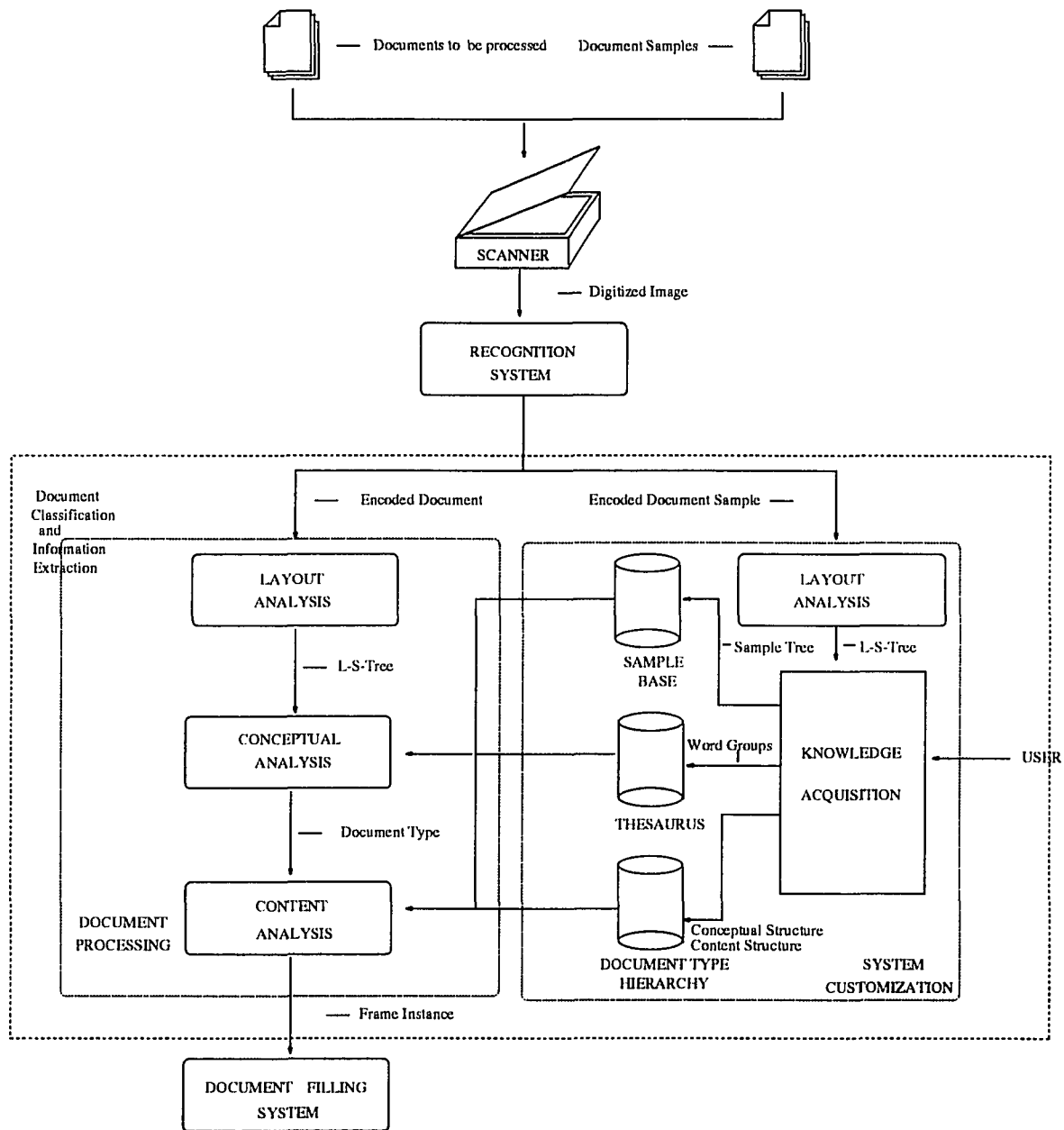
Figure 1.5 Organization of a document classification and information extraction system
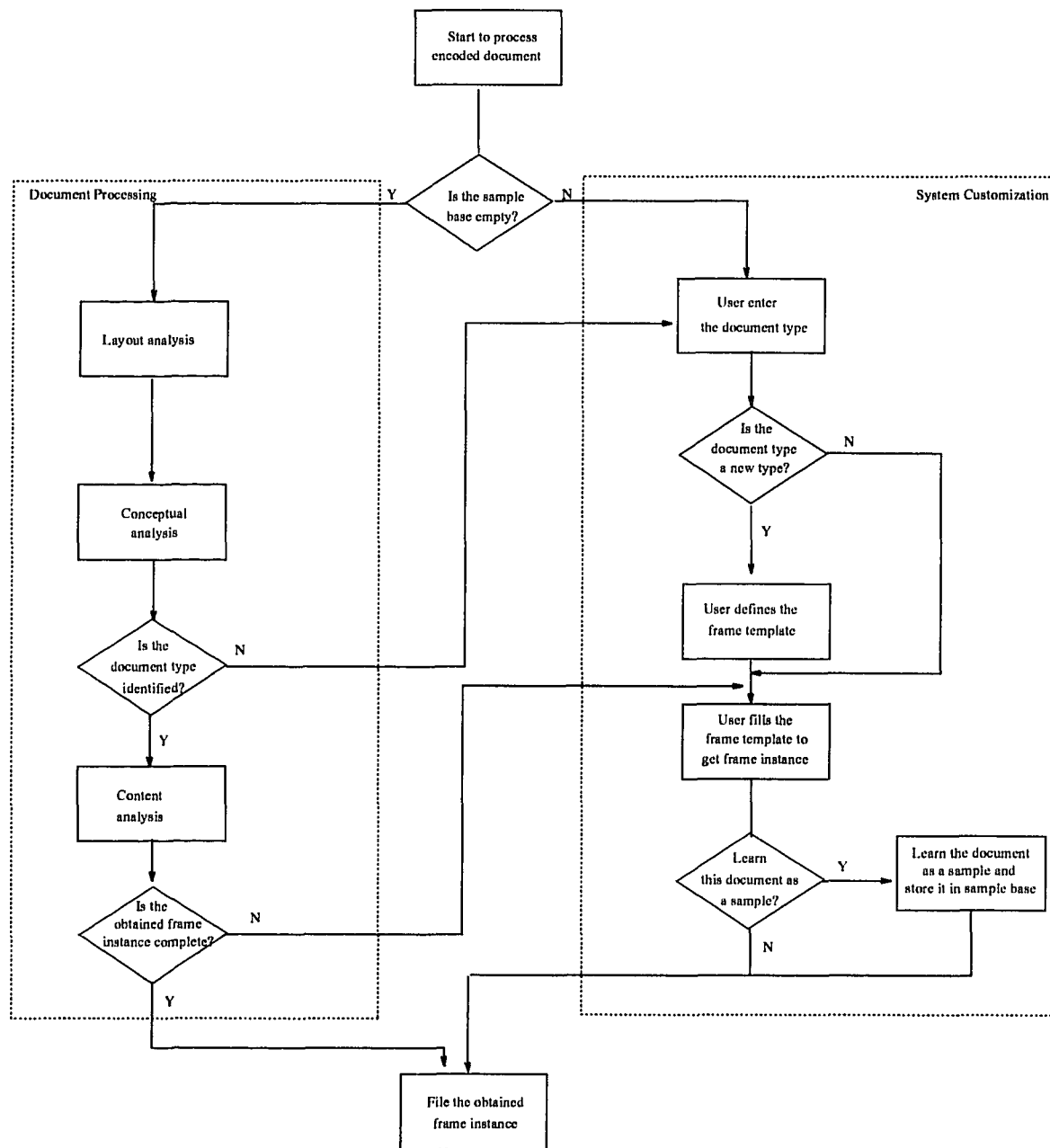
Figure 1.6 System control diagram

is recognized and the description of the non-textual part of the document such as logos, figures, and pictures, is extracted.

The document classification and information extraction system begins with the layout analysis process. During the process, the layout structure of a document is obtained in the form of nested segmentation of the document which is represented by a tree structure called the *Layout Structure Tree (L-S-Tree)*.

In the conceptual analysis, the conceptual structure of a document is identified by finding a document sample with the same conceptual functionalities. The layout structure of the document is used to facilitate searching such a document sample. Firstly, the layout similarities between the document and the document samples pre-stored in the document sample base of the system are determined by the approximate tree matching technique. Secondly, conceptual similarities between the document and document samples are determined by evaluating the "conceptual closeness degree". Based on the identified conceptual structure, part of the frame instance is obtained by extracting information from the structured part of the document.

In the very first time of executing the system, the sample base is empty and the system is not able to process automatically the document. The user will enter the document type and frame instance for this document through the system customization. And also, this document can be learned as a document sample which is used later to facilitate the automatic classification and information extraction of other documents of the same document type. The sample base grows as more documents of different document types, or of the same document type but with different layout are processed by the system.

In the content analysis, the appropriate content structure of a document is chosen based on the information extracted from the structured part of the document. Thus, the document type is identified based on its layout, conceptual and content

structures. The remaining part of the frame instance is obtained using heuristic free text analysis including the sentence classification and thematic analysis.

1.5 Overview of the Thesis

Chapter 2 presents the survey of related work on document analyses, document classification and information extraction. Chapter 3 discusses the layout analysis including the algorithm of nested segmentation and L-S-Tree construction. Chapter 4 discusses the conceptual analysis including the definition of conceptual structure and sample-based document conceptual analysis. The procedure of the information extraction from the structured part of the documents is also given in Chapter 4. The detail of content analysis is discussed in Chapter 5. It covers the definition of content structure, the identification of the document type based on the information extracted from the structured part of the documents, and the procedure of the information extraction from the unstructured part of the documents. Chapter 6 summarizes the thesis.

CHAPTER 2

RELATED WORK

2.1 Document Classification

Research on automatic classification of the documents began before 1960, in direct response to the needs for handling large-scale and complex data by computers in a fast and consistent manner [17]. In the 60's and 70's, many research work of automatic classification of the documents focused on the term statistics. The documents are classified as their type by checking the statistics on the frequency of some key-terms in the documents [41, 16, 18].

In the 80's and 90's, due to the progress of image processing and pattern recognition, the layout analysis of the document plays a significant role in the document processing. The organization of a document was described in two folds: one is the conceptual structure which is content oriented, and the other is syntactical structure including layout and logical structures [33, 1]. Many researchers began to study how to analyze documents based on these structures.

2.1.1 Document Structure Analysis

The document structure analyses can be divided into two categories: the document layout analysis and the document conceptual analysis.

2.1.1.1 Document Layout Analysis Most research work [6, 27, 8, 32, 45, 43, 31] focused on detecting the similarities of the layout structure of the documents without taking content analysis into consideration. ANASTASIL (A Hybrid Knowledge-based System for Document Layout Analysis) [6, 27] is a system for analyzing single-sided business letters. It is a knowledge-base system for identifying the different regions of a document image such "receiver", "subject", "date", etc. in the business letter.

14

A letter is divided into segments and a search tree called geometric tree is used to represent all possible segmentations for the business letters. The regions of a letter are identified by searching an appropriate segmentation in the geometric tree using best-first search. In [32], block matrix and rules are used to represent relative position of layout objects in a document. In [8], a pattern-oriented segmentation method is used to allow document images of tabular form to be analyzed during the process of document structure analysis. In [45], a text reading system is introduced for analyzing newspaper. The system consists of three major components, namely, the document analysis, the document understanding, and the character segmentation and recognition. The document analysis component extracts lines of text from a page for recognition. The document understanding component extracts the logical relationship among the layout objects such as the association between the the topic and paragraphs of a article in the newspaper. The character segmentation and recognition component extracts and recognizes characters from a text line. In [43], the result of the layout analysis is used to deduce the conceptual structure of a document based on the direct connection between the layout structure and the conceptual structure of the document. In [31], a goal-directed top-down approach employs a three-level rule hierarchy to interpret and classify the information of the document image. The system was applied in the domain of the postal mail-pieces. Because these works only deal with a restricted type of documents such as electronic mails [7, 25], business letters [6], or form documents [8] which have inherently fixed layout structures, the simple segmentation technique for identifying their layout structures is sufficient. In contrast, in an office environment, documents of various types are used. These documents, such as memos, technical reports, and research papers, often have complex layout structures and contents. The one level segmentation technique is usually not accurate enough to reflect the layout structure of the documents, as most of the documents usually use more than one spacing scale for separating their layout objects. In this

thesis, a nested segmentation technique is proposed to capture the layout structures of the documents accurately based on the different line spacing scales used in the documents.

2.1.1.2 Document Conceptual Analysis In [25, 7, 54, 3], the conceptual structure of the document provides information that assists document classification in the system. Without involving rigorous layout analysis, most of these approaches mentioned above used the keyword search to find the relationship between the layout objects (such as, "block", "paragraph" ,etc.) in the document and the semantic objects (such as, "receiver", "sender", etc.) in the conceptual structure. In [25, 7], a knowledge based document classification system is designed to support integrated document handling. It provides two functional capabilities, namely, the conceptual and content descriptions. The conceptual description describes the conceptual structure of a document type in terms of a tree structure. The content description describes the relationship between the semantic objects and the keywords in the original document. Given documents, their types are determined based on the predefined description of the document types. In [54], similar to the approach, expert system techniques are used. Instead of using tree structure as in [25, 7, 54], a semantic network representation is used in [3] to describe the conceptual structure of a document. Due to the lack of the context analysis, the word-based techniques usually simply recognize phrases or keywords. However, we observe that the keywords in an office document play certain conceptual roles only if they appear in certain places in the document, i.e., they must follow a certain layout structure in order to play those roles[1]. Thus, the word-based techniques have difficulties in resolving the ambiguities if the same keyword appears several times in different places in the document.

---

[1]For example, the keyword "To" appearing in the top left corner of a memo indicates that the phrases following it are the "receiver" of the memo, whereas in the content (e.g., in a phrase such as "To my knowledge"), it may have different meanings.

2.1.2 Mechanism of the Document Classification

In this subsection, we shall investigate two mechanisms for classifying documents. They are the *generalization-based* approach, and the *sample-based (example-based)* approach [34].

The generalization-based approach requires a strong domain theory to summarize the cases of classifying documents at the training phase in terms of concept descriptions and to classify new documents using these descriptions. In [7, 25], this approach is employed to create and use definition of document types for classifying the documents. The Conceptual Structure Definition (CSD) and the Content Description Language (CDL) are used to define document types. The CSD specifies the conceptual structure of a document and CDL specifies the relationships between the semantic objects defined in CSD and the layout objects in the document. The document type of a given document is determined by testing whether it complies with the predefined conceptual structure of a document type. In this approach, in order to classify a document of a new type, the user has to analyze thoroughly many documents of this type for generalizing the relationships between the semantic objects in the conceptual structure and the layout objects in the documents of the type. But there lacks a domain theory to support such generalization in the office document domain. This type of domain is called *the domain with weak theory* [34]. In addition, this generalization-based approach assumes that the documents of the same type have the similar layout structure. The assumption is not true for the domain of office documentation. The documents of the same type may have different layout structures. For example, in Figure 2.1, these are two journal papers which usually are classified as the documents of the same type (journal papers); one of them is a paper from IEEE Transactions and the other is a paper from ACM Transactions. Their layouts are obviously different.

A Comparison of Measures of Control
Flow Complexity

Telos: Representing Knowledge About
Information Systems

Figure 2.1 Two examples of journal papers

For documents of the same type having different layout, the generalization-based approach could classify them as different types of documents. For example, these IEEE and ACM Transactions papers could be treated as two different document types such as "IEEE Journal paper" and "ACM Journal paper". However, this would require the user to do extraneous work in defining the document types. Moreover, from the user's viewpoint, this classification does not make any sense because he/she may not care if the paper is an IEEE paper or an ACM paper.

In contrast, the sample-based approach does not has this problem as discussed above. In the sample-based classification, instead of asking the user to generalize the relationships between the conceptual and layout structures of a document type, a set of document samples of the same type are stored in an appropriate way so that a document can be classified with certainty if it belongs to a type of the pre-stored samples. This approach attempts to achieve reliability and efficiency of document classification by maximizing the use of direct match between a sample and a document [34]. A sample base is created by acquiring all samples of various document types from the user. A document is classified by comparing its layout and conceptual features against the samples in the sample base.

2.2   Information Extraction

There are two basic approaches for extracting information from the text. One is word-based approach which examines the key words appearing in the documents. Automatic indexing, which is a widely used word-based approach, includes dictionary look-up, stop-wording, word stemming, and term-phrase formation [42]. This approach was adopted by several document processing systems [7, 3]. The other one is to employ natural language understanding (NLP) technique to analyze the text. Examples of this approach are : BORIS and IPP, which obtain a summary of the text based on the understanding of the text, and SCISOR and NLPDA, which extract

information related to the topic of text by incorporating the syntax and semantic analyses of the text based on the domain knowledge.

"BORIS" is a narrative understanding system [20, 21]. It attempts to understand what it reads, to as great a depth as possible. It consists of a conceptual analyzer, an event assimilator, a question answering module and a English generator. The conceptual analyzer accepts English sentences as input and then constructs the "Conceptual Dependency Structures." The event assimilator contains top-down expectations about the events would occur next, and uses this information for filling in missing role bindings in the conceptual structures. The question answering module interprets questions and searches for conceptual answers. Finally, the English generator produces English expressions as output. The system focused on the complex stories involving divorce. There have been a few on-going researches investigating this approach [39, 38, 37, 40].

The "Integrated Partial Parser (IPP)" is another example of this approach for understanding natural language text. In [19], the IPP reads news stories, generalizes them and understands the new stories based on the generalization of the stories it remembered. The IPP focused on the stories about international terrorism taken from local newspapers and the UPI news wire.

"SCISOR" [15, 36] is a system extracting information from the on-line news. The SCISOR employs lexical analysis, separation of raw news into story structure, topic determination of story and natural language analysis using an integration of two interpretation strategies — "bottom-up" linguistic analysis and "top-down" conceptual interpretation. The system focused on the financial news, especially on the stories about corporate mergers.

NLPDA [2] is another system of this approach. NLPDA is used to extract the information from Patient Discharge Summaries (PDSs) written by physicians. Like SCISOR, the linguistic and detailed world knowledge are provided to the system. The

system intended to extract the explicit information as well as implicit information from PDSs. The prototype of the system was tested in a restricted medical domain: thyroid cancer care. There have been other researches on the information extraction [35] based on linguistic analysis and detailed world knowledge. So far, these researches focus on the information extraction from the free text in a very restricted domain. The text is related to a specified domain.

The information extraction of office documents is not quite the same as information extraction from the free text. The content of document is divided into structured and unstructured parts. The unstructured part of the document is referred to as the body of the document and, therefore, is of free text. It is usually domain related. In contrast, the structured part is referred to as the header of the document, and is document type related. For example, the conceptual components of a memo such as "sender", "receiver", etc. are related to "memo" type, and are denoted by the keywords such as "TO", "FROM" and their layout locations.

There are several efforts in conducting information extraction of office document [7, 3]. These systems do not take layout analysis into consideration. They only focus on the extraction from the structured part of the document. They usually use keyword searches and statistical techniques. As mentioned in the previous section, keywords alone cannot distinguish the relevant from the irrelevant text without considering their layout structure of the document. Thus, these systems can only handle the documents with relatively fixed layout structures. Our approach attempts to combine the result of layout analysis, keyword matching and natural language analysis to achieve the goal of information extraction of documents.

CHAPTER 3

DOCUMENT LAYOUT ANALYSIS

As we mentioned in Chapter 1, the layout, conceptual and content structures constitute the document structures. The layout analysis of the document helps the processes of document classification and information extraction. The method of analyzing the layout structure of a document proposed in this thesis is called *nested segmentation*, which divides the document into rectangular areas, called *segments*. Then the segmented document is represented by a tree structure (L-S-Tree). The Figure 3.1 shows the organization of the layout analysis procedure.

3.1 Concepts in Document Segmentation

In this section, the concepts of block and nested segmentation will be introduced.

3.1.1 Block

An encoded document obtained from the recognition process is represented in terms of a set of blocks.

A *block* is defined as a minimum rectangular portion of the document which is either a *textual block* or a *non-textual block* [44]. The textual block is associated with a set of text lines having the same typeface, which includes the font type and the font size, and consistent line spacing. The non-textual block is dealing with a figure, logo, picture, and so forth. Formally, each block is represented by a quintuple (*Id, Type, Content, Location, Size*), where

- *Id* is the unique identifier for the block;

- *Type* indicates whether the block is textual or non-textual;

22

```
            |  —— Encoded Document
            ↓
  ┌─────────────────────────┐
  │   Nested Segmentation   │
  └─────────────────────────┘
            │
            ↓  —— Nested Segmented Document

  ┌─────────────────────────┐
  │   L-S-Tree Construction │
  └─────────────────────────┘
            │
            |  —— L-S-Tree
            ↓
```
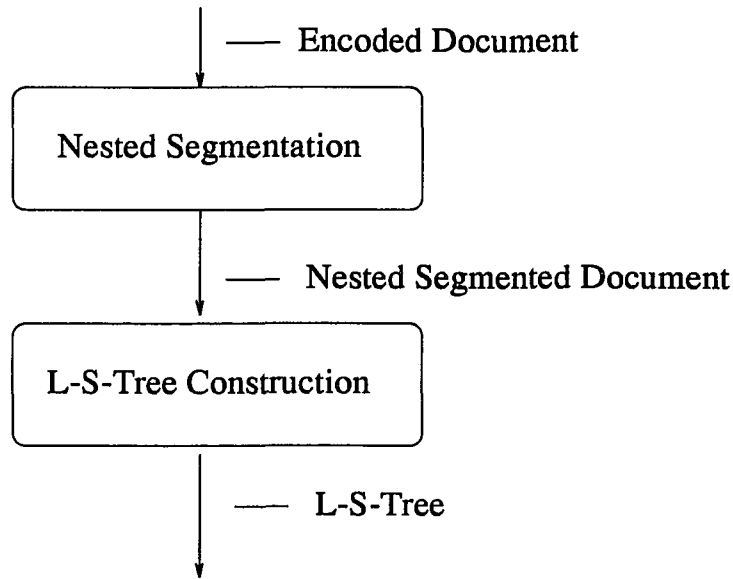
Figure 3.1 The Organization of Layout Analysis Procedure

- *Content* is the recognized text of a textual block or the description of a non-textual block;

- *Location*(x, y) specifies the location of the block in the document with respect to the upper-left corner of the document page, where x and y stand for the horizontal and vertical coordinates, respectively;

- *Size*(dx, dy) is the size of the block, where dx and dy stand for the width and the height of the block, respectively.

The *Size* of the block is measured in terms of the number of pixels, and the *Location* of the block is measured in terms of the coordinates of the underlying pixels.

Figure 3.2 shows the representation of a block, and Figure 3.3 shows the set of blocks after applying the recognition process to the memo in Figure 1.1.
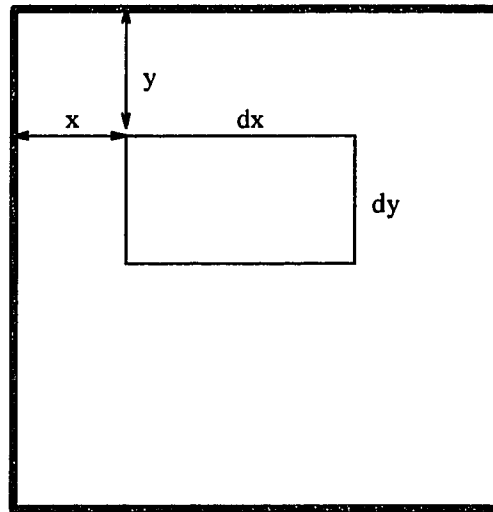
Figure 3.2 The representation of a document block

B_ID	B_Type	B_Content
1	LOGO	NJIT Logo
2	Textual	Ph.D Program Committee
3	Textual	MEMORANDUM
4	Textual	CONFIDENTIAL
5	Textual	TO:
6	Textual	John Smith
7	Textual	FROM:
8	Textual	Dr. Mike Thomas, Chairman Director of Ph.D Program in Computer Science
9	Textual	SUBJ:
10	Textual	CIS Qualifying Examination
11	Textual	DATE:
12	Textual	May 21, 1991
13	Textual	I would like to inform you that the CIS Qualifying Examination Committee has recommended to me that you conditionally pass the qualifying examination. However, upon the Committee's recommendation, you must take a written re-examination on Formal Language and Programming Language within a year.
14	Textual	In preparation for the partial re-examination on the above named areas, you are advised to repeat relevant courses in the topic area.
15	Textual	Cc:
16	Textual	Members of the Ph.D Program Committee in Computer Science. Full Professors, Associate Chairs.

Figure 3.3 Blocks of the document in Figure 1.1

3.1.2 Nested Segmentation

A common approach of recognizing the layout structure of a document is *segmentation*, which divides the document into rectangular areas, called segments. For example, in [6, 7, 8, 25, 27], the technique of one level segmentation is used. In [6, 27] the document is divided into several segments; each of the segments is assigned with a semantic meaning and is associated with a semantic object such as title, subject, date, etc. Because these works only deal with a restricted type of documents such as electronic mails [7, 25], business letters [6], or form documents [8] which have inherently fixed layout structures, the simple segmentation technique for identifying their layout structures is sufficient. In contrast, in an office environment, documents of various types are used. These documents, such as memos, technical reports, and research papers, often have complex layout structures and contents. Usually, the one level segmentation technique is not accurate enough to reflect the layout structure of the documents, as most of the documents usually use more than one spacing scale for separating their layout objects.

Consider the sample of a memorandum shown in Figure 3.4. The document uses more than one spacing scale in a nested manner. In fact, almost all types of documents use more than one spacing scale. The variations of spacing scale used between the layout objects of a document reflect the recognition that layout objects which lie close together tend to have semantically related contents. Based on these observations, we introduce a nested segmentation procedure to obtain the accurate layout structure of office documents.

In the *nested segmentation*, a document page is divided into *segments*. Each segment is a rectangular portion of the document which contains at least one block. A segment itself can be further divided horizontally or vertically into smaller segments. Therefore, there are two types of segments. One is the *basic segment*, which contains only one textual (or non-textual) block and cannot be further divided into smaller

NJIT ◄─────────► Ph.D Program Committee

New Jersey Institute of Technology

MEMORANDUM

CONFIDENTIAL

TO: ◄─► John Smith

FROM: ◄► Dr. Mike Thomas, Chairman
        Director of Ph.D Program in Computer Science

SUBJ: ◄─►CIS Qualifying Examination

DATE: ◄► May 21, 1991

I would like to inform you that the CIS Qualifying Examination
Committee has recommended to me that you conditionally
pass the qualifying examination. However, upon the Comm-
ittee's recommendation, you must take a written re-examina-
tion on Formal Language and Programming Language within
a year.

In preparation for the partial re-examination on the above
named areas, you are advised to repeat relevant courses in
the topic area.

Cc: ◄─►Members of the Ph.D Program Committee in Computer
        Science. Full Professors, Associate Chairs.

Figure 3.4 A memo with different line spacings

segments. The second is the *composite segment*, which can be divided vertically or horizontally into smaller segments.

In general, the process of the nested segmentation is as follows: a document page is first divided into segments which are at level 1. All the composite segments at level i are divided into several smaller segments at level i+1, each of which is assigned with an identifier (a number). The segmentation process terminates when all composite segments cannot be further divided.

In the nested segmentation, a segment can be represented by a quadruple (*Id, Type, Orientation, Composition*), where

- *Id* is the identifier for the segment;

- *Type* indicates whether the segment is basic or composite;

- *Orientation* specifies if the composite segment is divided horizontally or vertically; and it has no value if the segment is basic;

- *Composition* is represented by the identifier of the block contained in the segment if the segment is basic. If the segment is composite, the *Composition* specifies the identifiers of the segments contained in this segment. Suppose that the identifiers of these segments are $S_1, S_2, ...,$ and S_n, then the *Composition* is represented by $(S_1, S_2, ..., S_n)$, and the order is from top to bottom within the segment if the segment is divided horizontally, or from left to right if the segment is divided vertically.

3.2 Algorithm for Nested Segmentation

We now describe the algorithm used in the nested segmentation procedure. The input of this procedure is an encoded document, i.e., a document composed of a collection of rectangular blocks. For simplicity, a single page document is considered here.
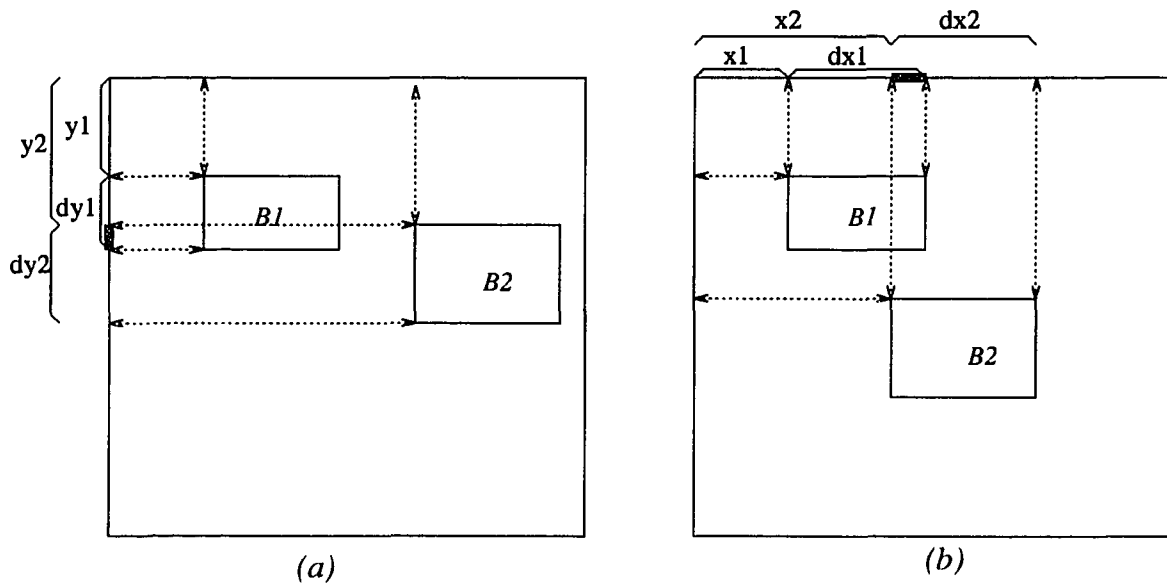
Figure 3.5 Examples of V-overlapping blocks and H-overlapping blocks

Some definitions are introduced first. Let B_is $(1 \leq i \leq 3)$ be the distinct blocks in a document, where (x_i, y_i) and (dx_i, dy_i) specify the location and size of a B_i, respectively.

Definition 5 *(V-overlapping and H-overlapping)*

Two blocks B_1 and B_2 *overlap vertically* (in abbreviation, *V-overlapping*), denoted as $B_1 \|_v B_2$, if $y_1 \leq y_2 \leq y_1 + dy_1$ or $y_2 \leq y_1 \leq y_2 + dy_2$.

Likewise, two blocks B_1 and B_2 *overlap horizontally* (in abbreviation, *H-overlapping*), denoted as $B_1 =_h B_2$, if $x_1 \leq x_2 \leq x_1 + dx_1$ or $x_2 \leq x_1 \leq x_2 + dx_2$.

Figure 3.5(a) shows two V-overlapping blocks and Figure 3.5(b) shows two H-overlapping blocks.

We project the blocks onto horizontal axis and vertical axis. Intuitively, two blocks are V-overlapping if their vertical projections overlap, and H-overlapping if their horizontal projections overlap.
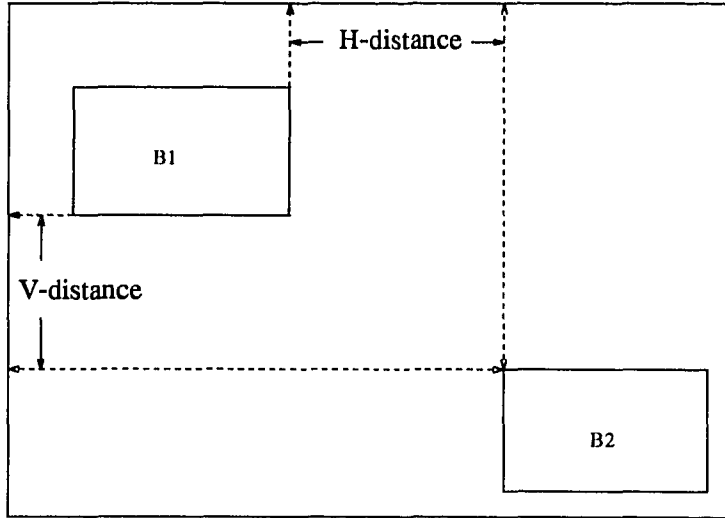
Figure 3.6 H-distance and V-distance

Definition 6 *(H-distance and V-distance of the blocks)*

We define the H-distance between two blocks B_1 and B_2, denoted as $H\text{-}distance(B_1, B_2)$, as:

$$H\text{-}distance(B_1, B_2) = \begin{cases} 0 & \text{if } B_1 =_h B_2 \\ x_2 - x_1 - dx_1 & \text{if } \neg(B_1 =_h B_2) \wedge (x_2 > x_1) \\ x_1 - x_2 - dx_2 & \text{if } \neg(B_1 =_h B_2) \wedge (x_1 > x_2) \end{cases}$$

Likewise, the V-distance between two blocks B_1 and B_2, denoted as $V\text{-}distance(B_1, B_2)$, is defined as:

$$V\text{-}distance(B_1, B_2) = \begin{cases} 0 & \text{if } B_1 \|_v B_2 \\ y_2 - y_1 - dy_1 & \text{if } \neg(B_1 \|_v B_2) \wedge (y_2 > y_1) \\ y_1 - y_2 - dy_2 & \text{if } \neg(B_1 \|_v B_2) \wedge (y_1 > y_2) \end{cases}$$

Figure 3.6 shows the H-distance and V-distance between two blocks.

Definition 7 *(H-adjacency)*
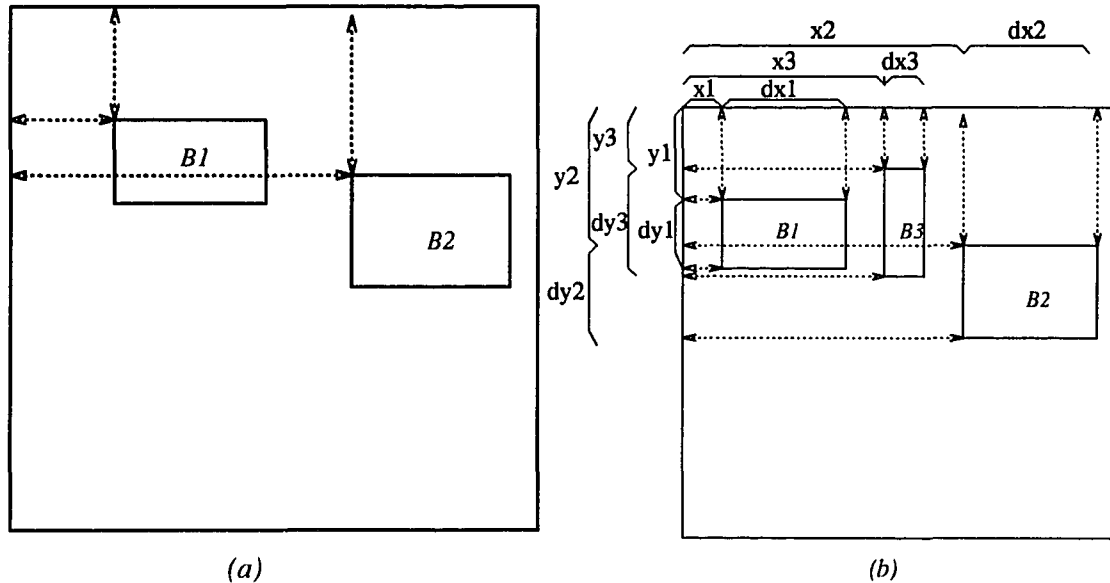
Figure 3.7 H-adjacent blocks

Two blocks B_1 and B_2 are *adjacent horizontally* (in abbreviation, *H-adjacent*), denoted as $B_1 \sim_h B_2$, if:

- $B_1 \|_v B_2$, and

- there are no other blocks, say B_3, satisfying $x_1 < x_3 < x_2$ or $x_2 < x_3 < x_1$, such that $B_1 \|_v B_3$ and $B_3 \|_v B_2$.

In Figure 3.7(a), B_1 and B_2 are two H-adjacent blocks, but in Figure 3.7(b), B_1 and B_2 are not H-adjacent because of the presence of B_3. In Figure 3.7(b), B_1 and B_3 are H-adjacent, so are B_3 and B_2.

Definition 8 *(V-adjacency)*

Two blocks B_1 and B_2 are *adjacent vertically* (in abbreviation, *V-adjacent*), denoted as $B_1 \sim_v B_2$, if:

- $B_1 =_h B_2$, and
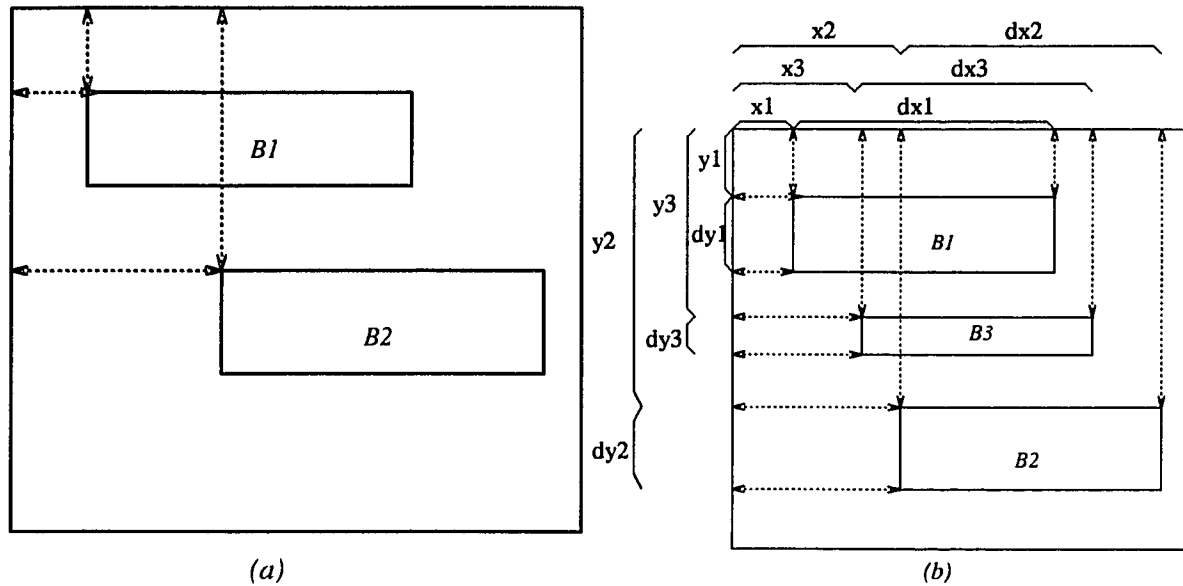
Figure 3.8 V-adjacent blocks

- there are no other blocks, say B_3, satisfying $y_1 < y_3 < y_2$ or $y_2 < y_3 < y_1$, such that $B_1 =_h B_3$ and $B_3 =_h B_2$.

In Figure 3.8(a), B_1 and B_2 are two V-adjacent blocks, but in Figure 3.8(b), B_1 and B_2 are not V-adjacent because of the presence of B_3. In Figure 3.8(b), B_1 and B_3 are two V-adjacent blocks, so are B_3 and B_2.

Definition 9 *(D-adjacency)* Without loss of generality, let $x_1 \geq x_2$ and $y_1 \geq y_2$. Two blocks B_1 and B_2 are *adjacent diagonally* (in abbreviation, *D-adjacent*), denoted as $B_1 \sim_d B_2$, if:

- neither $B_1 \|_v B_2$ nor $B_1 =_h B_2$, and

- there are no other block, say B_3, which overlaps the area where its four corners' coordinates are $(x_1 + dx_1, y_1 + dy_1)$, $(x_1 + dx_1, y_2)$, $(x_2, y_1 + dy_1)$, (x_2, y_2).

In Figure 3.9(a), B_1 and B_2 are two D-adjacent blocks, whereas in Figure 3.9(b), B_1 and B_2 are not D-adjacent because of the presence of B_3.
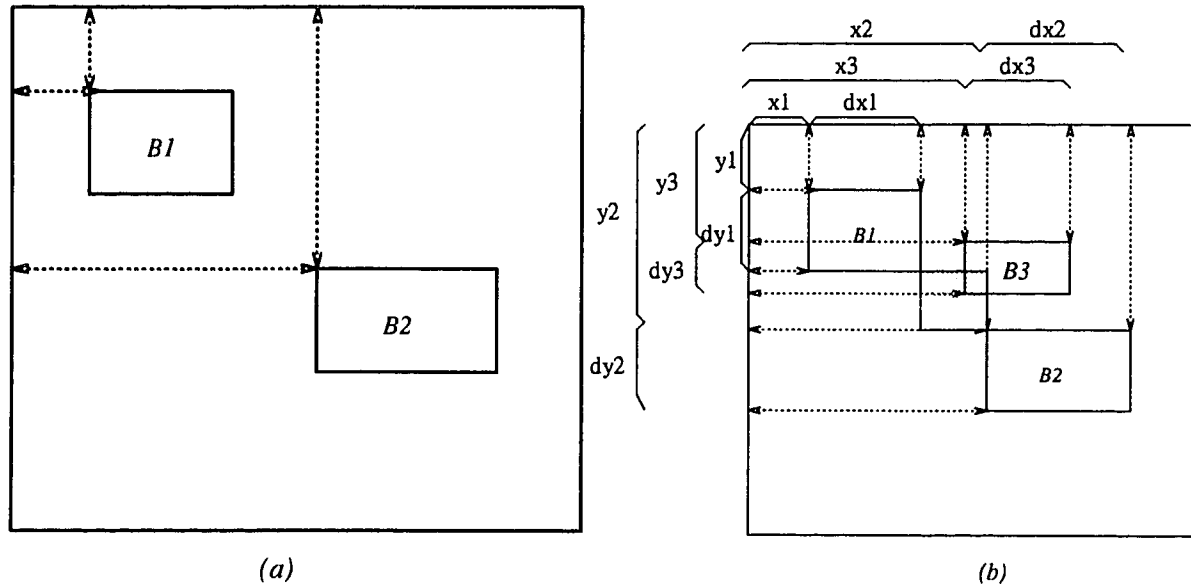
Figure 3.9 D-adjacent blocks

Definition 10 *(Adjacent Block Graph)*

An *adjacent block graph* $G(N, E, W)$ for a given document D is a weighted undirected graph, where

1. each node in N corresponds to one of the document blocks;

2. each edge $e = (B_1, B_2)$ in E is one of the following:

   - *H-edge* if $B_1 \sim_h B_2$;

   - *V-edge* if $B_1 \sim_v B_2$;

   - *D-edge* if $B_1 \sim_d B_2$.

3. In W, the *weight* of an edge $e = (B_1, B_2)$ is defined as $(H\text{-}weight(e), V\text{-}weight(e))$, where $H\text{-}weight(e)$ is the H-distance between B_1 and B_2, and $V\text{-}weight(e)$ is the V-distance between B_1 and B_2.

Note that, the weight of an H-edge is $(H\text{-}distance, 0)$; the weight of a V-edge is $(0, V\text{-}distance)$; and the weight of a D-edge is $(H\text{-}distance, V\text{-}distance)$. We call an adjacent block graph *trivial* if it has only one node.

Figure 3.10 shows the corresponding adjacent block graph of the memo in Figure 3.3.

Definition 11 *(Minimal Cut in Adjacent Block Graph)*

We define a cut-set of an adjacent block graph G to be a set of edges whose removal disconnects G. For our purpose, H-edge and V-edge are not allowed to be in the same cut-set. However, H-edge and D-edge, or V-edge and D-edge, can be in the same cut-set. A *minimal cut* of G, denoted as MC_G, is a cut-set which does not properly contain any other cut-set.

The *weight* of a minimal cut MC_G, denoted as $weight(MC_G)$, is defined by either the $H\text{-}weight$ or the $V\text{-}weight$ of a edge in the minimal cut MC_G.

- $weight(MC_G) = H\text{-}weight(e)$ where e is a edge in the minimal cut MC_G if:

  - MC_G contains at least one H-edge and $\forall e' \in MC_G$, $H\text{-}weight(e) \leq H\text{-}weight(e')$, or

  - all edges in MC_G are D-edges and $\forall e' \in MC_G$, $H\text{-}weight(e) \leq min\{H\text{-}weight(e'), V\text{-}weight(e')\}$.

- $weight(MC_G) = V\text{-}weight(e)$ where e is a edge in the minimal cut MC_G if:

  - MC_G contains at least one V-edge and $\forall e' \in MC_G$, $V\text{-}weight(e) \leq V\text{-}weight(e')$, or

  - all edges in MC_G are D-edges and $\forall e' \in MC_G$, $V\text{-}weight(e) \leq min\{H\text{-}weight(e'), V\text{-}weight(e')\}$.
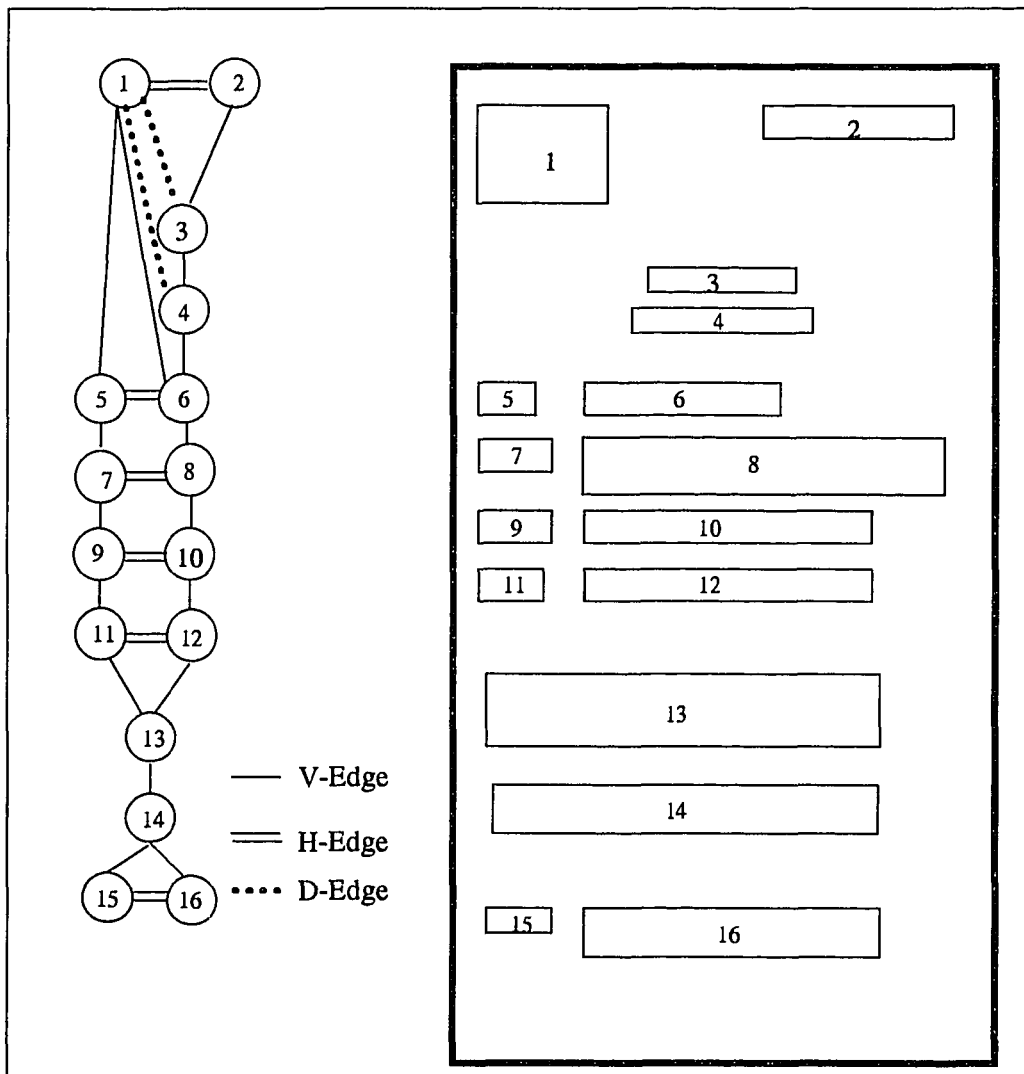
Figure 3.10 Adjacent block graph of the memo

The *type* of the minimal cut MC_G, denoted as $type(MC_G)$, is defined as follows:

$$type(MC_G) = \begin{cases} \text{H-type} & \text{if } weight(MC_G) = H\text{-}weight(e), e \in MC_G \\ \\ \text{V-type} & \text{if } weight(MC_G) = V\text{-}weight(e), e \in MC_G \end{cases}$$

Note that if there is a tie between $H\text{-}weight(e_1)$ and $V\text{-}weight(e_2)$ for the weight of MC_G which contains only D-edges, the $H\text{-}weight(e_1)$ is selected as the $weight(MC_G)$.

Definition 12 *(Path and Cycle)* Given a graph G, we define a *path* from a node u in G to a node v in G, denoted as $u \leftrightarrow v$, as an alternating sequence of nodes and edges,

$$n_1, e_1, n_2, e_2, ..., n_{k-1}, e_{k-1}, n_k,$$

where $n_1 = u$, $n_k = v$, all the nodes and edges in the sequence are distinct, and the successive nodes n_i and n_{i+1} are endpoints of the intermediate edge e_i. A path is said to be a *cycle* if its first and last nodes (only) coincide.

Lemma 1 *Let B_1 and B_2 be the distinct blocks in a document. Let n_1 and n_2 be their corresponding nodes in the adjacent block graph of the document. If $B_1\|_v B_2$ or $B_1 =_h B_2$ then $n_1 \leftrightarrow n_2$.*

Proof: Consider the case $B_1\|_v B_2$. Proof of $n_1 \leftrightarrow n_2$ can be done by reducing the problem size to the H-distance.

1. If $B_1 \sim_h B_2$, then according to the definition of adjacent block graph (Definition 6), there is a H-edge between n_1 and n_2; and therefore $n_1 \leftrightarrow n_2$.

2. If $\neg(B_1 \sim_h B_2)$, according the definition of H-adjacency (Definition 3), there exists at least a block, saying B_3, such that its corresponding node n_3 in the adjacent block graph satisfies: $((x_1 < x_3 < x_2) \vee (x_2 < x_3 < x_1)) \wedge (B_1\|_v B_3) \wedge$

$(B_3\|_v B_2)$). By definition of H-distance (Definition 2), $(H\text{-}distance(B_1, B_2) > H\text{-}distance(B_1, B_3)) \land (H\text{-}distance(B_1, B_2) > H\text{-}distance(B_3, B_2))$.

And, if $n_1 \leftrightarrow n_3$ and $n_3 \leftrightarrow n_2$ then $n_1 \leftrightarrow n_2$. That is, we reduce the problem of proving $n_1 \leftrightarrow n_2$ to the problem of proving $n_1 \leftrightarrow n_3$ and $n_3 \leftrightarrow n_2$ with shorter H-distances.

If there is no H-edge between n_1 and n_3, nor n_3 and n_2 (that is, if $\neg(B_1 \sim_h B_3)$ or $\neg(B_3 \sim_h B_2)$), we can reduce the problem of proving $n_1 \leftrightarrow n_3$ or $n_3 \leftrightarrow n_2$ again.

By the definition of H-distance (Definition 2), since the H-distance of any two V-overlapping blocks is greater than zero, at last, we can reduce the problem to the two blocks which are H-adjacent, such that their corresponding nodes have edge. Note that all the edges on the path are H-type.

We can prove the case $B_1 =_h B_2$ in the same manner. □

Definition 13 *(Connected Graph)*

A graph G is called a *connected graph* if every two nodes n_1 and n_2 in G, $n_1 \leftrightarrow n_2$.

Theorem 1 *Any non-trivial corresponding adjacent block graph of a segment in the document is a connected graph.*

Proof: To prove the connectivity, we need to prove that there is a path for every two nodes in the adjacent block graph. Given any two blocks B_1 and B_2, they must have one of the following relationships: (1) $B_1\|_v B_2$; (2) $B_1 =_h B_2$; and (3) $\neg(B_1\|_v B_2) \land \neg(B_1 =_h B_2)$.

According to Lemma 1, if B_1 and B_2 satisfy the relationships (1) or (2), then $n_1 \leftrightarrow n_2$. Now we prove the third case. We use the same method to reduce the size of the problem to both H-distance and V-distance.
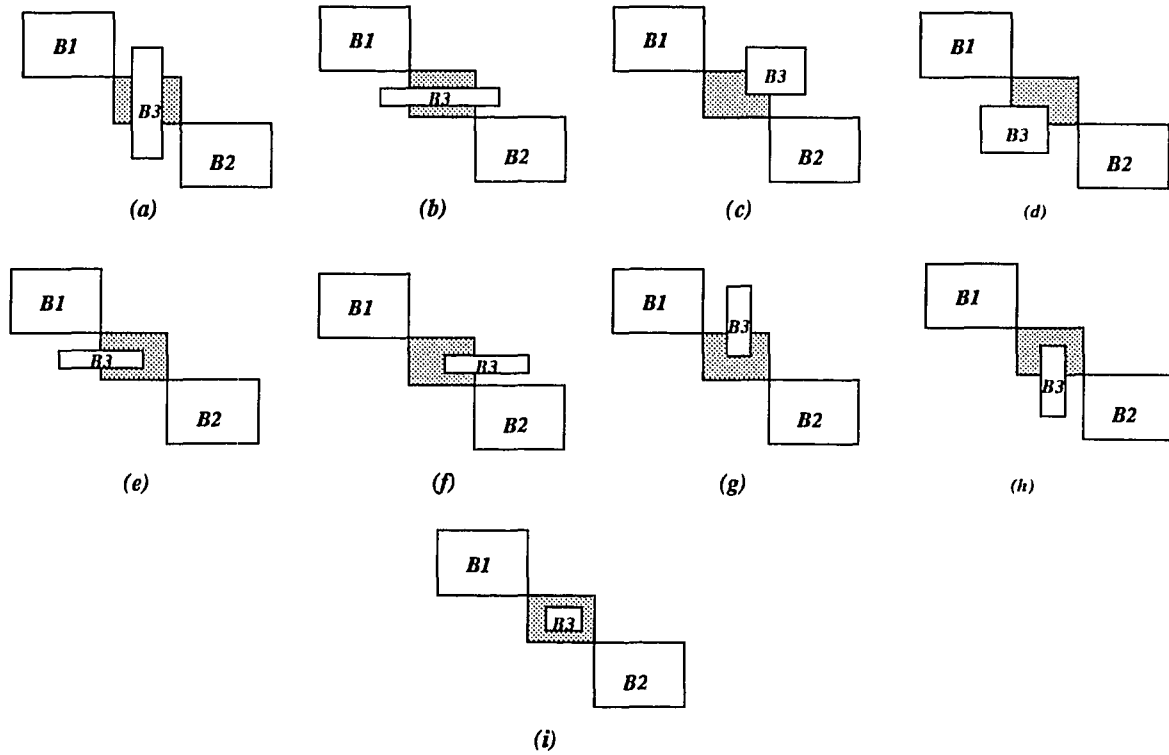
Figure 3.11 All possibilities of locations of B_3

1. If B_1 and B_2 are D-adjacent, then according to the definition of adjacent block graph (Definition 6), there is a D-edge between n_1 and n_2; thus $n_1 \leftrightarrow n_2$.

2. If B_1 and B_2 are not D-adjacent, according to the definition of D-adjacency (Definition 5), there exists at least a block, saying B_3 and its corresponding node n_3 in adjacent block graph. Figure 3.11 illustrates all the possibilities of locations of B_3 implied by the definition of D-adjacency.

In case (a), since $B_1 \|_v B_3$ and $B_3 \|_v B_2$, according to Lemma 1, we have $n_1 \leftrightarrow n_3$ and $n_3 \leftrightarrow n_2$, and therefore $n_1 \leftrightarrow n_2$. The same conclusion holds in case (b).

In case (c), since $B_1 \|_v B_3$, according to Lemma 1, we have $n_1 \leftrightarrow n_3$. Therefore, we can reduce the problem to the proof of $n_3 \leftrightarrow n_2$, where $\neg(B_3 \|_v B_2) \wedge \neg(B_3 =_h$

B_2), to their H-distance. The similar conclusions hold for the cases (d), (e), (f), (g) and (h) to the H-distance or V-distance.

For the case (i), obviously, we can reduce the problem to the proof of $n_1 \leftrightarrow n_3$ and $n_3 \leftrightarrow n_2$, where $\neg((B_1\|_v B_3) \vee (B_1 =_h B_3)) \wedge \neg((B_3\|_v B_2) \vee (B_3 =_h B_2))$, to their both H-distance and V-distance. $\square$

Definition 14 *(H-Segmentation and V-Segmentation)*

Given a segment S of blocks and $S_1, S_2 \subset S$, we call (S_1, S_2) a *H-segmentation* on S if:

- $S_1 \neq \phi$ and $S_2 \neq \phi$;

- $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \phi$;

- $\forall B_1 \in S_1,\ \forall B_2 \in S_2,\ \neg(B_1 =_h B_2)$.

Similarly, we call (S_1, S_2) a *V-segmentation* on S if:

- $S_1 \neq \phi$ and $S_2 \neq \phi$;

- $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \phi$;

- $\forall B_1 \in S_1,\ \forall B_2 \in S_2,\ \neg(B_1 \|_v B_2)$.

The case that there are more than two segmentations on segment S can be defined by applying the above definitions recursively.

Definition 15 *(Spanning tree of a graph and chords of spanning tree)*

Given a connected graph $G(V, E)$ where V is the set of nodes in the G and E is the set of edges in the G, a tree $T(V', E')$ is called a *spanning tree* of G if $V' = V$ and $E' \subseteq E$. An edge of G not lying in T is called a *chord* of T.

Definition 16 *(fundamental cycle)*

Let $G(V, E)$ be a connected graph where V is the set of nodes in G, and E is the set of edges in G; and let T be the spanning tree of G. The cycle created by adding one chord to the T is called a *fundamental cycle* in G.

Lemma 2 *Given a connected graph G and a spanning tree T of G, an edge e of T plus some chords create a minimal cuts. Those chords must be the chords such that: when they are added to T, a fundamental cycle containing e is created. This minimal cut is called a fundamental minimal-cut* [5]. □

Definition 17 *(Symmetric difference)*

Given two subgraphs G_1 and G_2 of a graph G, the *symmetric difference* of G_1 and G_2 is the graph that results by removing any edges from G that G_1 and G_2 have in common as well as any isolated nodes that result after the removal of these edges.

Theorem 2 *Let G be a connected graph, and let T be a spanning tree of G. Then, every minimal cut MC_G is the symmetric difference of the fundamental minimal cuts determined by the edges of MC_G lying T* [28]. □

Theorem 3 *Given a segment S of blocks of a document, if the number of blocks is greater than 1, there exists at least one segmentation on S.*

Proof: Let G be the corresponding adjacent block graph of S. G is a non-trivial graph since the number of blocks is greater than 1. By theorem 1, G is a connected graph, such that we always can find a spanning tree of the G [28]. By theorem 2, there exist minimal cuts in G.

We need to show that each minimal cut of H-type corresponds a V-segmentation on S and each minimal cut of V-type corresponds a H-segmentation on S.

Let MC_G be the minimal cut of H-type. Let and the C_1 and C_2 be the two components of G resulted from the removal of MC_G, and, S_1 and S_2 be their corresponding block sets respectively. We need to show that (S_1, S_2) is a V-segmentation

Obviously, $S_1 \neq \phi$, $S_2 \neq \phi$, and $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \phi$. It remains to prove that $\forall B_1 \in S_1$, $\forall B_2 \in S_2$, $\neg(B_1 =_h B_2)$.

Suppose that the statement is false. Then $\exists B_1 \in S_1, \exists B_2 \in S_2$ $B_1 =_h B_2$ and their corresponding nodes in G are n_1 and n_2.

By Lemma 1, we know that $n_1 \leftrightarrow n_2$ and every edge of the path is a V-edge. That is, there must be nodes $n_1' \in C_1$ and $n_2' \in C_2$ connecting by a V-edge which is in the MC_G. Otherwise, MC_G would not be a cutset. But this contradicts the fact that MC_G is of H-Type, which does not contain any V-edges. □

In the process of the nested segmentation, the encoded document is transformed to the corresponding adjacent block graph. Then the nested segmentation focuses on solving the problem of finding a minimal cut that has the maximal weight in the adjacent block graph (Possibly, there are more than one minimal cuts having the same maximal weight. For this case, all these cuts are selected.) If a minimal cut of H-type is found, then a V-segmentation can be applied between the portions of the document which correspond to the subgraphs resulted from the removal of the found minimal cut. If the found minimal cut is of V-type, then a H-segmentation can be applied to the corresponding portions of the document. After the found minimal cuts are removed, we can apply the same process recursively to the components of the graph until no segment can be further segmented. Figure 3.12 summarizes the algorithm. D is the encoded document to be segmented and is represented by a set of blocks. SD is the segmented document which is the result of applying the nested segmentation to D. SD is represented by a segment quadruple $(Id, Type, Orientation, Composition)$.

An example of applying the process of segmentation to a memo is shown in Figure 3.13. Before the nested segmentation, the document represented in terms of blocks is transformed to the adjacent block graph. After the nested segmentation, the segmented document is transformed to a tree structure.

Nested Segmentation (D, SD)
begin
    transform D into the corresponding adjacent block graph G;
    /* based on the definition of adjacent block graph */
    if G is a trivial graph **then**
    /* G is a trivial graph if it contains only one node, in other words, document
    D contains only one block. Let B be the identifier of the block contained in D*/
        assign a segment identifier ID to SD; $SD := (ID, Basic, NULL, (B))$;
        exit. /* Algorithm finishes */
    else /* G is not a trivial graph */
        assign a segment identifier ID to SD;
        $SD := (ID,'','','')$; /*'' means the value of the item is not available at present. */
        while there exists a non-trivial component in G **do**
            begin
                for every non-trivial component C in G **do**
                    begin
                        find the segment quadruple $T = (ID,'','','')$ which is
                        associated with the C;
                        find a minimal cut which has the maximal weight MW
                        and the other minimal cuts of the same type that have the weight
                        W satisfying $MW - W < \epsilon$; [a]
                        remove all found minimal cuts;
                        for every component resulted from the removal of
                        the found minimal cuts **do**
                          begin
                              associate a segment quadruple TT with the segment;
                              assign a segment identifier ID; $TT := (ID,'','','')$;
                        end; /* for */
                        /* Let the newly assigned IDs be $S_1, S_2, ..., S_n$, located in the order
                        from top to bottom if the minimal cut is of V-type, or from
                        left to right if the minimal cut is of H-type. */
                        $T := (ID, Composite, Minimal\_cut\_type, (S1, S2, ..., Sn))$;
                  end; /* for */
            end; /* while */
        for every trivial component left **do**
            begin
                associate a segment quadruple T with the component;
                assign a segment identifier ID to it;
                /* Let B be the identifier of the block contained in the segment*/;
                $T := (ID, Basic, NULL, (B))$;
        end; /* for */
end; /* Nested Segmentation */

---

[a]The ϵ is a relatively small number.

Figure 3.12 Algorithm for the nested segmentation
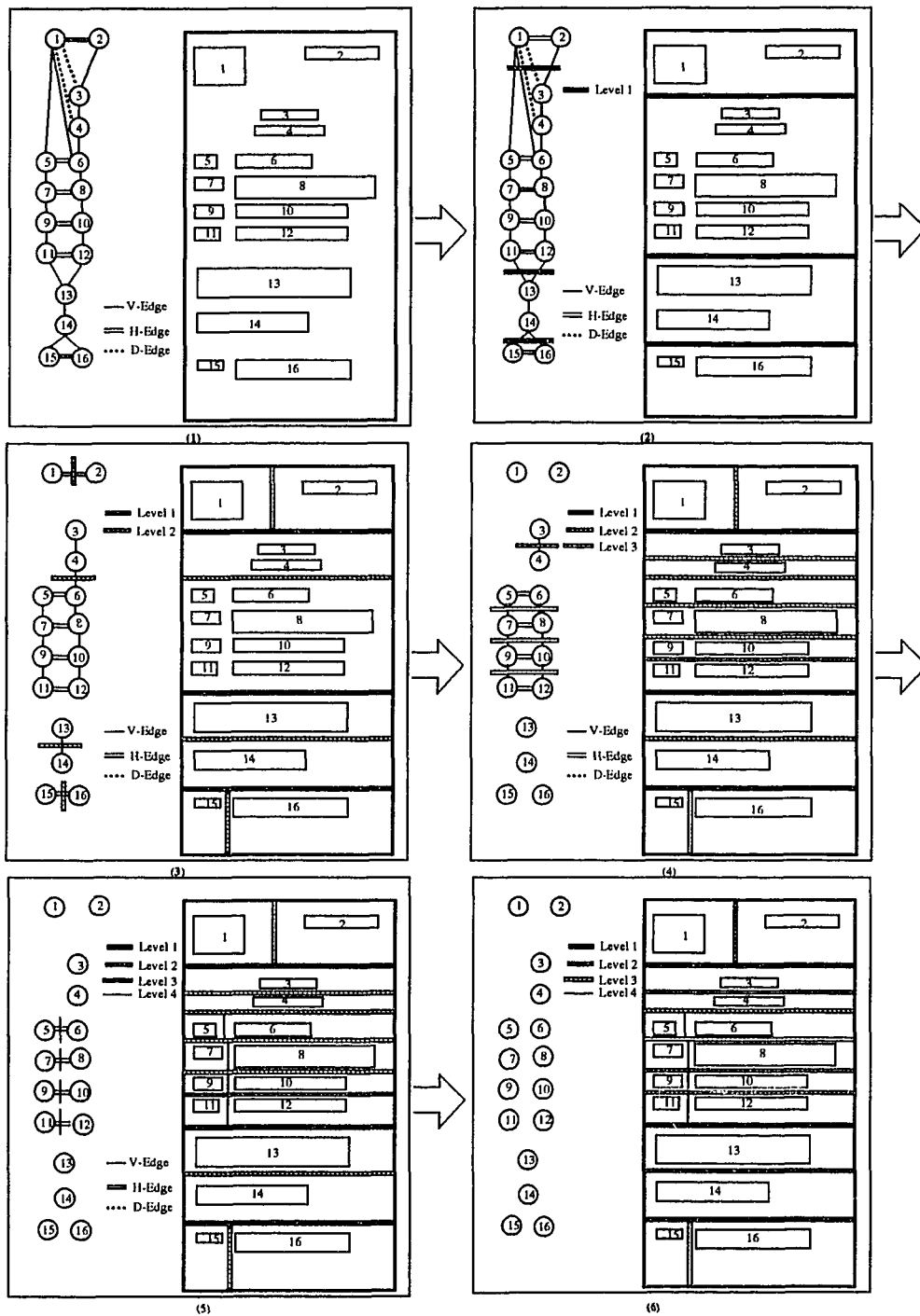
Figure 3.13 Illustration of the process of the nested segmentation

3.3 Representation of Nested Segmentation of Document

To describe the layout structure of a document accurately, a tree structure called the *Layout Structure Tree (L-S-Tree)* is proposed to represent the nested segmentation of the document. The L-S-Tree is an ordered labeled tree. The label is the type of the node. There are three types of nodes in the L-S-Tree: basic node *(B-node)*, horizontal node *(H-node)* and vertical node *(V-node)*. The order of the physical locations of siblings in the tree reflects precisely the order of segment locations in the document.

The process of transforming a nestedly segmented document into a L-S-Tree is as follows:

- If a segment is basic, then it is represented by a B-node;

- If a segment is composite and is divided horizontally into smaller segments, then it is represented by an H-node. The smaller segments are represented as the children of the H-node. The order of the children in the tree, from left to right, represents the order of segments in the document, from top to bottom (see Figure 3.14(a)) .

- If a segment is composite and is divided vertically into smaller segments, then it is represented by a V-node. The smaller segments are represented as the children of the V-node. The order of the children in the tree, from left to right, represents the order of segments in the document, from left to right (see Figure 3.14(b)).

The application of the above process of transformation to the nestedly segmented document in Figure 3.13 yields the L-S-Tree in Figure 3.15.
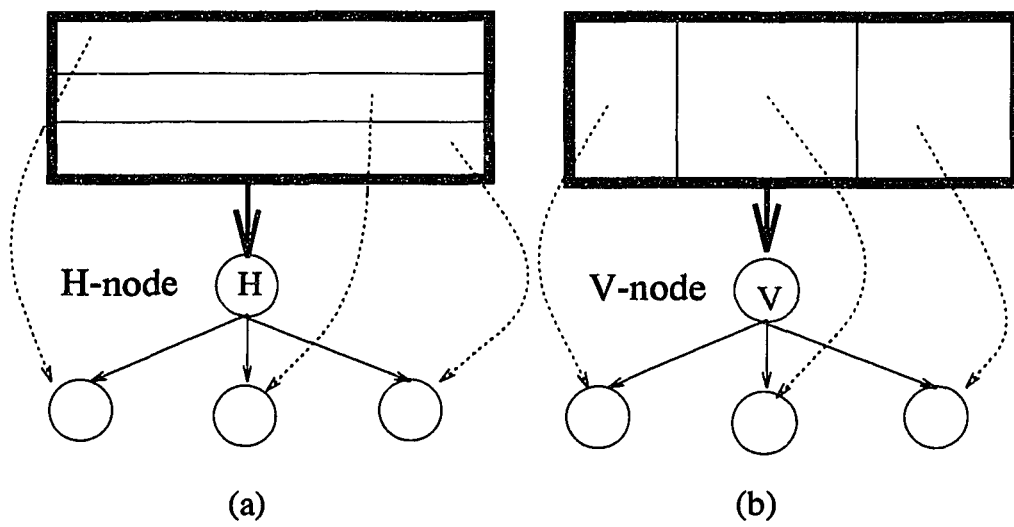
Figure 3.14 Transformation from segments to a L-S-Tree
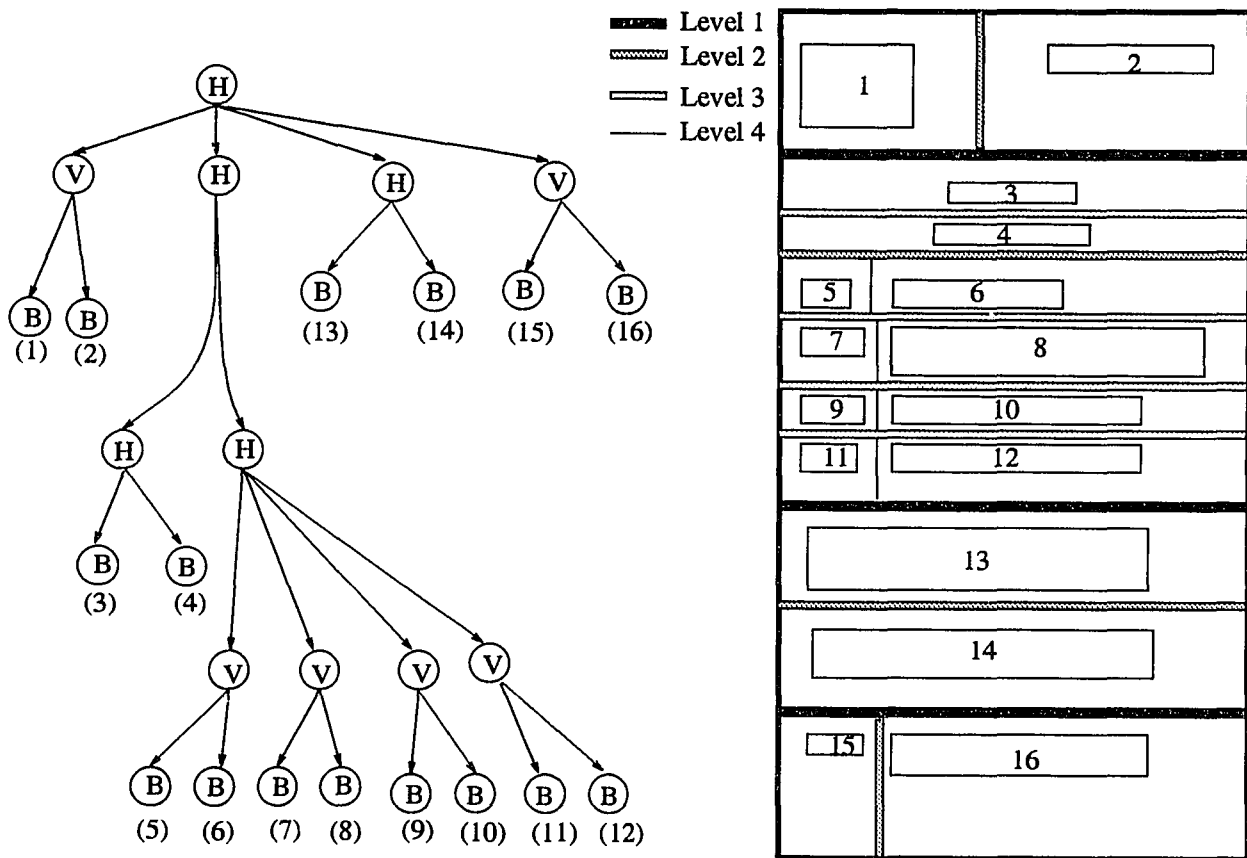
Figure 3.15 The corresponding L-S-Tree of the nested segmentation of the document in Figure 3.3

CHAPTER 4

CONCEPTUAL ANALYSIS — ANALYSIS FOR STRUCTURED PART OF THE DOCUMENTS

4.1 Conceptual Structure

In Chapter 1, the conceptual structure was introduced to facilitate the instantiation of the attributes of the frame template from the *structured part* of a document. The conceptual structure of a document is represented by a set of attribute descriptors which specify the properties of the values that may assign to the attributes to obtain the frame instance. Formally, the conceptual structure of a document D, denoted by $CCP\_S(D)$, is represented as

$$\{SAD_1, SAD_2, ..., SAD_n\}$$

where $SAD_i(1 \leq i \leq n)$ is an *structured part attribute descriptor* and is composed of

an attribute name which specifies the name of the attribute, denoted as $SAD_i(attr\_name)$;

an attribute type which specifies the type of the attribute, either atomic, composite or set, denoted as $SAD_i(attr\_type)$; and

an attribute domain which specifies the restrictions on values that may assign to this attribute, denoted as $SAD_i(attr\_domain)$, and is one of the following:

- o one of the data type such as integer, real, string, boolean, and text if $SAD_i(attr\_type)$ is atomic; or

- an attribute descriptor of the set element if $SAD_i(attr\_type)$ is set; or

- a set of attribute descriptors of the sub-attributes which form this attribute and composite pattern of SAD_i if $SAD_i(attr\_type)$ is composite.

46

Composite pattern is introduced to assist instantiation of the composite attributes. The composite pattern of a composite attribute defines the forms of all possible values that may be extracted from the document for the composite attribute. A composite attribute is instantiated by parsing the content of the associated portion of the document based on the pre-defined composite patterns.

Formally, let A be a composite attribute of the form $A(A_1, A_2, ..., A_n)$ where A_i ($1 \leq i \leq n$) is a sub-attribute of A and can be an atomic attribute or again a composite attribute. Let $A_{i_1}, A_{i_2}, ..., A_{i_j}$ and $A_{i_{j+1}}, A_{i_{j+2}}, ..., A_{i_n}$ be of atomic and composite attributes respectively. The *composite pattern* for an attribute A, denoted $CP(A)$, defines all the possible forms (the string patterns) of values for the attribute A.

Formally, the syntax of $CP(A)$ is given as follows:

$CP(A) ::= \ < string\ patterns >$

$< string\ patterns > ::= \ (< string\ pattern >)$

$\qquad\qquad\qquad | \ (< string\ pattern >)\ \text{OR}\ < string\ patterns >$

$< string\ pattern > ::= \ < symbol >< string\ pattern >$

$\qquad\qquad\qquad | \ < variable >< term1 > \ | \ < variable >$

$< term1 > ::= \ < symbol >< term2 > \ | \ < symbol > \ | \ < repeat\_symbol >$

$< term2 > ::= \ < variable >< term1 > \ | \ < variable >$

$< symbol > ::= \ < special\ symbol > \ | \ < punctuation\ mark >$

$\qquad\qquad | \ SPACE \ | \ < wild\ card >$

$< repeat\_symbol > ::= \ < symbol >^R$

$< variable > ::= \ A_{i_1} \ | \ A_{i_2} \ | \ ... \ | \ A_{i_j}$

$\qquad\qquad | \ CP(A_{i_{j+1}}) \ | \ CP(A_{i_{j+2}}) \ | \ ... \ | \ CP(A_{i_n})$

$< special\ symbol > ::= [\ | \ ] \ | \ \{ \ | \ \} \ | \ - \ | \ \_ \ | \ \backslash \ | \ @ \ | \ \sharp \ | \ \$ \ | \ \% \ | \ \wedge \ | \ \& \ | \ (\ | \ ) \ | \ = \ | \ \langle \ | \ \rangle$

$< punctuation\ mark > ::= ! \ | \ ? \ | \ , \ | \ . \ | \ : \ | \ ; \ | \ " \ | \ '$

$< wild\ card > ::= \ * \ | \ \#$

The grammar above has the following two characteristics:

1. There needs to be at least one variable in the composite pattern, otherwise there would be no instantiation;

2. There is at least one symbol between any two variables in the composite pattern so that the ambiguity in instantiation can be avoid.

For example, for the composite attribute (Date, (Year, Month, Day)), its composite pattern can be defined as:

$$CP(Date) = (\mathbf{Month}SPACE^R\mathbf{Day}, SPACE^R\mathbf{Year})$$
$$\text{OR}$$
$$(\mathbf{Month/Day/Year})$$
$$\text{OR}$$
$$(\mathbf{Year}SPACE^R\mathbf{Month}SPACE^R\mathbf{Day})$$

The given composite pattern includes three sub-patterns. Each sub-pattern defines a possible form for "Date". In each sub-pattern, the bold strings are the variables (attribute names) that are to be instantiated. The other symbols are used as delimiters to assist the instantiation of the variables.

Figure 4.1 shows a conceptual structure for a memo. As shown in the figure, usually, the attributes—sub-attributes relationships in a conceptual structure can be briefly described by a tree structure. Figure 4.2 shows the association between the attributes of conceptual structure and the contents of the structured part of the memo.

We mentioned in Chapter 1 that the documents of the same document type share the same frame template. We group these document types further based on their conceptual structures. The document types which share the same conceptual structure are of the same super document type. Thus, a conceptual structure

MEMO

Attr_name	Receiver
Attr_type	Composite
Attr_domain	(see sub-table below)

Attr_name	Name
Attr_type	Atomic
Attr_domain	String[10]
Attr_name	Title
Attr_type	Atomic
Attr_domain	String[20]
Composite Pattern	Name,[space]Title

Attr_name	Sender
Attr_type	Composite

Attr_name	Name
Attr_type	Atomic
Attr_domain	String[10]
Attr_name	Title
Attr_type	Atomic
Attr_domain	String[20]
Composite Pattern	Name,[space]Title

Attr_name	Subject
Attr_type	Atomic
Attr_domain	String[20]
Attr_name	Date
Attr_type	Composite

Attr_name	Year
Attr_type	Atomic
Attr_domain	Integer
Attr_name	Month
Attr_type	Atomic
Attr_domain	Integer
Attr_name	Day
Attr_type	Atomic
Attr_domain	Integer
Composite Pattern	Month[<space>] Day,[<space>]Year OR Month/Day/Year OR Year[<space>]Month[<space>] Day

Attr_name	Cc
Attr_type	Atomic
Attr_domain	Text

MEMO

Receiver — Sender — Subject — Date — Cc

Receiver: Name, Title
Sender: Name, Title
Date: Year, Month, Day

Figure 4.1 The conceptual structure of QE memo

NJIT

New Jersey Institute of Technology

Ph.D Program Committee

MEMORANDUM

CONFIDENTIAL

TO:	John Smith
FROM:	Dr. Mike Thomas, Chairman
	Director of Ph.D Program in Computer Science
SUBJ:	CIS Qualifying Examination
DATE:	May 21, 1991

I would like to inform you that the CIS Qualifying Examination Committee has recommended to me that you conditionally pass the qualifying examination. However, upon the Committee's recommendation, you must take a written re-examination on Formal Language and Programming Language within a year.

In preparation for the partial re-examination on the above named areas, you are advised to repeat relevant courses in the topic area.

| Cc: | Members of the Ph.D Program Committee in Computer Science. Full Professors, Associate Chairs. |

MEMO

Receiver   Sender   Subject   Date   Cc

Name   Title   Name   Title   Year   Month   Day

Figure 4.2 The association between the conceptual structure and the document
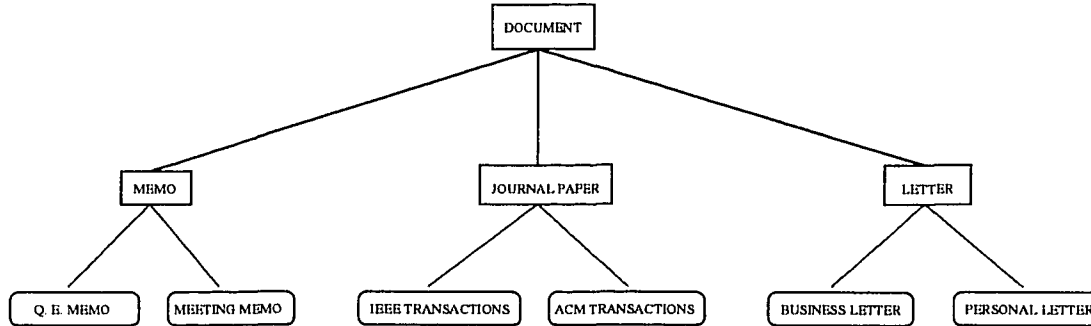
Figure 4.3 The document type hierarchy

is associated with a super document type. The document types and their super document type form a document type hierarchy. For example, Figure 4.3 shows a document type hierarchy with super type MEMO (the memorandum).

4.2 Document Sample Base

The information of the structured part of a given document is extracted by analyzing its layout and conceptual structures. The layout and conceptual structures of some pre-analyzed documents are kept in the sample base as document samples. For any incoming document, its layout and conceptual structures are analyzed by matching its structures with the structures of pre-stored document samples. In the following subsections, we will discuss the representation of the document samples and document structure analysis.

4.2.1 Representation of the Document Sample

From office documents of various types, we observed that each document can be divided into *structured* and *unstructured* parts. The structured part is further divided into two parts: *static* and *dynamic* parts. The static part has a fixed location and it has semantically the same content in different documents of the same document type. On the contrary, the dynamic part may vary considerably
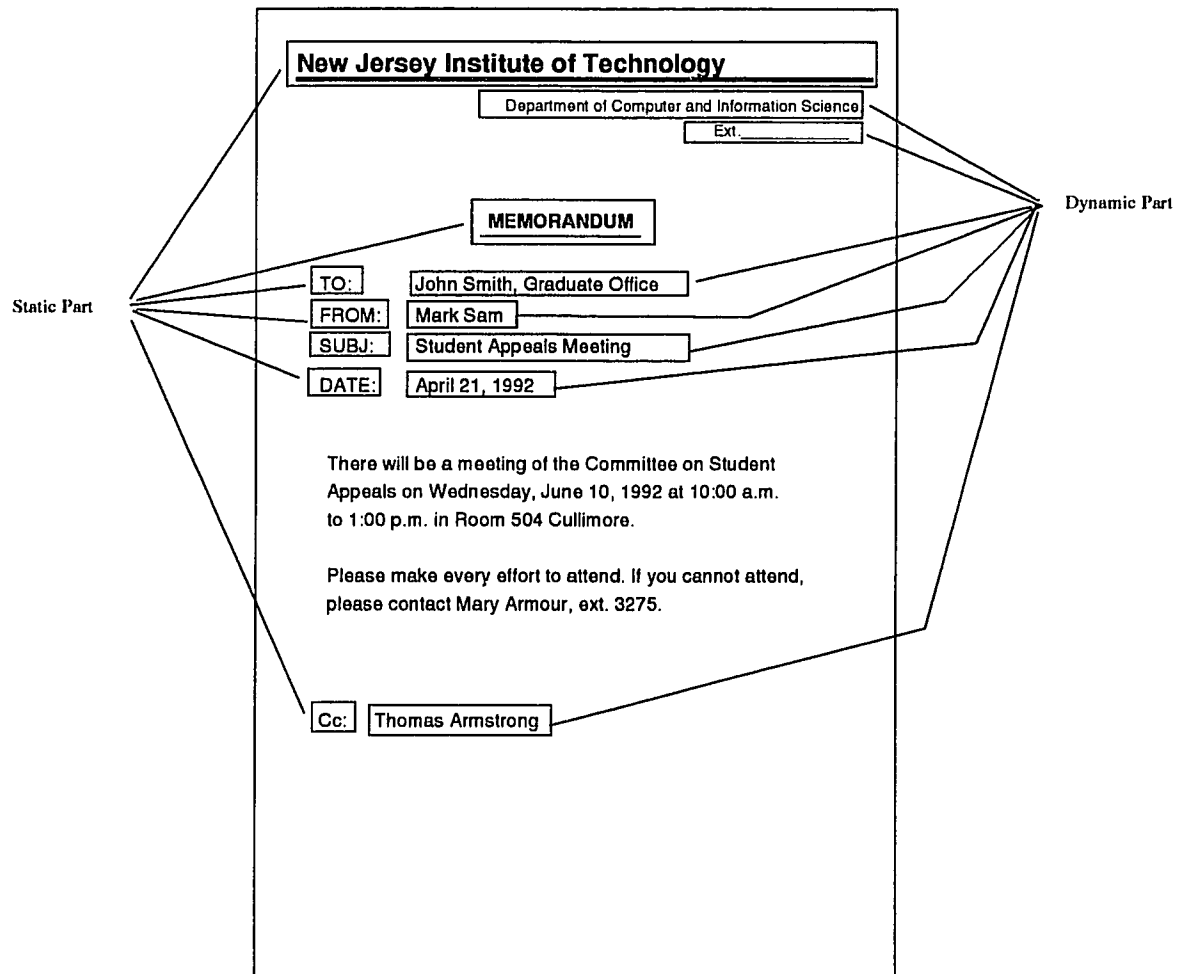
Figure 4.4 A sample memo

among documents. For example, Figure 4.4 shows a document sample with static and dynamic parts. The static part of the memo in the figure includes the words (or terms) "MEMORANDUM", "TO", "FROM", "SUBJ", "DATE", "Cc", etc. The dynamic part of the memo refers to the various strings "John Smith, Graduate Office", "Mark Sam", "Student Appeals Meeting", "April 21, 1992", and "Thomas Armstrong". The words appeared in the static part may be in different forms in the documents of the same type, but they have the same meaning. For example, the words "SUBJ" and "RE" are used in different memos to refer to the subject of the memo. These synonymous words are treated to be semantically equivalent and are stored in the thesaurus (We use "==" to denote semantical equivalence.). A dynamic part can be semantically associated with a static part. This kind of relationship is referred to as "semantic association". For example, "John Smith, Graduate Office" is semantically associated with "TO" because "TO" denotes that the functionality of "John Smith, Graduate Office" is the receiver of the memo.

A document sample contains the knowledge describing the layout and conceptual characteristics of a group of documents of the same document super type. It is represented by a *document sample tree*. The document sample tree is an L-S-Tree with its leaf nodes containing additional conceptual information regarding their corresponding blocks in the document sample. Specifically, each leaf node (also called basic node) of the document sample tree corresponds to a block of the structured part of the document. The unstructured part of the sample document is represented by a don't care node, labeled by a variable preceded with an underscore (Details about don't care node are described in Section 4.3.1.2). Each leaf node N of the document sample tree contains the content of its corresponding basic segment and the following attributes:

conceptual type, denoted by $N(type)$, specifies if the corresponding segment of the node N is static, dynamic, mixed or unstructured. The term "mixed" means the segment contains both static and dynamic type information.

conceptual role, denoted by $N(role)$, specifies the conceptual role of the corresponding segment of the node N, where

- $N(role)$ contains the content of the segment if $N(type)$ is static;

- $N(role)$ contains the attribute name of the corresponding conceptual structure if $N(type)$ is dynamic or mixed and the value of the attribute in the frame instance will be extracted from the content of N;

- $N(role)$ is null otherwise.

static term, denoted by $N(static\_term)$, specifies the content of the static part of the segment when $N(type)$ is mixed, and null otherwise.

semantic association, denoted by $N(association)$ and is used when the $N(type)$ is static, specifies the nodes N's whose $N'(type)$ is dynamic and $N'(role)$ is semantically associated with $N(role)$.

importance, denoted by $N(importance)$, shows to what the degree the node contributes to the identification of a document type.

Intuitively, the importance of a basic node N depends on how frequently N (including all the nodes N's with $N'(role)$ being semantically equivalent to $N(role)$) appears in the sample tress of the same document type in the sample base. Specifically, the $N(importance)$ is calculated as follows. Consider a set $\mathcal{S}$ of sample trees of the same document type. Let N be a basic node (leaf node) of a sample $S \in \mathcal{S}$. Then

$$N(importance) = |\ \{S'\ |\ S' \in \mathcal{S}\ \text{and}\ \exists\ \text{basic node}\ N' \in S', N'(role) == N(role),\ N \in$$
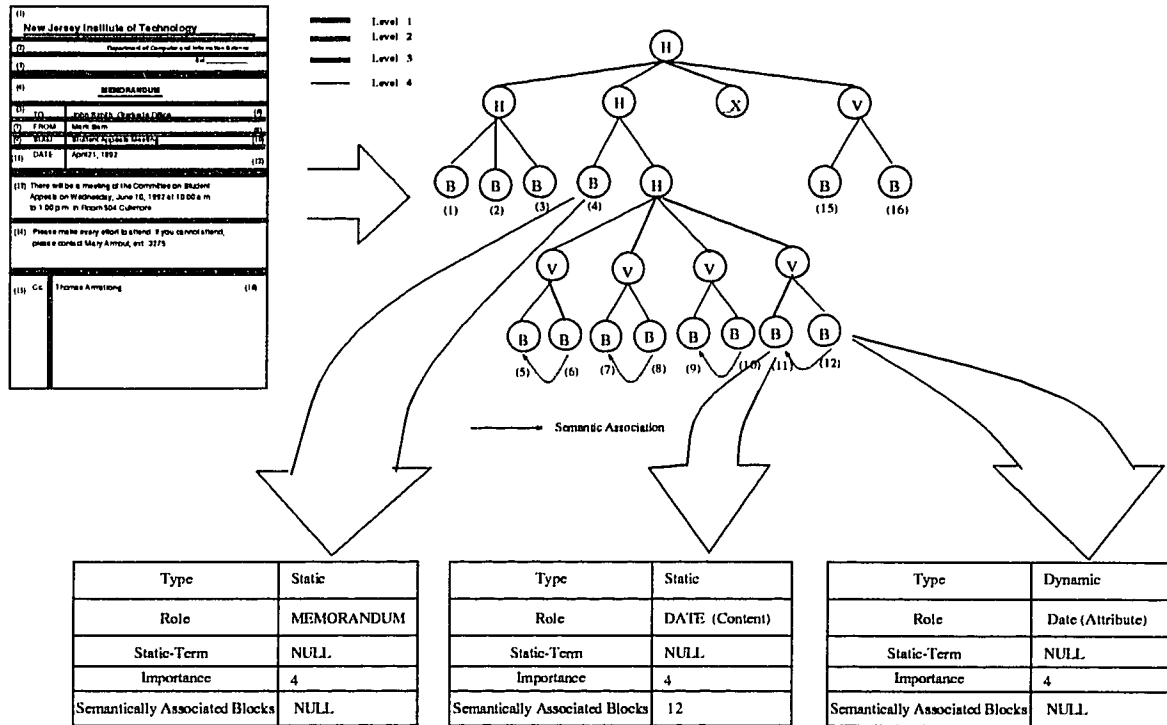
Figure 4.5 A sample memo and its corresponding sample tree

S} |

where |.| denotes the cardinality of the set.

Figure 4.5 show the sample tree obtained from the memo in Figure 4.4.

4.2.2 Document Sample Base

In our system, a document sample base is maintained to store all the document samples. In the document sample base, all the document samples are organized in a hierarchical form, namely, a document type (a document super type) is classified by several document samples. Figure 4.6 illustrates the organization of the document sample base. The document sample base, denoted by SD, is organized into a set of document types { $ST_1, ST_2, ..., ST_n$ }. Each ST_i ($1 \leq i \leq n$) is associated with a group of document sample trees { $DS_1{}^i, DS_2{}^i, ..., DS_{mi}{}^i$ } where $DS_j{}^i$ is the jth

DOCUMENT SAMPLE BASE
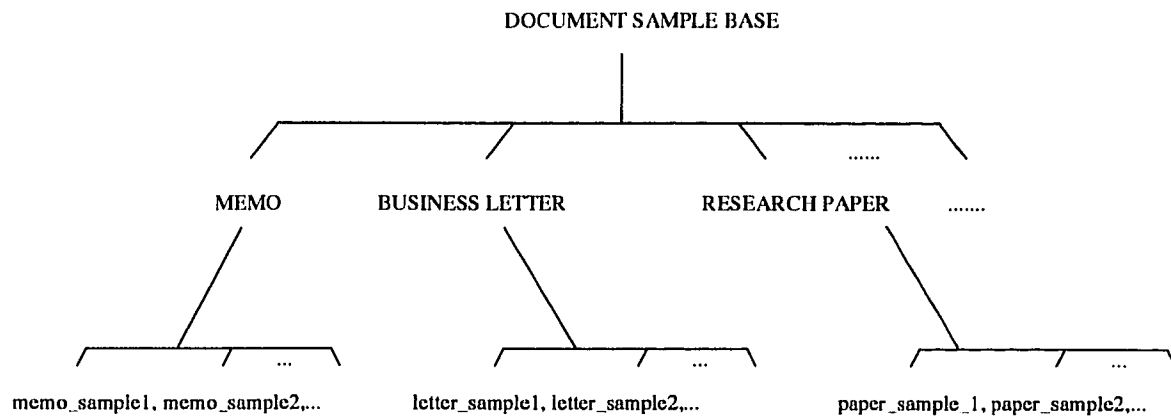
Figure 4.6 Document sample base

document sample tree of type ST_i. Recall that each document type is associated with one conceptual structure. Thus, the document sample tress $DS_1{}^i, DS_2{}^i, ..., DS_{mi}{}^i$ are associated with the same conceptual structure.

4.2.3 Document Sample Acquisition

Having discussed the representation of document samples and the sample base, we now discuss the process for acquiring document samples. The procedure of the acquisition of document sample is shown in Figure 4.7.

A document sample is first transformed to an encoded document represented by a collection of blocks. Then the user enters the following information for the sample:

- the document type of the sample (e.g., the sample is a memo, or journal paper or technical report etc.); and

- the type for each block, which can be

  - *static*, if the block contains only the materials of the static part;

  - *dynamic*, if the block contains only the materials of the dynamic part;
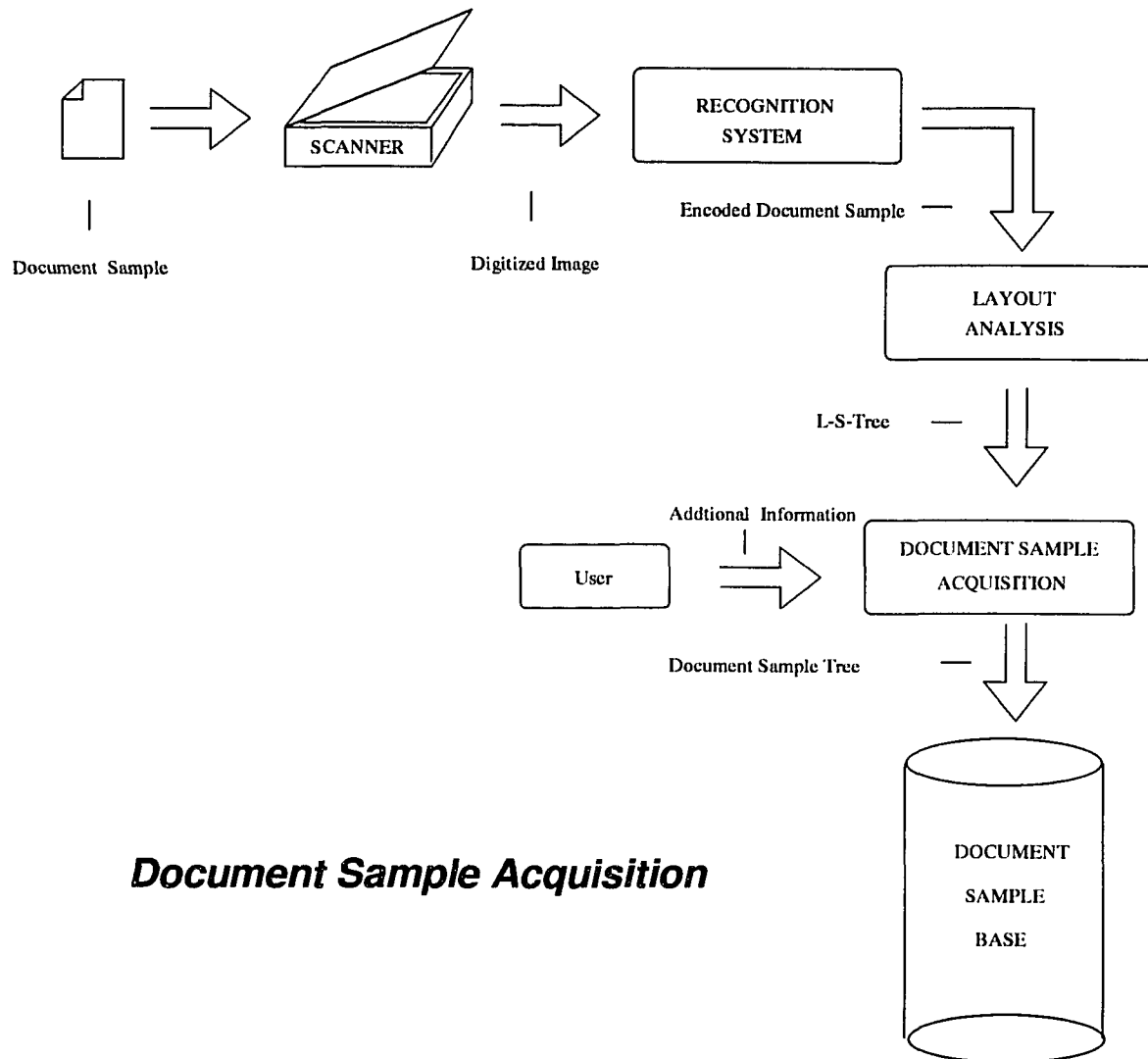
Figure 4.7 Procedure of document sample acquisition

– *mixed*, if the block contains the materials of both the static and dynamic part; or

– *unstructured*, if the block contains the materials of the unstructured part;

In addition, for the static and dynamic blocks that are semantically associated with each other, the user has to highlight their relationships by tagging the identifiers of the dynamic blocks to the corresponding static blocks (recalling that each block has an identifier). For the dynamic blocks whose contents correspond to the attributes of the conceptual structure, the user has to tag the attribute names to the corresponding blocks.

Figure 4.8 shows an interface screen of the document sample acquisition component of the prototype system. When the user wants to store a document as a document sample, the encoded form of document represented by a collection blocks is shown on the screen. The interface screens for acquiring the information for a block of the static type ("MEMORANDUM"") and a block of the dynamic type ("John Smith, Graduate Office") are shown in Figure 4.8 and Figure 4.9, respectively.

The document sample is represented by a document sample tree which is the corresponding L-S-Tree with its leaf node containing the above user-input information. Also, in the sample tree, the unstructured part of the sample document, which may include more than one block, is represented by one node in the sample tree called the don't care node, labeled with a variable preceded with an underscore. Thus, the sample tree incorporates the layout structure represented by a L-S-Tree, with the conceptual structure associated with sample tree's leaf nodes based on the user input.

4.3 Sample-Based Document Structure Analysis

The goal of document structure analysis is to identify the conceptual structure of a given document. In our sample-based approach, the conceptual structure of

Commands

Load Document
Classfication
Extraction
Filing
Quit

Document Type | Memo

Messages

OK. Now you need to give
a little bit more
information for each
block in the document.

DINEX SYSTEM

TEXPROS
PROJECT

Document

New Jersey Institute of Technology

MEMORANDUM

TO: John Smith, Graduate Office
FROM: Mark San
SUBJ: Student Appeals Meeting
DATE: April 21, 1992

There will be a meeting of the Committee on Student
Appeals on Wednesday, June 10, 1992 at 10:00 a.m.
to 1:00 p.m. in Room 504 Cullimore.

Please make every effort to attend. If you cannot attend,
please contact Mary Armour, ext. 3276.

CC: Thomas Armstrong

Frame Instance

Sender
Receiver
Subject
Date
Copy to

Figure 4.8 One interface of document sample acquisition component

Figure 4.9 Document sample acquisition for a block of the static type

Figure 4.10 Document sample acquisition for a block of the dynamic type

a given document is identified by finding a document sample which matches the given document. Recall that each document sample is associated with a conceptual structure. The layout analysis is used to facilitate the search of such a document sample.

Using nested segmentation, the L-S-Tree of a given document is constructed. Then, the constructed L-S-Tree is compared against each document sample tree in the sample base in order to find the document sample which matches the given document. The comparison is accomplished by using an approximate tree matching tool [48, 49, 57]. Figure 4.11 shows the procedure of document structure analysis. In the layout comparison phase, the layout similarity between the document and a sample is measured by the *edit distance* between the L-S-Tree of the document and the sample tree. If the sample passes the layout comparison (a pre-defined threshold is used to decide if the sample passes the layout comparison), the conceptual comparison phase proceeds. In the conceptual comparison phase, the conceptual similarity between the document and the sample is measured by the "conceptual closeness degree" between the L-S-Tree of the document and the sample tree. If the sample passes the conceptual comparison (a pre-defined threshold is used to decide if the sample passes the conceptual comparison), the conceptual structure associated with the sample tree is identified as the conceptual structure of the document. In the following subsections, we discuss the layout and conceptual comparisons in detail.

4.3.1 Layout Comparison

In the layout comparison phase, the edit distance between the L-S-Tree of the document and the sample tree is used to measure the layout similarity between the document and the sample.
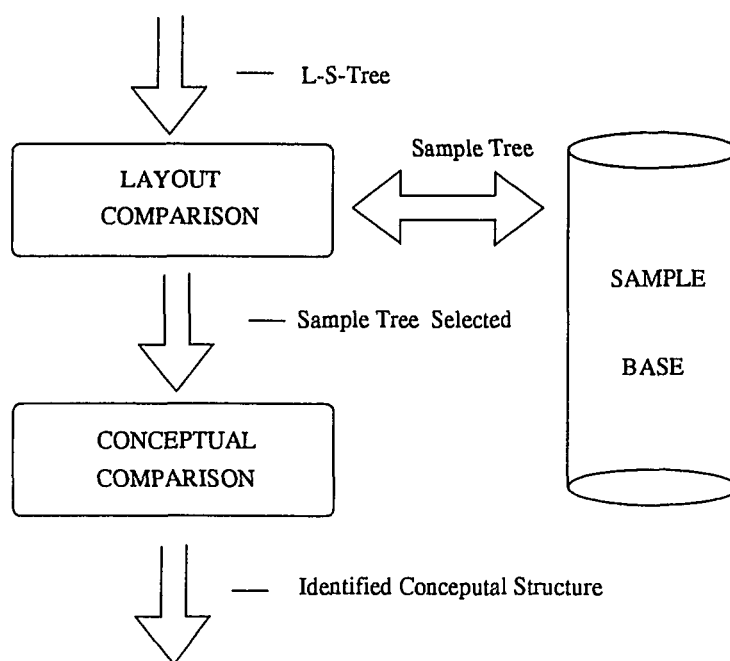
Figure 4.11 The procedure of sample-based document structure analysis

4.3.1.1 Edit Operations and Editing Distance for Approximate Tree Matching In approximate tree matching, the similarity between two trees is computed by editing one tree so that it is identical to the other tree. There are three types of edit operations: *relabel, delete,* and *insert* a node. We represent these operations as $u \to v$, where u and v are either a node or the null node(Λ). We call $u \to v$ a relabeling operation if $u \neq \Lambda$ and $v \neq \Lambda$; a delete operation if $u \neq \Lambda$ and $v = \Lambda$; or an insert operation if $u = \Lambda$ and $v \neq \Lambda$. Let T_2 be the tree obtained from the application of an edit operation $u \to v$ to tree T_1. This is written $T_1 \Rightarrow T_2$ via $u \to v$. Figure 4.12 illustrates the edit operations.

Let S be a sequence $s_1, s_2, ..., s_k$ of edit operations. A tree T is transformed to T' by applying S (or we say S transforms a tree T to T') if there is a sequence of trees $T_0, T_1, ..., T_k$ such that $T = T_0, T' = T_k$ and $T_{i-1} \Rightarrow T_i$ via s_i for $1 \leq i \leq k$.

The definition of edit operations is an abbreviation for the specification. Consider a single edit operation, e.g., one that transforms T_{i-1} to T_i. If it is a
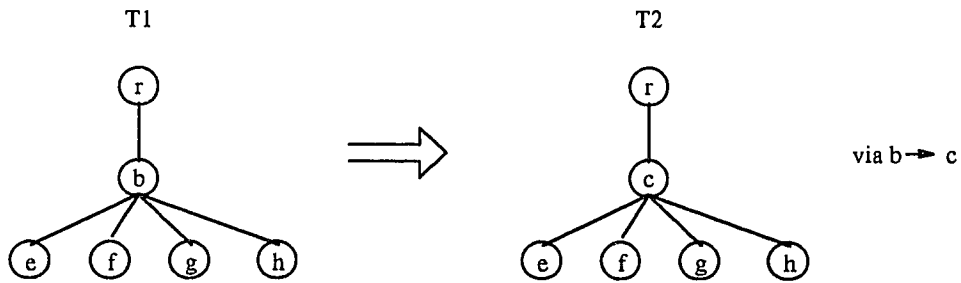
(1) Relabeling to change one node label b to another c.
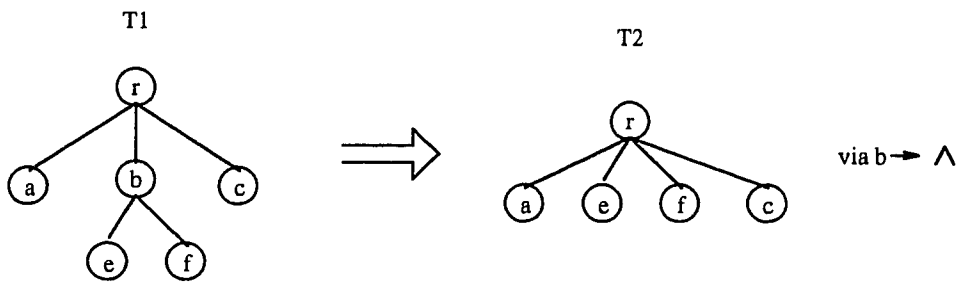
(2) Deletion of a node. (All children of the deleted node b becomes children of the parent r.)
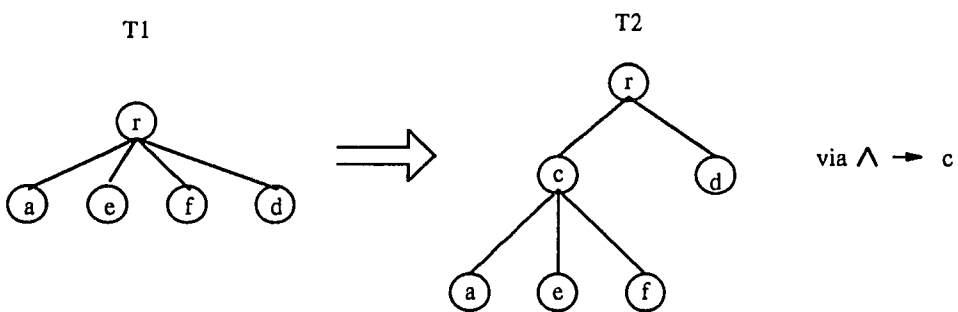
(3) Insertion of a node. (A consecutive sequence of siblings among the children of r (here, a, e and f) become the children of c.)

Figure 4.12 Examples illustrating the edit operations.

relabeling or delete operation, we specify the node to be relabeled or deleted in T_{i-1}. For an insert operation, we specify the parent p of the node n to be inserted and the consecutive sequence of siblings among the children of p will be the children of n. If this consecutive sequence is empty, then we need to specify the position of n among the children of p. The abbreviation of edit operations will be used if the specifications are clear from the mapping structure defined below.

Let γ be a cost function that assigns to each edit operation $u \to v$ a nonnegative real number $\gamma(u \to v)$. Let γ be restricted to be a distance metric satisfying the following three properties:

- $\gamma(u \to v) \geq 0$ and $\gamma(u \to u) = 0$;

- $\gamma(u \to v) = \gamma(v \to u)$ (symmetry);

- $\gamma(u \to w) \leq \gamma(u \to v) + \gamma(v \to w)$ (triangle inequality).

We extend γ to a sequence of edit operations $S = s_1, s_2, ..., s_k$ by letting $\gamma(S) = \sum_{i=1}^{k} \gamma(s_i)$. The editing distance, or simply the distance, from a tree T to another tree T', denoted as $dist(T, T')$, is defined to be the minimum cost of all sequences of edit operations which transform T to T', i.e., $dist(T, T') = \min \{\gamma(S) \mid S$ is a sequence of edit operations transforming T to $T'\}$.

By the definition of γ, this distance is a distance metric; it means that given three trees T, T', and T'', $dist(T, T'') \leq dist(T, T') + dist(T', T'')$.

The edit operations applied to each node in the two trees correspond to a *mapping*. The mapping in Figure 4.13 shows a way of transforming T into T'. It corresponds to a sequence of edit operations: delete (node with label d), insert (node with label d). A dotted mapping line from a node u in T to a node v in T' indicates that u should be changed to v if $u \neq v$, or that u remains unchanged if $u = v$. The node of T not touched by a dotted line are to be deleted from T and the nodes of T' not touched are to be inserted into T.
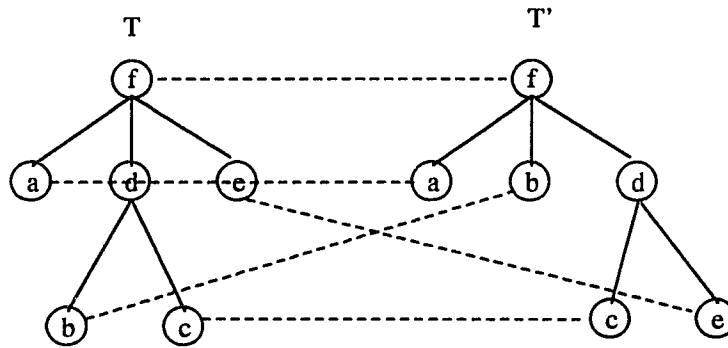
Figure 4.13 A mapping from T to T'

Formally, a mapping from a tree T to a tree T' is a triple (M, T, T') (or simply M if there is no confusion), where M is defined as follows:

Let n_1 and n_2 be the nodes of T, n_1' and n_2' be the nodes of T'. M is a mapping which consists of two edit operations, denoted as $n_1 \xrightarrow{M} n_1'$, and $n_2 \xrightarrow{M} n_2'$, satisfying the following conditions:

1. $n_1 = n_2$ if and only if $n_1' = n_2'$ (one to one);

2. n_1 is to the left of n_2 if and only if n_1' is to the left of n_2' (sibling order preserved);

3. n_1 is an ancestor of n_2 if and only if n_1' is an ancestor of n_2' (ancestor order preserved).

Let M be a mapping from T to T'. Let I and J be the sets of nodes in T and T' respectively, where $n_i \xrightarrow{M} n_j, (n_i \in I, n_j \in J)$. Then the cost of M from T to T' is defined as:

$$\gamma(M) = \sum_{n_i \in I, n_j \in J} \gamma(n_i \to n_j) + \sum_{n_i \notin I} \gamma(n_i \to \Lambda) + \sum_{n_j \notin J} \gamma(\Lambda \to n_j).$$

Given a sequence S of edit operations from T to T', it can be shown that there exists a mapping M from T to T' such that $\gamma(M) \leq \gamma(S)$; conversely, for any mapping M, there exists a sequence S of edit operations such that $\gamma(S) = \gamma(M)$ [56].

Hence, we have

$$dist(T, T') = \min \{\gamma(M) \mid M \text{ is a mapping from } T \text{ to } T'\}.$$

4.3.1.2 Approximate Tree Matching An approximate tree matching is an approximate comparison of ordered, labeled trees [57, 49]. An ordered, labeled tree is a tree whose nodes are labeled and the order among its siblings from left to right is significant. The L-S-Tree and sample tree are ordered, labeled trees. The order of the siblings reflects the corresponding layout locations of the blocks labeled as H, V or B in the document. Approximate tree matching is used for measuring the similarity of two trees by finding a minimum-cost set of deletion, insertion and relabeling operations that converts one tree to the other. The mapping corresponding to the sequence of operations is called *the best mapping* [57, 49].

In our document conceptual analysis, both the document sample and the document to be analyzed are represented by the ordered, labeled trees. In addition to having constant nodes whose labels are specified as H, V and B, a sample tree may contain the variable nodes, denoted as _x, _y, etc. They correspond to the unstructured part of the document whose layout structure should not affect the result of the conceptual analysis. When a sample is matched against a document, a variable node of the sample tree will be instantiated into a subtree of the L-S-Tree of the document with zero cost. The detail about algorithm of approximate tree matching can be found in [57].

For example, Figure 4.14 illustrates the mapping of how approximate tree matching would transform the L-S-Tree of memo D of Figure 3.15 to the sample tree S of Figure 4.5. The transformation reconciles the difference between the top header portions of each memo's stationary. The sample S has three vertically adjacent blocks ("New Jersey Institute of Technology," "Department of Computer and Information Science," and "Ext ___"), where the document D has two horizontally adjacent blocks

(NJIT logo and "Ph.D Program Committee"). The transformation "relabels" node 2 in D (i.e., the block in the top header portion of the memo's stationary) as an H-node, followed by inserting the missing B-node as the rightmost child of node 2. Second, the transformation reconciles the missing "confidential" in D (node 15 — one of the two basic nodes which are the children of H-node 12) by deleting nodes 12 and 15. (Node 14 — the word "MEMORANDUM" — remains.) The variable $\_x$ in S is instantiated by the subtree rooted at node 4 in D. The "cost" of a mapping is defined to be the cost of inserting unmapped nodes of D (i.e., those not touched by a "mapping line" — see Figure 4.14), plus the cost of deleting nodes of S not touched by a mapping line, plus the cost of relabeling nodes in those pairs related by mapping lines with different labels. The approximate tree matching tool calculates the edit distance between a document tree D and a sample tree S with variables by first finding the best substitution of the variables in S, and then finding the "best mapping" (i.e., the minimal cost mapping) between the resulting variable-free trees. The edit distance of D and S, denoted $dist(D,S)$, is equal to the cost of the best mapping between D and S [56]. The mapping in Figure 4.14, for example, is the best one.

In comparing two trees, we use a cost function to evaluate the cost of edit operations applied to a node based on the number of its descendants. Intuitively, the cost of an edit operation applied to a node depends on how much the node plays a role in the layout structure of the corresponding document. Formally, given a node N in the L-S-Tree, the cost function of an edit operation applied to N, denoted $cost(N)$, is defined as follows:

- If N is a leaf node, then $cost(N) = 1$.

- Otherwise, if $N_1, N_2, ..., N_m$ are the children of N, then $cost(N) = \sum_{i=1}^{m} cost(N_i)$.
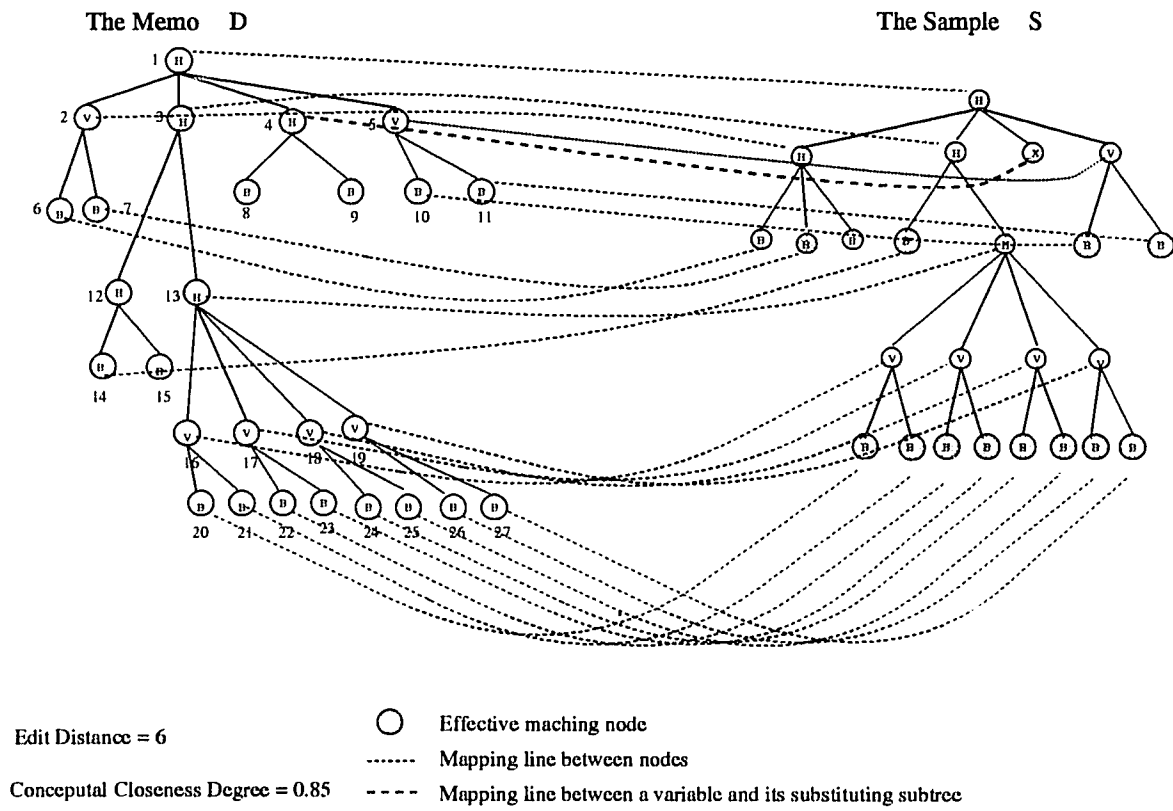
Figure 4.14 The best mapping between a document tree and a sample tree

For example, the cost of mapping in Figure 4.14 is the cost of relabeling node 2 in D as an H-node (which is 2), plus the summation of the cost of inserting a B-node as the rightmost child of node 2 (which is 1), the cost of deleting node 12 (which is 2), and the cost of deleting node 15 (which is 1). Thus, the cost of this mapping $dist(D, S) = 6$.

A threshold $(dist\_Threshold(S))$ is used for determining the layout similarity in terms of the edit distance between a L-S-Tree D and a sample tree S. The $dist\_Threshold(S)$ is calculated based on the threshold defined for the conceptual similarity. If the edit distance between S and D is bounded by the $dist\_Threshold(S)$, the conceptual comparison phase proceeds to determine the conceptual similarity between S and D.

4.3.2 Conceptual Comparison

In the conceptual comparison phase, the *conceptual closeness degree* between the L-S-Tree of a document and a sample tree is used to measure the conceptual similarity between the document and the sample. The conceptual closeness degree is defined in terms of *effective matching nodes*.

Definition 18 *(Effective Matching Nodes)*

Given a sample tree S and the L-S-Tree D of a document to be analyzed, let M be the best mapping yielding the edit distance between S and D. Let $N_S \xrightarrow{M} N_D$ be an edit operation applied to two basic nodes (blocks) $N_S \in S$ and $N_D \in D$ in M. Let $N_D(content)$ refer to the content of a block (recalling that each block has a content component; see Chapter 3). We will use the number of effective matching nodes to calculate the *degree of conceptual closeness* between two documents.

There are three kinds of effective matching nodes in S.

- A static node $N_S \in S$ is said to be an effective matching node if there exists a basic node $N_D \in D$ such that

- $N_S \xrightarrow{M} N_D$ and

- $N_S(role) == N_D(content)$.

- A mixed node $N_S \in S$ is said to be an effective matching node if there exists a basic node $N_D \in D$ such that

  - $N_S \xrightarrow{M} N_D$ and

  - there exists a term T in $N_D(content)$ such that $N_S(static\_term) == T$.

- A dynamic node $N_S \in S$ is said to be an effective matching node if there exists a basic node $N_D \in D$ such that

  - $N_S \xrightarrow{M} N_D$ and

  - there is a static node $N_S' \in S$ such that N_S' is semantically associated with N_S where N_S' must also be an effective matching node.

For example, consider again the L-S-Tree and the sample tree in Figure 4.14. The shaded nodes in the sample tree represent the effective matching nodes found by the mapping. The similarity between the document and the sample is evaluated by their *degree of conceptual closeness*.

Definition 19 *(Degree of Conceptual Closeness)*

Suppose that the sample tree S contains m basic nodes $N^1, N^2, ..., N^m$. Let M be the best mapping yielding the minimum edit distance between S and the document tree D. Let $N_e^1, N_e^2, ..., N_e^k$ in S be the effective matching nodes found by M, where $k \leq m$. We define the *degree of conceptual closeness* between S and D, denoted as $C\_DEG(S, D)$, as:

$$C\_DEG(S, D) = \frac{\sum_{i=1}^{k} N_e^i(importance)}{\sum_{i=1}^{m} N^i(importance)}$$

Clearly, $0 \leq C\_DEG(S, D) \leq 1$. Intuitively, the $C\_DEG(S, D)$ expresses explicitly how much the portions which characterize the conceptual functionality of the document type appear in the document D.

For example, suppose the importance of each basic node of S in Figure 4.14 is 1. Then $C\_DEG(S, D) = \frac{11}{14} = 0.85$.

We use an constant $C\_DEG\_Threshold$ to measure the degree of conceptual closeness. The $dist\_Threshold(S)$ is computed based on the $C\_DEG\_Threshold$ as follows:

$$dist\_Threshold(S) = Cost(S) \times (1 - C\_DEG\_Threshold)$$

where $Cost(S) = \sum cost(N)$, $N \in S$. Intuitively, the edit distance between D and S increases as the sample tree size increases while maintaining the same degree of conceptual closeness between D and S.

4.4 Identification of Document Super Type

In the conceptual analysis, the conceptual structure of a incoming document D is identified by finding a document sample S in the sample base such that the $C\_DEG(S, D)$ is greater than $C\_DEG\_Threshold$. The conceptual structure of the found document sample is identified as the conceptual structure of the document. The identified conceptual structure will then be used for extracting information from the structured part of the document. Since a document super type is associated with each conceptual structure, thus the super type of the document is identified.

Figure 4.15 summarizes the algorithm.

4.5 Instantiation of the Conceptual Structure

The information of the structured part of a document is extracted by instantiating the attributes of the identified conceptual structure. The conceptual structure is instan-

Conceptual Analysis (D, SB)

/* D is the document to be processed; it is represented by a L-S-Tree. */

/ * SB is the document sample base. */

begin

    repeat

        find a sample S of super type T such that $dist(D, S) \leq dist\_Threshold(S)$;

        calculate $C\_DEG(S, D)$;

        if $C\_DEG(S, D) \geq C\_DEG\_Threshold$

            then identify D as having the super type of T; **exit**

    until trying out all qualifying samples;

    if D cannot be identified by the samples in SB

        then store D as a new sample in SB ;

end;

Figure 4.15 Algorithm for super type identification

tiated at several levels by providing it with values extracted from the document. The first level of instantiation begins by associating the attributes of the conceptual structure with the corresponding blocks of the document. The second level of instantiation extracts the values of the attributes from the contents of the associated blocks.

4.5.1 Associations Between Attributes and Blocks

For extracting information from the structured part of the document, we need to know first which part of the content in the document is related to the attribute of the conceptual structure. This is done by associating each attribute of the conceptual structure with a block of the document based on the mapping found between the L-S-Tree D of the document and the sample tree S of a document sample.

For determining the degree of conceptual closeness between a sample and the document to be classified, every node of the sample tree is assigned by a weight (the importance of the node). By taking only some nodes of the sample tree with relative large weights into consideration, it is possible to obtain the degree of conceptual closeness which is above the threshold. Thus, the number of attributes which are

associated with the nodes involved in the matching may be less than the number of entire attributes in the conceptual structure. However, in order to extract information from the structured part of the document, the contents of the document which are related to all the attributes of the conceptual structure need to be located based on the mapping found in the tree matching. Therefore, we need to find a sample which has the most attributes involved in the matching common with the conceptual structure of the document.

Definition 20 *(Degree of Completeness of Matching and Complete Matching)*

Given the L-S-Tree D of a document whose super type identified by a sample tree S, let M be the matching between D and S and let $N_1, N_2, ..., N_i$ be the nodes of S which are associated with the attributes of the conceptual structure of S. Let $N^1, N^2, ..., N^j$ be the effective matching nodes of $N_1, N_2, ..., N_i$ ($j \leq i$). The *degree of completeness of matching*, denoted as $DC(M)$, is define to be:

$$DC(M) = \frac{j}{i}.$$

A matching M between a document D and a sample S is called a *complete matching* if $DC(M)$ is 1.

The procedure of associating an attribute A of the conceptual structure of the document with a block of the document includes following steps:

1. After the document D is classified by a sample tree S' of the sample base, if the matching between them is not a complete matching, a search is activated to find the sample S of the same document type which has the highest degree of completeness of matching;

2. For the leaf node $N_S \in S$ where $N_S(type)$ is dynamic and $N_S(role)$ is the attribute A in the corresponding conceptual structure (see the representation of sample tree and Figure 4.5), find the node $N_D \in D$ where $N_D \xrightarrow{M} N_S$.

3. For each $N_D \in D$ found in step 2, construct the attribute-value pair $(A, N_D(content))$ by associating the content of the block $N_D(content)$ with the attribute A.

Figure 4.16 illustrates the procedure. The associations between the attributes of the conceptual structure of the document and the blocks in the document can be determined using the mapping between the effective matching nodes of the sample tree and the basic nodes of the L-S-Tree of the document. The dotted lines represent the mapping lines between the L-S-Tree and the sample tree and the solid lines represent the associations between the blocks in the document and the basic nodes in the L-S-Tree of the document, and the associations between the nodes of dynamic type in the sample tree and the attributes in the conceptual structure of the document.

Based on the association between the attribute of the conceptual structure of the document class and the contents of blocks of the document, the next step is to instantiate the attributes by the the content of associated blocks.

4.5.2 Instantiation of Attributes of Conceptual Structure

Based on the associations between the attributes of the conceptual structure and the contents of blocks of the document, we instantiate the attributes by assigning them with the values extracted from the contents of the associated blocks. The atomic attribute can be instantiated by assigning the associated content of the block of the document as its value. For instance, in Figure 4.16, "CIS Qualifying Examination" is assigned as the value of the attribute "Subject". The instantiation of composite attributes such as "Receiver (Name, Title)", "Sender (Name, Title)" and "Date (Year, Month, Day)" is slightly complicated and requires further analysis of the contents of their associated blocks using composite pattern.

Let CP(A) be the composite pattern of attribute A for the content S of a block. The composite attribute A is instantiated by finding a sub-pattern of CP(A) such
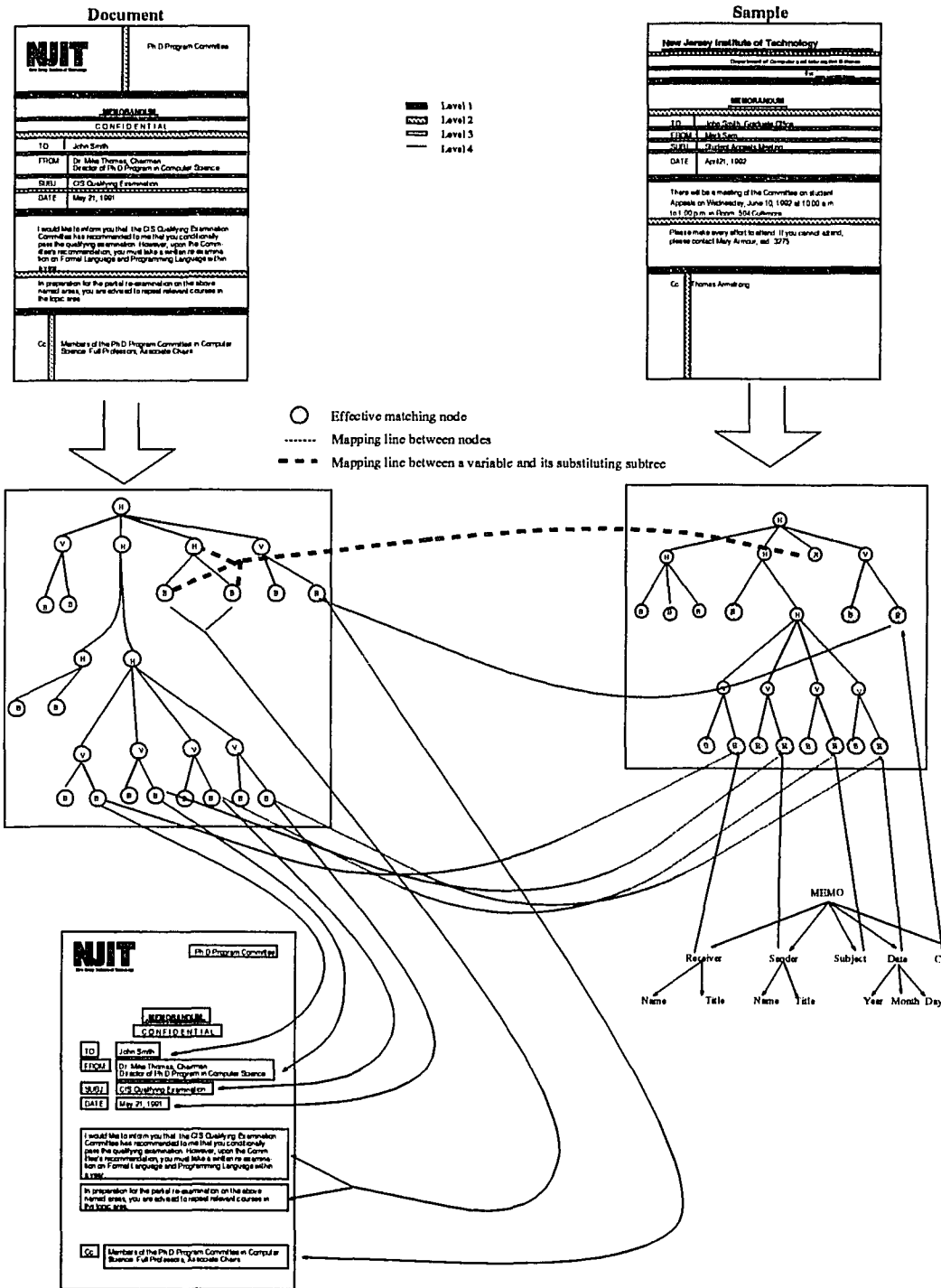
Figure 4.16 Illustration of the associations between attributes and blocks

Receiver	Name	John Smith
	Title	none
Sender	Name	Dr. Mike Thomas
	Title	Chairman Director of Ph.D Program in Computer Science
Subject	CIS Qualifying Examination	
Date	Year	1991
	Month	May
	Day	21

Figure 4.17 Frame instance of structured part of the QE memo

that all the variables in this sub-pattern are instantiated by sub-strings of S; and the result is a string equivalent to the original content S.

As an example, consider the content of a block "May 21, 1991" in the memo in Figure 4.16 and the composite attribute "Date" including three sub-attributes "Year", "Month" and "Day". After completing the process described in the previous subsection, "Date" is associated with "May 21, 1991". The composite pattern of "Date" includes three sub-patterns: (Month$SPACE^R$Day,$SPACE^R$Year), (Month/Day/Year), and (Year$SPACE^R$Month$SPACE^R$Day). The sub-pattern (Month$SPACE^R$Day,$SPACE^R$Year) is used to instantiate the attribute "Date". After the variables **Year, Month** and **Day** are instantiated by by filling the substrings "1991", "May" and "21" of "May 21, 1991", the sub-pattern yields a string equivalent to "May 21, 1991" provided that the repeated symbols match one or more the same symbols. The result of instantiating the attribute "Date" is (Date, ((Year, "1991"), (Month, "May"), (Day, "21"))).

Figure 4.17 shows the part of the frame instance as the result of information extraction from the structured part of the Q.E. memo given in Figure 1.1.

CHAPTER 5

CONTENT ANALYSIS — ANALYSIS FOR UNSTRUCTURED PART OF THE DOCUMENTS

In the previous chapter, we discussed how information can be extracted from the structured part of a document. In this chapter, we shall discuss how to extract information from the unstructured part of a document which consists of free text.

After the document is scanned through the scanner, the document which is in the form of image is converted into an encoded document. Then the encoded document is converted into a tree structure (L-S-Tree) by nested segmentation procedure. The unstructured part of the document is represented by a subtree of the L-S-Tree. Each leaf node of this subtree corresponds to a block in the unstructured part of the encoded document. Because the nested segmentation procedure divides the document into segments based on the line spacing scale, each leaf node of this subtree actually corresponds to a paragraph in the unstructured part of the document. And each paragraph is in the form of free text, which is a sequence of characters with arbitrary length.

Figure 5.1 shows the procedure of the unstructured information extraction. The procedure includes sentence classification, syntactical analysis, and heuristic thematic analysis. Before we discuss them in detail, we introduce the thesaurus which is used through the procedure of the unstructured information extraction.

5.1 Thesaurus

A thesaurus is maintained to facilitate the information extraction. The thesaurus contains two types of information about phrases. One is about synonyms. The other is about the "is a kind of" relationship between the phrases. In previous chapter,

78

Free Text

SENTENCE
CLASSFICATION

Relative Sentences

SYNTACTICAL
ANALYSIS

Parse Tree

HEURISTIC
THEMATICAL
ANALYSIS

Frame Intance for Unstructured Inforamtion
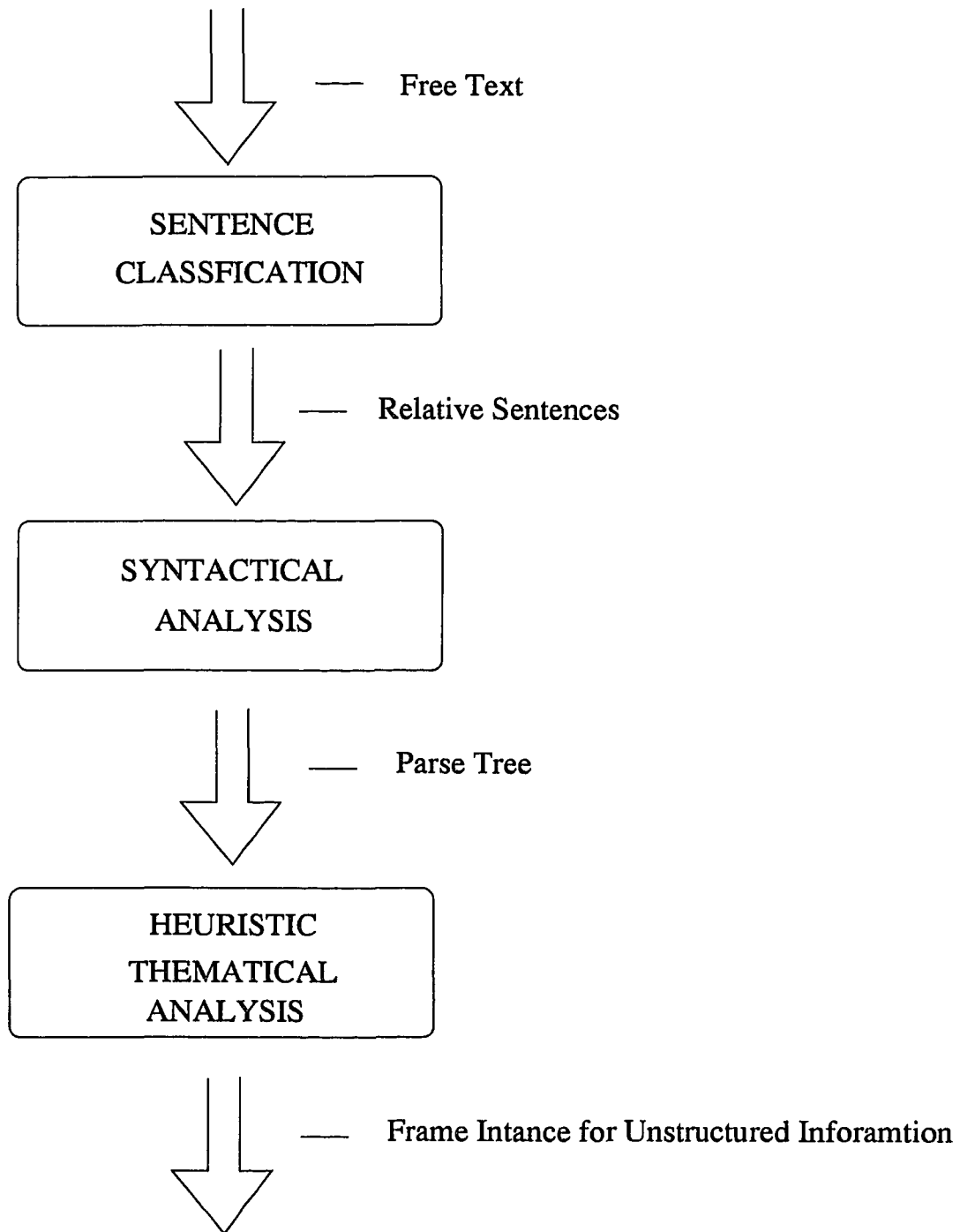
Figure 5.1 Procedure of the unstructured information extraction

we have illustrated that synonyms (semantical equivalence) can facilitate information extraction from the structured part of a document. In this section, the organization of thesaurus is discussed and in the next section, we will discuss how the "is a kind of" relationship between the phrases can help the information extraction from the unstructured part of a document.

The basic element of the thesaurus is a *word group* which is either a word or a phrase. The thesaurus, denoted as TS, is organized into a set of concept classes:

$TS = \{CC_1, CC_2, ..., CC_n\}$ where CC_i $(1 \leq i \leq n)$ is a concept class.

A concept class is a hierarchy of concept nodes. Therefore, a concept class is also called a concept hierarchy. Formally, A concept node, denoted as CN, is a set of word groups: $CN = \{WG_1, WG_2, ..., WG_k\}$ where WG_i $(1 \leq i \leq k)$ is a word group. And, $\forall i, j (1 \leq i, j \leq k), WG_i$ and WG_j are synonymous. Two word groups are said to be synonymous if they can be interchanged in certain context without changing the meaning of the statement. Each word group in the thesaurus is also called an *entry* of thesaurus.

For any concept nodes CN and CN' of the same concept class CC, let CN be $\{WG_1, WG_2, ..., WG_m\}$ and CN' be $\{WG'_1, WG'_2, ..., WG'_n\}$, CN is a child of CN' in CC if $\exists i, j \ WG_i$ "is a kind of" WG'_j.

For example, the Figure 5.2 shows a portion of concept class regarding to the word "course".

After introducing the organization of the thesaurus, we give the definitions of *concept, semantical equivalence, sense* and *instance*.

Definition 21 *(Concept)*

A *concept* C appearing in a free text T, which is a sequence of characters with arbitrary length, is defined as a subsequence of T such that :

1. C is a word group in the thesaurus, i.e., C is an entry of thesaurus; and

····

{ Course (8)}

{lecture course (1)}   {seminar (1)}

...   ······

{Programming Language (1), CIS 632 (1)}   {Formal Lanuage (1), CIS 633 (1)}
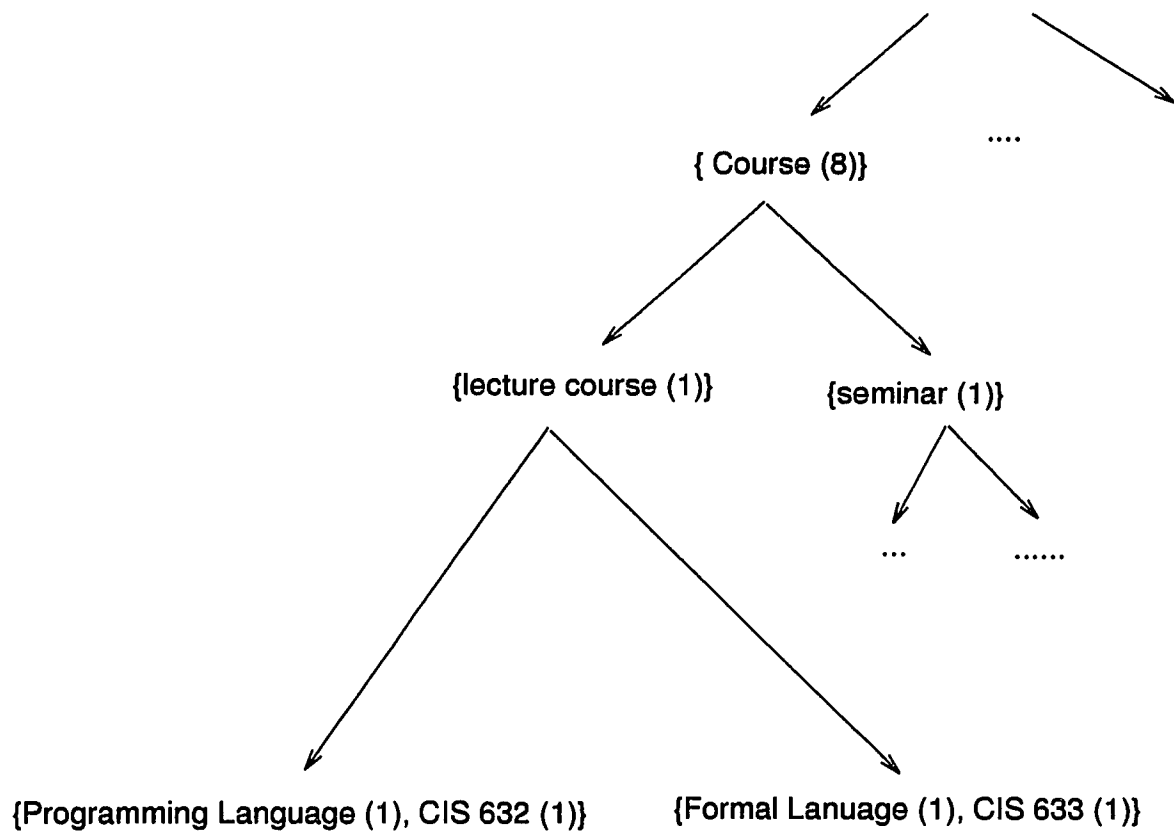
Figure 5.2 Illustration of hierarchy of concept class for "course"

2. there is no other word group in T contains C.

For example, in the sentence *Every student must take at least three lecture courses in one semester*, the word groups "course" and "lecture course" are both in the thesaurus, i.e., they are both entries of the thesaurus. But according to the definition of concept, only "lecture course" is a concept in this sentence.

Definition 22 *(Semantical Equivalence)*

Two concepts C_1 and C_2 are said to be *semantical equivalence*, denoted by $C_1 == C_2$, if they are in the same concept nodes of the thesaurus. In other words, they are semantical equivalent if they are synonymous.

Notice that a word group may represent several different concepts. In other words, the same word group may appear multiple times as entries in the thesaurus. That is, a word group may present different meanings in different context. Each of these meanings is called a sense of this word group.

Definition 23 *(Sense)*

For a given word group, each of its occurrences in the thesaurus is called one of its senses.

For example, the following are some of the senses of the word group "time":

sense 1 A sufficient period of time; e.g., "I didn't have time to finish."

sense 2 A suitable moment; e.g., "it is time to go."

sense 3 Fourth dimension; a measurement.

sense 4 Clock time;

sense 5 Time as age; e.g., "he was a great actor is his time."

sense 6 An instance or occasion for some event; e.g., "This time he succeeded."

sense 7 An person's experience on a particular occasion; e.g., "he had a time holding back the tears."

sense 8 Time as meter;

sense 9 The continuum of experience in which events pass from the future through the present to the past.

A sense number is assigned to each sense of a word group. In Figure 5.2, the number besides each word group is its sense number.

Definition 24 *(Instance)*

Given two word groups WG_1 and WG_2, where WG_1 is in concept node CN_1 and WG_2 is in concept node CN_2, WG_2 is called an *instance* of WG_1 if

- CN_1 and CN_2 are of the same concept class; and

- CN_2 is one of the descendant nodes of CN_1 in this concept class.

Figure 5.3 illustrates some word groups of which the word "time" is the instance, and Figure 5.4 shows all the instances of the sense 4 of the word "time".

5.2 Content Structure

In Chapter 1, we mentioned that the content of a document can be divided into structured and unstructured parts. And the content structure is used to facilitate the instantiation of the attributes of the frame template from the *unstructured part* of a document. In this section, we will give the representation of the content structure and show how to use the content structure to facilitate the instantiation of the attributes of the frame template from the *unstructured part* of a document.
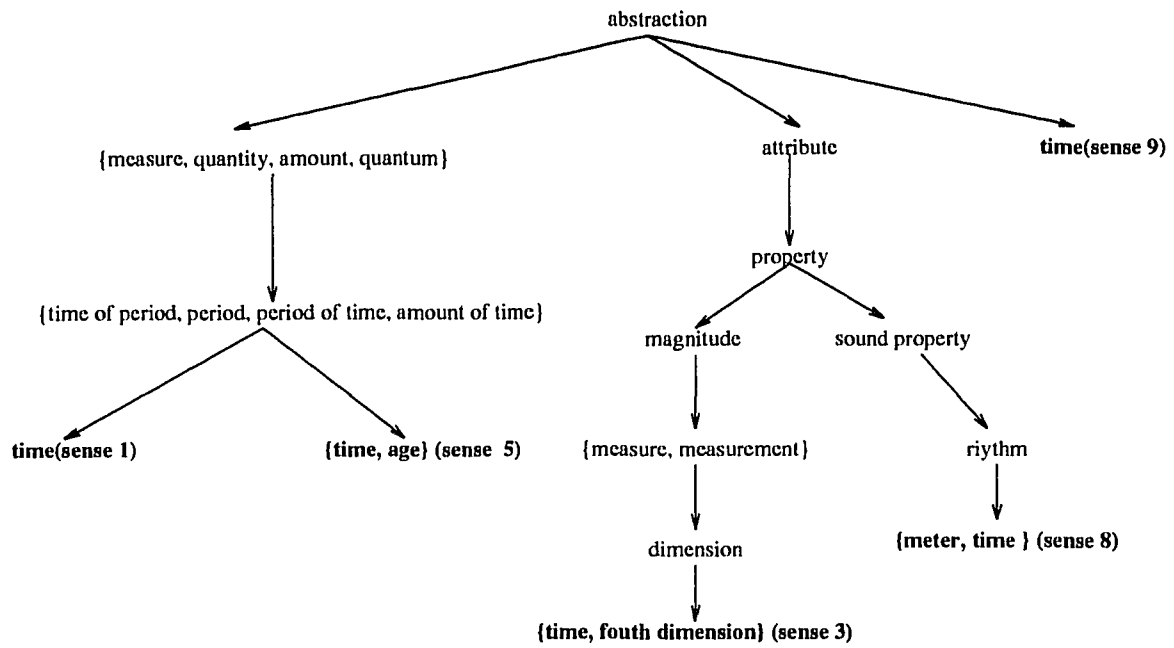
Figure 5.3 Some word groups of which "time" is the instance
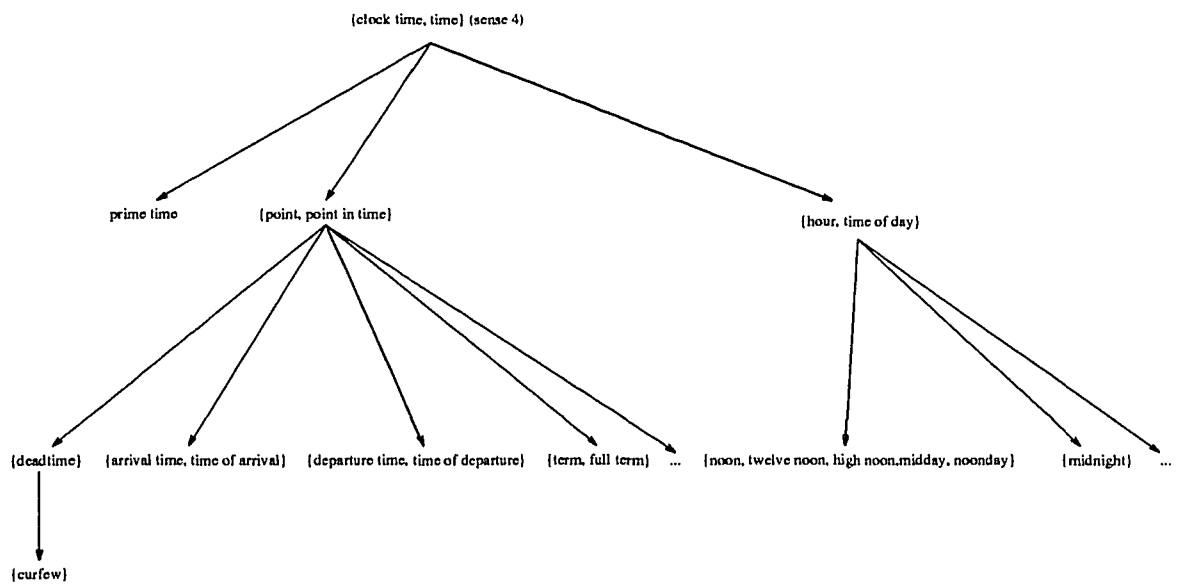
Figure 5.4 All instances of the sense 4 of word "time"

The content structure is represented by an activation condition and a set of attribute descriptors. The activation condition specifies under what condition the content structure is used as the knowledge to extract information from the unstructured part of a document. And each attribute descriptor specifies the properties of the values that may assign to the attributes to obtain the frame instance. Formally, the content structure, denoted by $CONT\_S$, is represented as

$$(AC, \{UAD_1, UAD_2, ..., UAD_m\})$$

where

- AC is called *activation condition* which describes the conditions where the $CONT\_S$ is chosen for extracting the information from the unstructured part of the document. It is represented as

  $(AN_1, KT\_List_1), (AN_2, KT\_List_2), ..., (AN_n, KT\_List_n)$ where AN_i ($1 \leq i \leq n$) is an attribute name in the corresponding frame template whose value is extracted from the structured part of the document and KT_List_i ($1 \leq i \leq n$) is a list of key terms which can be part of the value of this attribute.

- UAD_i($1 \leq i \leq m$) is a *unstructured part attribute descriptor* and is composed of

  an attribute name which specifies the name of this attribute, denoted as $UAD_i(attr\_name)$.

  an attribute domain which specifies the restrictions on values that may assign to this attribute, denoted as $UAD_i(attr\_domain)$ and is composed of:

    sense which specifies the sense number of the $UAD_i(attr\_name)$ in the thesaurus, denoted by $UAD_i(attr\_domain(sense))$.

thematic role which specifies the expected thematic role of $UAD_i(attr\_name)$, denoted by $UAD_i(attr\_domain(t\_role))$. The detail of thematic role will be discussed in Section 5.6

restrictions which are a set of rules governing the extraction of $UAD_i(attr\_name)$ from the sentences containing the values for $UAD_i(attr\_name)$, denoted by $UAD_i(attr\_domain(restriction))$.

For example, Figure 5.5 shows the content structures for the document type "QE Memo" and 'Meeting Memo".

5.3   Selection of Content Structures

Given a document, its corresponding conceptual structure is first identified by the conceptual analysis discussed in previous chapter. The information of its structured part can be extracted through the use of the conceptual structure of the document.

In order to extract the information from the unstructured part of the document, its content structure has to be identified. The content structure is selected by evaluating the "activation condition" of each content structure stored in the system based on the information extracted from the structured part of the document. A content structure is chosen if the value of each attribute of frame instance specified in the activation condition of the content structure contains at least one of the specified key terms

As an example, consider the document in Figure 5.6. After the document is classified as a memo, its conceptual structure is identified. Based on its conceptual structure, the information of the structured part of the document is extracted to form the part of a frame instance.

Since the value of "Subject" contains "Qualifying Examination" which is specified as one of the key terms in the AC (activation condition) of $CONT\_S(QE\ memo)$,

CONT_S (QE_Memo)			
AC	(Subject, {"qualifying examination", "QE"})		
UAD1	attr_name	QE Result	
	attr_domain	Sense	2
		t_role	action
		restrictions	thematic object contains "qualifying examination"
UAD2	attr_name	Courses Retaken	
	attr_domain	Sense	8
		t_role	topic
		restrictions	thematic object contains "qualifying examination"

CONT_S (Meeting_Memo)			
AC	(Subject, {"meeting"})		
UAD1	attr_name	Meeting Date	
	attr_domain	Sense	5
		t_role	date
		restrictions	thematic object contains "meeting"
UAD2	attr_name	Meeting Time	
	attr_domain	Sense	4
		t_role	time
		restrictions	thematic object contains "meeting"
UAD3	attr_name	Meeting Locaation	
	attr_domain	Sense	1
		t_role	location
		restrictions	thematic object contains "meeting"

Figure 5.5 The content structures for "QE Memo" and "Meeting Memo"

NJIT

New Jersey Institute of Technology

Ph.D Program Committee

MEMORANDUM

CONFIDENTIAL

TO: John Smith

FROM: Dr. Mike Thomas, Chairman
        Director of Ph.D Program in Computer Science

SUBJ: CIS Qualifying Examination

DATE: May 21, 1991

I would like to inform you that the CIS Quality Examination Committee has recommended to me that you conditionally pass the qualifying examination. However, upon the Committee's recommendation, you must take a written re-examination on Formal Language and Programming Language within a year.

In preparation for the partial re-examination on the above named areas, you are advised to repeat relevant courses in the topic area.

Cc: Members of the Ph.D Program Committee in Computer Science. Full Professors, Associate Chairs.

Receiver	Name	John Smith
	Title	none
Sender	Name	Dr. Mike Thomas
	Title	Chairman
		Director of Ph.D Program in Computer Science
Subject	CIS Qualifying Examination	
Date	Year	1991
	Month	May
	Day	21
CC	Members of the Ph.D Program Committee in Computer Science. Full Professors, Associate Chairs	

Figure 5.6 A QE memo and its structured part portion of frame instance

AC	(Subject, {qualifying examination, QE})	
UAD1	attr_name	QE Result	
	attr_domain	Sense	2
		restrictions	(Thematic_role = Action) AND (Thematic object contains " examination")
UAD2	attr_name	Courses Retaken	
	attr_domain	Sense	8
		restrictions	Thematic object contains "re-examinination"

AC	(Subject, {meeting})	
UAD1	attr_name	Meeting Date	
	attr_domain	Sense	5
		restrictions	thematic object contains "meeting"
UAD2	attr_name	Meeting Time	
	attr_domain	Sense	4
		restrictions	thematic object contains "meeting"
UAD3	attr_name	Meeting Location	
	attr_domain	Sense	1
		restrictions	thematic object contains "meeting"

Receiver	Name	John Smith
	Title	
Sender	Name	Mile Thomas
	Title	Chairman, Director of Ph.D Program in Computer Science
Subject	CIS Qualifying Examination	
Date	Year	1991
	Month	May
	Day	21
Cc	Members of the Ph.D Program Commitee in Computer Science. Full Professors, Associate Chairs.	

Figure 5.7 Selection of content structure

$CONT\_S(QE\ memo)$ is chosen as the content structure to extract information from the unstructured part of the document. In other words, the information expected to extract from the unstructured part of this QE memo are "QE result" and "Course retaken". Figure 5.7 illustrates the content structure selection.

5.4 Sentence Classification

As mentioned in section 5.1, the unstructured part of a document is in the form of free text. The task of sentence classification is to extract the sentences from the free text which is relevant to the user's concerns represented by the attributes of

the frame template. We observed that these sentences usually contain the words or phrases which are conceptually relevant with the user's concerns and can potentially be the values of the attributes of the corresponding frame instance. Based on the above observation, a sentence is classified as conceptual relevant by identifying these words or phrases. The first step of the procedure of sentence classification is sentence segmentation.

5.4.1 Sentence Segmentation

As mentioned before, the input of sentence classification is a free text consisting of a sequence of characters with arbitrary length. In order to classify the sentences in the free text , the sentence segmentation is needed to separate the free text into a set of sentences. Superficially, a sentence is a subsequence of the free text, which is ended by a period and spaces. However, this rule becomes ambiguous if the abbreviation of a word appears in a sentence. Consider the following text:

Please make every effort to attend the meeting. If you cannot attend, please contact Mary Armout, Ext. 5889.

The sentences contained in the above text are:

1. Please make every effort to attend the meeting.

2. If you cannot attend, please contact Mary Armon, Ext. 5889.

If the period and spaces were the only delimiters used to separate the text, then the sentences would be:

1. Please make every effort to attend the meeting.

2. If you cannot attend, please contact Mary Armon, Ext.

3. 5889.

Sentence Segmentation (T)
/* T is a free text corresponding to one paragraph in the unstructured part of
the document.*/
begin
while NOT end_of_(T)
   begin
   read one character c from T;
   if need to skip c **then** skip the c and read another character
     else if c does not mark the end of a word **then**
       continue to read another character
        else begin
           put the word scanned into the word buffer which contains
           the words obtained for the current sentence;
           if c does not mark the end of sentence
              then continue to obtain the word for the current sentence
                else write out the word buffer which is corresponding
                  to one sentence of T
        end
  end /* while */
end /* Sentence Segmentation */

Figure 5.8 Algorithm of sentence segmentation

The example shows that the abbreviations appearing in the free text need to be taken care of in order to resolve the ambiguity. A list of abbreviation words is maintained for the purpose of identifying the abbreviations in the sentence. The algorithm of sentence segmentation is given in Figure 5.8:

In the algorithm, the rules for deciding if skip is needed are :

- a space is followed after a space, or

- an end_of_line character is followed after another end_of_line character; or

- an end_of_line character is followed after a space.

The first rule is used to skip multiple spaces between the words in the sentence. The second rule is used to skip empty lines. The third rule is used to skip the indent in the paragraph.

The rules for deciding if the currently scanned character is the end of a word are:

- the currently scanned character is a space, or

- the currently scanned character is a end_of_line.

The rules for deciding if the currently scanned character is the end of a sentence are:

- the currently scanned character is a period, and

- the word previously put into the word buffer is not in the list of the abbreviation words.

After each sentence is segmented into words, the words are indexed. In the process of word indexing, all the articles, pronouns, prepositions, conjunctions and auxiliary verbs are tagged so that later on they will not be considered in the sentence classification because these words do not form concepts.

Figure 5.9 illustrates the sentence segmentation and word indexing for the unstructured part of the document in Figure 5.6.

5.4.2   The Procedure of Sentence Classification

In this subsection, we give the algorithm of sentence classification. The Algorithm is show in Figure 5.10. And Figure 5.11 illustrates the algorithm of sentence classification for a sentence of the unstructured part of the document in Figure 5.6.

Note that in the algorithm, if $UAD(attr\_name)$ is not an entry of thesaurus, i.e., it is not a concept, then the concept in $UAD(attr\_name)$ will be used to classify the

I would like to inform you that the CIS Qualifying Examination
Committee has recommended to me that you conditionally
pass the qualifying examination. However, upon the Comm-
ittee's recommendation, you must take a written re-examina-
tion on Formal Language and Programming Language within
a year.

In preparation for the partial re-examination on the above
named areas, you are advised to repeat relevant courses in
the topic area.

Sentence Segmentation

1. I would like to inform you that the CIS Qualifying Examination
   Committee has recommended to me that you conditionally
   pass the qualifying examination.

2. However, upon the Committee's recommendation , you must
   take a written re-examination on Formal Language and
   Programming Language within a year.

3. In preparation for the partial re-examination on the above
   named areas, you are advised to repeat relevant courses in
   the topic area.

Word Indexing

1. {I*, would*, like, to*, inform, you*, that*,CIS, Qualifying, Examination,
   Committee, has*, recommended, to*, me*, that*, you*, conditionally,
   pass, the*, qualifying, examination }

2. {However*, upon*, the*, Committee's, recommendation , you*, must*,
   take, a*, written, re-examination, on*, Formal, Language, and*,
   Programming, Language, within*, a*, year }

3. { In*, preparation, for*, partial, re-examination, on*, the*, above*,
   named, areas, you*, are*, advised, to*, repeat, relevant, courses, in*,
   the*, topic, area }

Figure 5.9 Sentence segmentation and word indexing

Sentence Classification (T, $CONT\_S$)

/* T is a free text corresponding to the unstructured part of the document. */

/* $CONT\_S$ is the content structure of the document*/

begin

Sentence Segmentation(T)

for each sentence S in T

   begin

   Indexing for each word in S by tagging the articles, pronouns, prepositions, conjunctions and auxiliary verbs;

   for each concept C in S

      begin

      if there exists an unstructured part attribute descriptor UAD in $CONT\_S$

         where CP is an instance of $UAD(attr\_name)$ with sense specified in

         $UAD(attr\_domain(Sense))$

         then S is classified to be conceptual relevant by UAD;

            continue to classify the next sentence;

     end /* for */

   end; /* for */

All the sentences which are not classified to be conceptual relevant are classified to be conceptual irrelevant;

end /* Sentence Classification */

Figure 5.10 Algorithm of sentence classification
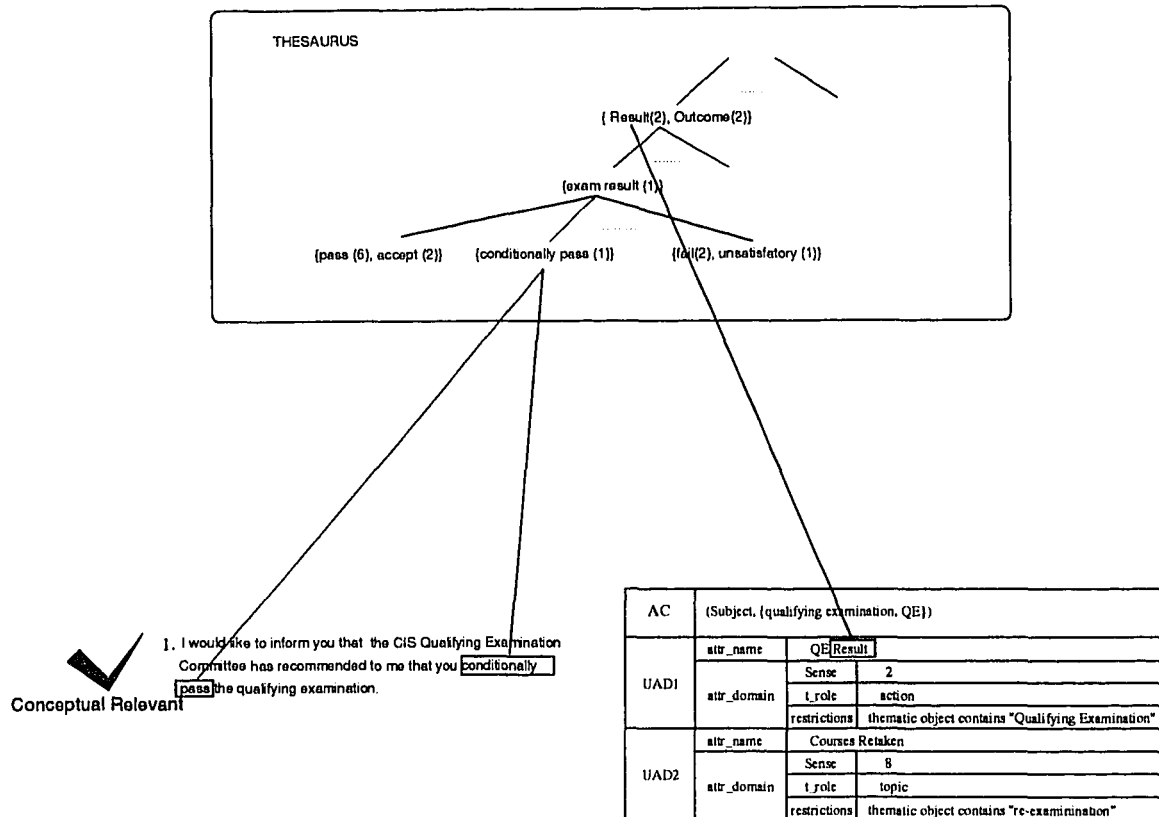
Figure 5.11 Illustration of algorithm of sentence classification

sentence. Consider the sample in Figure 5.11, "QE result" is not a concept because it is not an entry of thesaurus. Thus, the concept in "QE result" which is "result" is used to classify the sentence.

5.5 Thematic Analysis — Identification of Thematic Roles

Sentence classification classifies a sentence by checking if there exists a concept appearing in the sentence which is the instance of the attribute name of the corresponding content structure. However, a sentence classified to be conceptual relevant does not always contain the information that the user wants to extract. For example, consider the following two sentences:

1. The meeting will be held at room 4402.

2. The meeting will discuss about the usage of room 4402.

Suppose that the attribute name of the corresponding content structure is "meeting location". Because "meeting location" is not a concept of thesaurus, the "location" is used to classify the sentence. From the sentence classification, both sentences are classified to be conceptual relevant because both of them contain "room 4402" which is a instance of "location". However, the "room 4402" in the first sentence specifies the meeting location and "room 4402" in the second sentence does not.

In order to extract the information more precisely, further analysis of the thematic roles of the concepts in the sentence is needed. The way a phrase participates in describing an action of a sentence is called its *thematic role*. Each noun phrase or verb phrase has its thematic role in a sentence. For example, the sentence "Bob has passed the qualifying examination" carries information about "Bob" who is the agent performing the action of passing the "qualifying examination". The "qualifying examination" is the object to be passed, and "has passed" is the action taken by the agent.

The number of thematic roles embraced by various theories varies depending on the different domains on which they are applied [52]. The following thematic roles are commonly used in the office document domain [1]:

thematic object The thematic object is an entity upon which the action is applied. Often, the thematic object is the same as the syntactic direct object, as in "Robbie hit the *ball*." On the other hand, in passive sentence, the thematic object appears as the syntactic subject as in "*The ball* was hit by Robbie."

---

[1]Some of the thematic roles discussed here are from [53]

agent The agent is an entity that causes the action to occur. The agent is often the syntactic subject, as in *"Robbie* hit the ball." But in a passive sentence, the agent may also appear in a prepositional phrase: " The ball was hit *by Robbie."*

coagent The word *with* may introduce a noun phrase that serves as a partner to the principal agent. The two carry out the action together: "Robbie played tennis *with Suzie."*

beneficiary The beneficiary is the person for whom an action is performed: "Robbie bought the balls *for Suzie."*

action The action is performed by the agent. The action is often the verb of the sentence: "Robbie *hit* the ball."

location The location is where the action occurs. Usually the location is appeared as a prepositional phrase in the sentence: "Robbie and Suzie studied *in the library, at a desk, by the wall, under a picture, near the door."*

date The date specifies the date when the actions occurs. Prepositions such as *on* usually introduce noun phrases serving as date role filler, as in " Robbie is going to Chicago *on Nov. 26, 1993, on Friday."*

time Time specifies when the action occurs. Prepositions such as *at, before*, and *after* introduce noun phrase serving as time role filler, as in " Robbie and Suzie left *before noon, at 8 am."*

duration Duration specifies how long the action takes. Prepositions such as *for* indicate duration. "Robbie and Suzie jogged *for an hour."*

topic Topic specifies the possible domain of the object. Prepositions such as *about, on*, and *of* often indicate topic. "Robbie and Suzie are discussing the problem *about programming language."*

instrument The instrument is a tool used by the agent to perform the action. The preposition *with* typically introduces instrument noun phrases : "Robbie hit a ball *with a racket.*"

source and destination The source describes the initial position of the agent or thematic object, and the destination describes the final position: "Robbie went *from the dining room to the kitchen.*"

conveyance The conveyance is something in which or on which one travels: "Robbie always goes *by train.*"

Consider the sentence "Tom will attend a meeting about computer resources in the CIS Conference Room from 3:00 pm to 4:00 pm on September 12, 1993.". The word "meeting" is the thematic object which is the major concern of the sentence. The word "Tom" is the agent. The phrase "will attend" is the action taken by Tom. The phrase "computer resources" is the topic of the thematic object. The phrase "3:00 to 4:00" specifies the time and duration. The phrase "CIS Conference Room" specifies the location. And the phrase "September 12, 1993" specifies the date.

In the thematic analysis, syntactic analysis can be used to facilitate the identification of thematic roles of words and phrases.

5.5.1 Syntactic Analysis

A *parser* is a syntactic analyzer, which consists of the following two parts: a body of syntactic knowledge for specifying the sentences allowed in the language called *grammar*, and a procedure for using the knowledge called *interpreter*.

The most influential theory of grammar is the theory of *formal language* introduced by Noam Chomsky in the 1950s. Within the theory of formal language, Chomsky defined four types of grammars, namely, *non-restricted, context sensitive, context-free* and *regular grammars.*

There are numerous parsing algorithms for determining whether a given string is an element of the language. Among them, the context-free grammars are widely used as the basic description of natural language grammars. However, since the natural language is not context-free, some types of grammars have also been investigated. Examples are the *transformational grammar* introduced by Noam Chomsky [4], the *systematic grammar* developed by Michael Halliday [9], and the *augmented transition networks (ATNs)* introduced by William Woods as a versatile representation of grammars for natural languages [55].

These grammars can be interpreted by various strategies. For example, transformational grammar and systematic grammar can be interpreted by top-down or bottom-up processing. The interpreting rules of ATNs are explicitly specified in the representation of ATNs.

The result of parsing a sentence is usually represented by a *parse tree*, a tree describing the syntactic structure of the sentence. For example, a parse tree is shown in Figure 5.12.

Using the syntactic analysis, a sentence is decomposed into phrases. Each phrase has its own type according to the properties of the words in the phrase (e.g., noun phrase, verb phrase, prepositional phrase). In general, every phrase is constructed by a noun phrase or a verb phrase. For example, the prepositional phrase is composed of a preposition and a noun phrase. The syntactic information of the sentence obtained through the syntactic analysis will help to identify the thematic roles of phrases of the sentence.

5.5.2 Heuristics for Identifying the Thematic Roles

Based on the results of syntactic analysis, several heuristic strategies for identifying the thematic role of a phrase are:

I would like to inform you that the CIS Qualifying Examination
Committee has recommended to me that you conditionally
pass the qualifying examination.

```
                    I would like to inform you that the CIS Qualifying Examination
                    Committee has recommended to me that you conditionally
                    pass the qualifying examination.

    SUBJECT         VERB              DO              IO (Clause )

      I        would like    IP        you      SUBJECT   VERB    DO    IO (Clause)

                          to    inform

                                       CIS Qualifying Examination Committee  has recommended to   me

                                                                SUBJECT      VERB        DO

                                                                  you   conditionally pass  the qualifying examiniation
```

IP : Indefinite Phrase
DO : Direct Object
ID : Indirect Object
PP: : Prepostional Phrase

Figure 5.12 A parse tree

Preposition	Allowable thematic role
by	agent or conveyance or location
with	coagent or instrument
for	beneficiary or duration
from	source
to	destination

Figure 5.13 Table of relations between the prepositions and thematic roles

1. In a sentence, each verb could give a hint about what thematic roles can appear in the sentence and where the noun phrases assuming those thematic roles.

2. The preposition limits the possibilities of the thematic roles of a noun phrase. Figure 5.13 lists the relations between prepositions and their possible thematic roles.

3. The noun phrase itself may limit its possible thematic role identifications. Consider the following two sentences: "Robbie was sent to the scrap heap by parcel post," and "Robbie was sent to the scrap heap by Marcel Proust." The parcel post is more likely to be a conveyance, whereas Marcel Proust is more likely to be an agent.

4. For most thematic roles, only one filler of a thematic role in a sentence is allowed. That is, no two noun phrases of a sentence have the same thematic roles. Thus, the identification of the thematic role of a noun phrase will help identification of the thematic roles of other noun phrases.

The steps for determining the thematic role of each phrase in a sentence are:

1. Obtain the possible meanings of the verb from the dictionary. Discard those meanings of the verb that are inconsistent with the verb's particle;

2. Find the thematic object among the noun phrases without a preceding preposition;

3. Discard the meanings of the verb from the dictionary that are inconsistent with the thematic object found in step 2;

4. For each remaining noun phrase, determine its thematic role with the help of the prepositional restrictions and the meaning of the noun;

5. Discard the meanings of the verb from the dictionary that are inconsistent with the identified thematic roles of noun phrases.

To illustrate the above steps, consider the following sentence: *Smith took the examination answers to John.* In step 1, the possible meanings of the verb "take" is:

1. *Take* means transport. Either a source or a destination or both should appear.

2. *Take* means swindle. The source and destination roles are absent when this meaning is intended. Only people can be swindled.

3. *Take* means to swallow medicine.

4. *Take* means to steal. People cannot be stolen.

5. *Take* means to initiate and execute a social event with another person. The particle *out* is always used.

6. *Take* means to remove. The particle *out* is always used. People cannot be removed.

7. *Take* means to assume control. The particle *over* signals this meaning.

8. *Take* means to remove from the body. The particle *off* is always used.

The meaning No. 5 of *take* to No. 8 are eliminated because there is no particle following the *take* in the sentence. In step 2, the noun phrases without propositions preceding them are *Smith* and *the examination answers*. Because the sentence is not a passive sentence, the syntactic object *the examination answers* is identified to be the thematic object. In step 3, the meaning No. 2 of *take* is discarded because the thematic object is not a instance of people. And No. 3 is discarded because the thematic object is not a instance of medicine. In step 4, by consulting the relations between prepositions and thematic roles in Figure 5.13, the thematic role of noun

phrase *John* is identified to be the destination. In step 5, the meaning No. 1 of *take* is determined based on the result of step 4.

5.6   Information Extraction Based on the Content Structure

For a given document D, after the information of its structured part is extracted, the information of its unstructured part T can be extracted in following steps:

1. Select an appropriate content structure $CONT\_S$ by evaluating each $CONT\_S$'s activation condition AC based on the information extracted from the structured part of D (see Section 5.3).

2. Divide the unstructured part T into sentences $\{S_1, S_2, ..., S_n\}$ and index the words in each S_i $(1 \leq i \leq n)$ (see Section 5.4.1).

3. Based on the content structure $CONT\_S$ selected in step 1, classify all the sentences into conceptual relevant and conceptual irrelevant. Let the conceptual relevant sentences are $\{RS_1, RS_2, ..., RS_m\}$ (see Section 5.4.2).

4. For each RS_j $(1 \leq j \leq m)$,

   (a) perform the syntactic analysis on RS_j using the parser to obtain a corresponding parse tree (see Section 5.5.1);

   (b) perform the thematic analysis on RS_j by finding the thematic role for each noun phrase and verb in RS_j using the heuristic strategies discussed in Section 5.5.2;

   (c) for the unstructured part attribute descriptor UAD in the $CONT\_S$ which classifies the RS_j to be conceptual relevant, evaluate $UAD(attr\_domain(restriction))$. If $UAD(attr\_domain(restriction))$ is true, find the phrase whose thematic role is the same as the one specified in the $UAD(attr\_domain(t\_role))$ and assign the phrase as the value of $UAD(attr\_name)$.

For example, let us consider the document in Figure 5.6 again. Figure 5.14 illustrates the process. After the syntactic and semantic analyses, the thematic role of each phrase of every sentence is obtained. For the sentence "you conditionally pass the qualifying examination", the thematic roles of phrases are:

- agent — you;

- action — conditionally pass;

- thematic object of the sentence — qualifying examination;

During the sentence classification, this sentence is classified to be conceptual relevant because "conditionally pass" is a instance of "result" [2]. Thus, this sentence is classified by the unstructured attribute descriptor UAD_1 in the content structure $CONT\_S$ (see Figure 5.14).

Because the thematic role of "conditionally pass" is action and the thematic object of this sentence ("qualifying examination") contains the word "examination", $UAD_1(attr\_domain(restriction))$ is true. Therefore, "conditionally pass" is extracted as the value of the $UAD_1(attr\_name)$ ("QE result"). Likewise, the sentence "you must take a written re-examination on Formal Languages and Programming Languages within a year" can be handled in the same way.

The part of the frame instance as the result of information extraction from the unstructured part of the QE memo is shown in Figure 5.15. By combining the extractions from the structured part and the unstructured part of the document, a complete frame instance is obtained. The complete frame instance of the "QE memo" is shown in Figure 5.16

---

[2]Because "QE result" is not an entry in thesaurus, "result" which is the concept in "QE result" is used to classify the sentence.
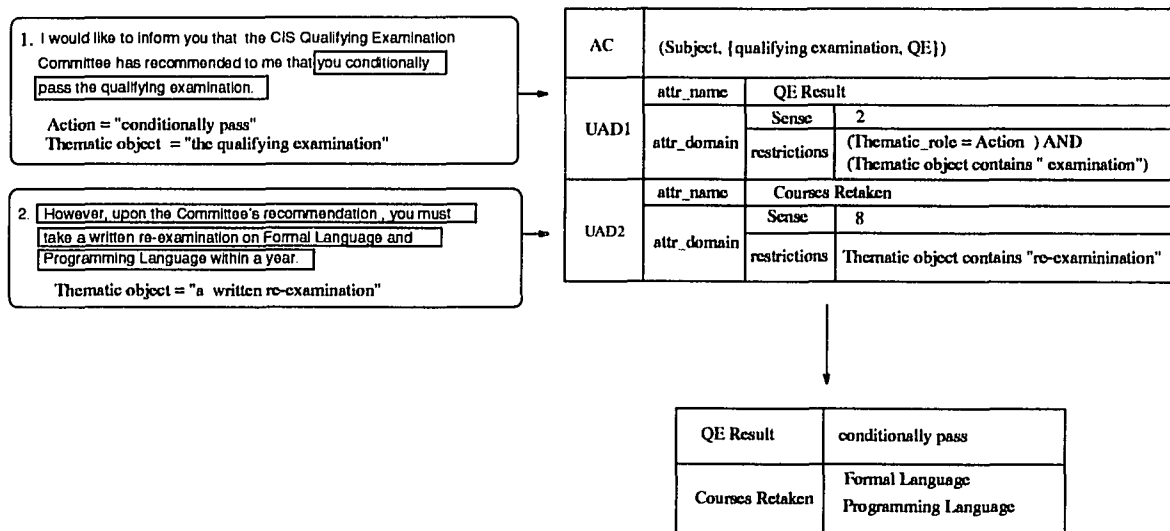
**Figure 5.14** Information extraction based on content structure

QE Result	conditionally pass
Course retaken	Formal Language and Programming Language

Figure 5.15 The part of the frame instance for unstructured part of the QE memo

Receiver	Name	John Smith
	Title	None
Sender	Name	Mile Thomas
	Title	Chairman, Director of Ph.D Program in Computer Science
Subject	CIS Qualify Examination	
Date	Year	1991
	Month	May
	Day	21
Cc	Members of the Ph.D Program Commitee in Computer Science. Full Professors, Associate Chairs.	
QE result	contionally pass	
Courses retaken	Formal Language and Programming Language	

Figure 5.16 The complete frame instance of the QE memo

CHAPTER 6

SUMMARY AND FUTURE RESEARCH

In this chapter we summarize what has been discussed in this thesis and then we present an outlook for future research.

6.1 Summary

The intent of this thesis is to present the design of a subsystem of TEXPROS which is used for classifying various office documents and extracting the useful information for the user. The system employs layout, conceptual and content analyses in document classification and information extraction. The layout structure of a document is represented by an ordered labeled tree (called the L-S-Tree). This tree structure is obtained by segmenting the document in a nested fashion based on the line spacings of the document. The conceptual structure of the document is identified by finding a document sample pre-stored in the document sample base based on the layout similarities (edit distance) and conceptual similarities (conceptual closeness degree) between the document and the document sample. The layout comparison between the document and the document sample is accomplished by applying the approximate tree matching technique to the L-S-Tree of the document and sample tree of the document sample. The conceptual comparison between the document and the document sample is accomplished by calculating the conceptual closeness degree based on the number of effective matching nodes in the sample tree.

After the conceptual structure of the document is identified, the super type of the document is identified and the conceptual structure is used to extract the information from the structured part of the document. The extraction begins by finding each block of the document which is associated with one of the attributes in

107

the corresponding conceptual structure based on the mapping found in the matching between the document and document sample. And the attributes of atomic type are instantiated by the content of the associated blocks. The attributes of composite type are instantiated based on the composite patterns specified in the conceptual structure.

Based on the information extracted from the structured part of the document, the content structure of the document is identified by evaluating the activation condition of each content structure stored in the system. After the content structure of the document is identified, the type of the document is identified and the content structure is used to extract the information from the unstructured part of the document. The extraction begins by classifying the sentences of the unstructured part of the document into conceptual relevant and irrelevant. The sentence classification is based on the conceptual relationship between the concepts in the sentence and the attribute name of the content structure by consulting the thesaurus. Then the thematic analysis is applied to the conceptual relevant sentences to instantiate the attributes of the content structure. By combining the instantiations of the attributes of the conceptual structure and the content structure, the complete frame instance of the document is obtained.

6.2   Future Work

The system discussed in this thesis successfully classifies various office documents and extracts information from these documents. However, there also exist some limitations. In the following, we will discuss these limitations and future work to resolve them:

- The layout analysis of our approach uses the nested segmentation to capture the layout characteristics of a office document. The document is segmented nestedly based on the different line spacing scales used in the document. The

segmented document is represented by a L-S-Tree. So far, the proposed nested segmentation technique can only analyze single page documents. This technique needs to be extended so that multi-page documents can be processed. One of the possible approaches is to concatenate all the pages together to form a virtual one page document so that the current nested segmentation technique can be used. However, when two pages are concatenated into one page, it is not trivial to determine the line spacing between the block residing at the bottom of the first page and the block residing at the top of the second page. We observed that in order to give a reasonable line spacing, several factors including the format information (i.e., indent of the paragraph, etc.) and sometimes the semantical meaning of the contents of these two blocks needed to be considered. How the segmentation technique can be extended to deal with multi-page document remains to be a future research issue.

- In our system, the sample-based approach is used for the conceptual analysis of the document. In order to find an appropriate sample in the sample base to process the incoming document, we compare the incoming document with the samples one by one until we try out all the samples in sample base. In terms of efficiency, this is not a very good approach. It would be better if we first search the incoming document for some common features inferred from a group of samples and then compare the document only with the samples of this group. How the common features of a group of samples can be inferred and how these features can be identified from the incoming document remains to be a future research issue.

- In our current content analysis of the unstructured part of the document, only the semantical relationship between the phrases in a sentence is considered by applying the thematic analysis. However, we do not analyze the semantical

relationship between the phrases in different sentence. In other words, we do not take context into consideration. It is not sufficient to extract information from the free text without context analysis. There exist a lot of cases that the context analysis is needed to extract information from the free text. Therefore, the context analysis, which is also one of major research issues of natural language understanding, is one of our future research issues.

REFERENCES

1. F. Barbic and F. Rabitti. "The Type Concept in Office Document Retrieval". In *Proc. of VLDB 85*, pages 34-47, Stockholm, Sweden, 1985.

2. N. Cavazza and P. Zweigenbaum. "Extracting Implicit Information from Free Text Technical Reports". *Information Processing & Management*, 28(5):609-618, 1992.

3. A. Celentano, M. G. Fugini, and S. Pozzi. "Classification and Retrieval of Documents Using Office Organization Knowledge". In *Proc. ACM Conf. on Organizational Computing Systems*, pages 159-164, Atlanta, Georgia, November 1991.

4. N. Chomsky. *Syntactic Structure*. The Hague Mouton, London, 1957.

5. N. Christofides. *Graph Theory, An Algorithmic Approach*. Academic Press, London, 1975.

6. A. Dengel and G. Barth. "ANASTASIL: A Hybrid Knowledge-Based System for Document Layout Analysis". In *Proc. IJCAI'89, Vol.2*, pages 1249-1254, Detroit, Michigan, August 1989.

7. H. Eirund and K. Kreplin. "Knowledge-Based Document Classification Supporting Integrated Document Handling". In *Proc. ACM Conf. on Office Information Systems*, pages 189-196, Palo Alto, CA, March 1988.

8. H. Fujisawa, Y. Nakano, and K. Kurino. "Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis". *Proceedings of the IEEE*, 80(7):1079-1091, July 1992.

9. M. Halliday. "Categories of the Theory of Grammar". *Word*, 17:241-292, 1961.

10. X. Hao, J. T. L. Wang, M. P. Bieber, and P. A. Ng. "A Tool for Classifying Office Documents". In *Proc. Fifth International Conference on Tools with Artificial Intelligence*, pages 427-434, Boston, Massachusetts, USA, November 1993.

11. X. Hao, J. T. L. Wang, M. P. Bieber, and P. A. Ng. "Heuristic Classification of Office Documents". *Int. Journal on Artificial Intelligence Tools*, 3(2):233-265, 1994.

12. X. Hao, J. T. L. Wang, and P. A. Ng. "Information Extraction from the Structured Part of Office Documents". *submitted to Information Science Journal*.

13. X. Hao, J. T. L. Wang, and P. A. Ng. "Nested Segmentation: An Approach for Layout Analysis in Document Classification". In *Proc. of Second IAPR Conference on Document Analysis and Recognition*, pages 319–322, Tsukuba Science City, Japan, October 1993.

14. W. Horak. "Office Document Architecture and Office Document Interchange Formats - Current Status of International Standardization". *IEEE Computer*, 18(10):50–60, October 1985.

15. P. S. Jacobs and L. F. Rau. "SCISOR: Extracting Information from On-line News". *Communication of the ACM*, 33(11):88–97, November 1990.

16. K. S. Jones. *"Automatic Keyword Classification for Information Retrieval"*. Butterworths, London, 1971.

17. K. S. Jones. "Notes and References on Early Automatic Classification Work". *SIGIR Forum*, pages 10–17, Spring 1991.

18. K. S. Jones and R. G. Bates. "Research on Automatic Indexing". Technical Report 5428, British Library R&D Report, Britain, 1975.

19. M. Lebowitz. "The Nature of Generalization in Understanding". In *Proc. of IJCAI-81*, pages 348–353, Vancouver, B. C., Canada, August 1981.

20. W. G. Lehnert, J. B. Black, and B. J. Reiser. "Summarizing Narratives". In *Proc. of IJCAI-81*, pages 184–189, Vancouver, B. C., Canada, August 1981.

21. W. G. Lehnert, M. G. Dyer, P. N. Johnson, C. J. Yang, and S. Harley. "BORIS— An Experiment in In-Depth Understanding of Narratives". *Artificial Intelligence*, 20:15–62, 1983.

22. Q. Liu. "An Office Document Retrieval System with the Capability of Processing Incomplete and Vague Queries," Ph.D Dissertation, New Jersey Institute of Technology, Newark, NJ. October 1994.

23. Q. Liu and P. A. Ng. "A Browser of Supporting Vague Query Processing in an Office Document System". *Journal of Systems Integration*, 5(1), January 1995.

24. Q. Liu, T. L. Wang, and P. A. Ng. "On Research Issues Regarding Uncertain Query Processing In an Office Document Retrieval System". *Journal of Systems Integration*, 3(2):163–194, June 1993.

25. E. Lutz, H. V. Kleist-Retzow, and K. Hoernig. "MAFIA-An Active Mail-Filter-Agent for an Intelligent Document Processing Support". In S. Gibbs and A. A. Verrijn-Stuart, editors, *Multi-User Interfaces and Applications*, pages 16–32. Elsevier Science Publishers B.V., 1990.

26. T. W. Malone, K. R. Grant, and K. Lai. "Semistructured Messages Are Surprisingly Useful for Computer-Supported Coordination". *ACM Trans. on Office Information Systems*, 5(2):115–131, April 1987.

27. N. M. Mattos and B. Mitschang. "An Approach to Integrated Office Document Processing and Management". In *Proc. ACM Conf. on Office Information Systems*, pages 118–122, Cambridge, Massachusetts, April 1990.

28. J. A. McHugh. *Algorithmic Graph Theory*. Prentice-Hall, Inc., New Jersey, 1990.

29. F. S. Mhlanga, J. T. L. Wang, T. H. Shiau, and P. A. Ng. "A Query Algebra for Office Documents ". In *Proc. of 2nd International Conference on Systems Integration*, pages 458–467, Morristown, NJ, June 1992.

30. J. M. Morrissey. "Imprecise Information and Uncertainty in Information System". *ACM Trans. on Office Information Systems*, 8(2):159–180, April 1990.

31. D. Niyogi and S. N. Srihari. "A Rule-based System for Document Understanding". In *Proc. AAAI-86*, pages 789–793, Philadephia, PA, August 1986.

32. B. Pagurek, N. Dawes, G. Bourassa, G. Evans, and P. Smithers. "Letter Pattern Recognition". In *Proc. of IEEE Sixth Conference on Artificial Intelligence Application*, pages 313–319, 1990.

33. A. J. H. M. Peels, N. J. M. Janssen, and W. Nawijn. "Document Architecture and Text Formatting". *ACM Trans. on Office Information Systems*, 3(4):347–369, October 1985.

34. B. W. Porter, R. Bareiss, and R. C. Holte. "Concept Learning and Heuristic Classification in Weak-Theory Domains". *Artificial Intelligence*, pages 229–263, April 1990.

35. D. V. Rama and P. Srinivasan. "An Investigation of Content Representation Using Text Grammars". *ACM Trans. on Information Systems*, 11(1):51–75, January 1993.

36. L. F. Rau, P. S. Jacobs, and U. Zernik. "Information Extraction and Text Summarization Using Linguistic Knowledge Acquisition". *Information Processing & Management*, 25(4):419–428, 1989.

37. E. Riloff. "Automatically Constructing a Dictionary for Information Extraction Tasks". In *Proc. of Eleventh National Conference on Artificial Intelligence*, 1993.

38. E. Riloff. "Using Cases to Represent Context for Text Classification". In *Proc. of AAAI Spring Symposium on Cased-Based Reasoning and Information Retrieval*, 1993.

39. E. Riloff and W. Lehnert. "Classifying Texts Using Relevancy Signatures". In *Proc. of Tenth National Conference on Artificial Intelligence*, pages 329–334, 1992.

40. E. Riloff and W. Lehnert. "Information Extraction as a Basis for High-Precision Text Classification". *ACM Transactions on Information Systems*, 12:269–333, 1994.

41. G. Salton. "A Theory of Indexing". In *Regional Conference Series in Applied Mathematics*, Philadephia, 1975.

42. G. Salton. *Automatic Text Processing*. Addison-Wesley, Massachusetts, 1989.

43. J. Schurmann, N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberlander. "Document Analysis – From Pixels to Contents". *Proceedings of the IEEE*, 80(7):1101–1119, July 1992.

44. F. Y. Shih, S. Chen, D. C. Hung, and P. A. Ng. "A Document Segmentation, Classification and Recognition System". In *Proc. 2nd. Int. Conf. on Systems Integration*, pages 258–267, Morristown, NJ, June 1992.

45. S. Tsujimoto and H. Asada. "Major Components for a Complete Text Reading System". *Proceedings of the IEEE*, 80(7):1133–1149, July 1992.

46. J. T. L. Wang, F.S. Mhlanga, and P. A. Ng. "A New Approach to Modeling Office Documents". *ACM SIGOIS*, 14(2):46–55, December 1993.

47. J. T. L. Wang and P. A. Ng. "TEXPROS: An Intelligent Document Processing System". *Int. Journal of Software Engineering and Knowledge Engineering*, 2(2):171–196, February 1992.

48. J. T. L. Wang, K. Zhang, K. Jeong, and D. Shasha. "A Tool for Tree Pattern Matching". In *Proc. 3rd IEEE Int. Conf. on Tools for Artificial Intelligence*, pages 436–444, San Jose, CA, 1991.

49. J. T. L. Wang, K. Zhang, K. Jeong, and D. Shasha. "A System for Approximate Tree Matching". *IEEE Trans. on Knowledge and Data Engineering*, 6(4):559–571, August 1994.

50. J.T.L. Wang, F.S. Mhlanga, Q. Liu, W.C. Shang, and P.A. Ng. "An Intelligent Documentation Support Environment". In *The Fifth International Conference on Software Engineering and Knowledge Engineering*, pages 429–436, San Francisco, CA, June 1993.

51. C. Wei, J. T. L. Wang, X. Hao, and P. A. Ng. "Inductive Learning and Knowledge Representation for Document Classification: The TEXPROS Approach". In *Proc. of 3nd International Conference on Systems Integration*, pages 1166–1175, Sao Paulo, SP, Brazil, August 1994.

52. P. H. Winston. *Artificial Intelligence.* Addison-Wesley Publishing, Massachusetts, 1984.

53. P. H. Winston. *Artificial Intelligence.* Addison-Wesley Publishing Company, Massachusetts, 1992.

54. K. Woehl. "Automatic Classification of Office Document by Coupling Relational Data Bases and PROLOG Expert System". In *Proc. of Tenth International Conference on VLDB*, pages 528–532, Singapore, August 1984.

55. W. Woods. "Transition Network Grammars for Natural Language Analysis". *CACM*, 13:591–606, 1970.

56. K. Zhang and D. Shasha. "Simple Fast Algorithms for the Editing Distance between Trees and Related Problems". *SIAM J. Comput.*, 18:1245–1262, December 1989.

57. K. Zhang, D. Shasha, and J. T. L. Wang. "Approximate Tree Matching in the Presence of Variable Length Don't Cares". *Journal of Algorithms*, 16(1):33–66, January 1994.

58. Z. Zhu, J.A. McHugh, J.T.L. Wang, and P.A. Ng. "A Formal Approach to Modeling Office Information Systems". *Journal of Systems Integration*, 4(4):373–403, December 1994.