MACQUARIE UNIVERSITY

SYDNEY ~ AUSTRALIA

ITEC 810 Minor Project

Inferring Document Structure

Final Report

Author: Weiyen Lin

SID: 41348133

Supervised by: Jette Viethen

4th June 2009

Abstract

PDF documents form a rich resource repository of knowledge on the Internet both for academia and for business. With their high portability and capability of rendering, they are easy for human readers to read, print, and exchange. However, the lack of logical structure information in PDF documents greatly limits the possibilities of Natural Language Processing tasks such as automatic information retrieval, high-accuracy search or in-depth corpus analysis. In order to enhance their usability, the logical entities of a PDF document, such as title, headings, paragraphs, or reference items for academic articles, need to be detected and annotated. This project builds on previous work extracting the physical layout information of conference papers from PDF files and aims to detect the logical structure from the physical layouts. We applied a two-phase strategy of detection and designed one algorithm for each phase, respectively. In Phase I, blocks with homogeneous physical features in the XML source files are aggregated. In Phase II, each block generated in the previous phase is further annotated with a logical label by heuristic rules based on the general format of the articles in the Association for Computational Linguistics (ACL) Anthology, which is the corpus of interest in our research. The algorithms are implemented using objected-oriented technology which has advantages such as knowledge transparency and high flexibility of extension. The results from a preliminary evaluation have shown this detection strategy on logical entities such as title, abstract heading and abstract, section headings, and authors and affiliation, obtains a satisfactory accuracy.

Acknowledgments

I'd like to express many thanks and appreciation here to my supervisor Jette Viethen, providing much information and advice on the design of algorithms and evaluation strategies. During the development with a tight time limit, her words are both valuable and encouraging. I also appreciate Brett Powley's efforts, who extracted a rich set of physical information from the ACL Anthology corpus. Moreover, comments on workshop paper from Professor Robert Dale and other two anonymous reviewers contribute to the final presentation of this research as well.

TABLE OF CONTENTS

TABLES

FIGURES

1 Introduction

1.1 Background

With the use of the Internet spreading fast in the last decade, the demand of transforming document images into machine-readable documents with logical structure is rapidly increasing. The document images available on the internet mostly come from early-age archives in libraries, such as out-of-date newspapers, magazines, etc., as well as a large volume of PDF documents from the later year of the digital age since the 1990s, such as academic articles, technical instructions, and business advertisements. Most document image are generated for the aims of reducing storage space or facilitating printing and representation. However, since they do not provide information about their logical structure, such as titles, authors, section headings, figures and tables, the possibilities for automatic information processing, such as retrieval, modification, and transformation, are limited in many ways. For example, we cannot perform specific queries on academic documents in PDF with specific search criteria, such as “Author = ‘John Smith’” or “Title LIKE ‘%Mobile Commerce%’”, since this logical information is not specified in those documents. This may result in low search accuracy and longer search times for Internet search engines that have to match the search keywords to the full text, instead of concentrating on title, abstract, or headings only. On the other hand, if logical structure information is provided, re-formatting can be easily done in an automated manner. Readers can choose preset stylesheets or even design their own to reformat the document according to their preference of reading without need to change the original document. In many academic areas, researchers can do more in-depth analysis. For example, linguists can analyse the writing style of an author.

Due to the many advantages of documents being annotated with logical meaning, there have been a large number of related studies on the detection of logical structure in document images during the last decades. This research builds on previous work by Powley et al. (2008) that has extracted the physical layout information from conference papers from PDF into an XML (Extensible Markup Language) format. Our aim is to detect the logical structure of the articles from these XML files. We developed two algorithms, one for homogeneous block aggregation and one for logical structure detection, and implemented them in an extensible object-oriented framework. We have perfomrmed a preliminary evaluation on a small number of unseen articles from the Association for Computational Linguistics Anthology, the corpus we are working on.

1.2 Terminology

Table 1: Terminology of document layout and structure

Terminology	Description	Alternate Terms
Physical layout [Namboodiri , 2007]	Physical location and boundaries of various regions in the document. [Namboodiri, 2007]	Physical structure Page layout [Conway, 1993] Geometric structure [Niyogi, 1995]
Logical Structure [Namboodiri , 2007; Conway, 1993]	Logical or functional structure in the document, such as title, paragraph, captions, etc, with meaningful purposes of reading or understanding. [Namboodiri, 2007]	Logical layout

Table 2: Terminology of document contents

Terminology	Description	Alternate Terms
Physical block [Niyogi, 1995]	Physically homogenous regions in the document, such as figures, background, text block, text lines, words, characters, etc. [Namboodiri, 2007]	Physical region [Namboodiri , 2007] Physical segmentation [Niyogi, 1995]
Logical entity [Niyogi, 1995]	Meaningful regions in the document for reading or understanding, such as title, paragraph, figure, table, caption, etc. [Namboodiri, 2007]	Logical component [Lee, 2003; Stehno & Retti, 2003] Logical label [Namboodiri, 2007]

Table 3: Terminology of document image analysis

Terminology	Description	Alternate Terms
Physical Layout Analysis [Mao, 2003]	Decompose a document image into hierarchy of homogenous regions, such as figures, background, text block, text lines, words, characters, etc. [Namboodiri, 2007]	Document layout analysis [Namboodiri, 2007; Niyogi, 1995] Geometric structure analysis [Lee, 2003] Layout analysis [Stehno, 2003] Document analysis [Nagy, 1992; Tsujmoto & Asada, 1992]
Logical Structure Analysis [Mao, 2003; Lee, 2003]	The process of extracting the logical structure from the document [Niyogi, 1995] or mapping from the physical regions (i.e. physical blocks) in the document to their logical labels (i.e. logical entities) [Namboodiri, 2007]	Logical structure derivation [Niyogi, 1995] Document structure understanding [Namboodiri, 2007] Document understanding [Stehno, 2003; Tsujmoto & Asada, 1992]

1.3 Document Organization

This research first examines related literature both on physical layout analysis and logical structure analysis. Some of the methodologies in the literature are similar to the ones adopted by this research. Section 3 outlines the structure of our source material. In Section 4, we introduce a two-phase processing strategy for logical structure detection and describe our algorithms for aggregating homogeneous blocks and detecting logical structure. The implementation of these algorithms is sketched in Section 5, followed by some final remarks in Section 6.

2 Literature Review

The study dealing with document image processing is recently referred as "***document analysis and understanding***", which involves two major processes: ***physical layout analysis*** and ***logical structure analysis*** [Lee, 2003; Namboodiri, 2003; Niyogi & Srihari 1995]. There have been a large number of studies done in this research area for the last two decades. This section first examines two major issues along with their terminologies in the field of document analysis and understanding (in Section 2.1), followed by a deep review on those previous studies on physical layout analysis (in Section 2.2) and logical structure analysis (in Section 2.3). The end of this section summarizes related literature and gives a suggestion to the methodology that will be adopted by this project.

2.1 Document Analysis and Understanding

Namboodiri [2003] defined document images as those digital documents with both texts and graphics, which are digitally generated from scanner or digital cameras. A digital library usually stores out-of-date newspapers or magazines in films or PDF, which are the major source of document images. Generated and stored digitally, document images were basically aimed to reduce the storage space of a great amount of paper-based literature in traditional libraries.

When reading a document, a human reader can use various clues from its formatting layout, such as the title, headings, tables, and figures, easily to gain the main ideas of the document or quickly to retrieve certain contents he or she is interested. In fact, those formatting features reveal the ***logical structure*** of a document with more meaningful instructions for human's processing (see Figure 1). However, even though document images are digitally stored and can be viewed by human readers from computer screens, they are not accessible by machines to retrieve document content in the same manner, let alone to understand the document. They only contain ***physical layout*** of a document, such as the position of each text, and its font size, style, and alignment, mainly used for rendering. Without logical or structural information embedded, document images are still as less efficient as paper-based documents from the perspectives of document processing, such as retrieval, modification, and transform [Lee et al., 2003; Stehno & Retti, 2003].

Document analysis and understanding is a field of document image processing with an aim of transforming a document image into logical or structural version. Many researchers [Lee, 2003; Namboodiri, 2003; Niyogi & Srihari 1995; Stehno & Retti, 2003] agreed that it consists of two major processes: physical layout analysis and logical structure analysis.

Namboodiri [2003] defined ***document layout analysis*** as a process of "decomposing a document image into a hierarchy of maximally homogeneous regions, where each region is repeatedly segmented into maximal sub-regions of specific type." Those homogeneous

regions or ***physical blocks*** include of figures, background, text block, text lines, words, and characters, etc. There are two major strategies for extracting those physical blocks from a document image: top-down and bottom-up methods, which will be introduced in Section 2.2.

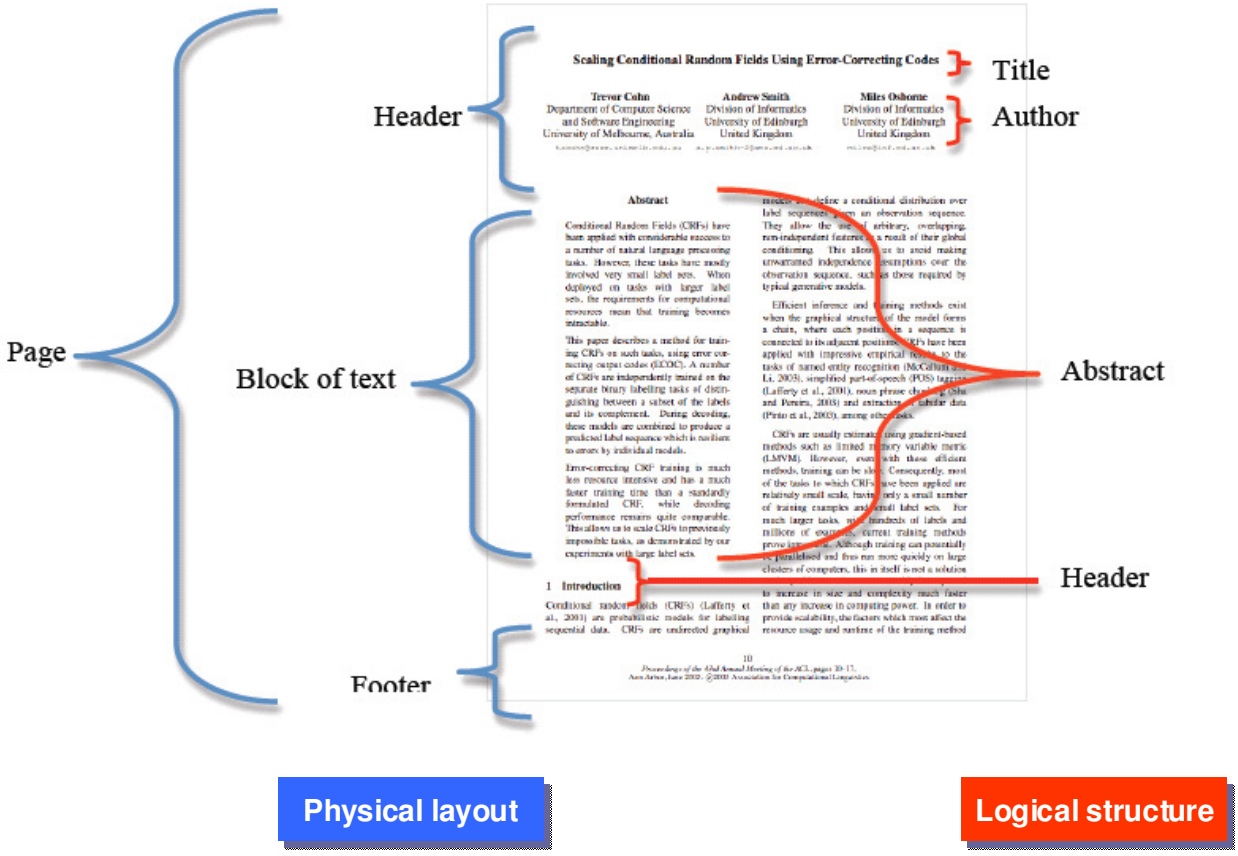


Figure 1: Physical layout and logical structure of a document image

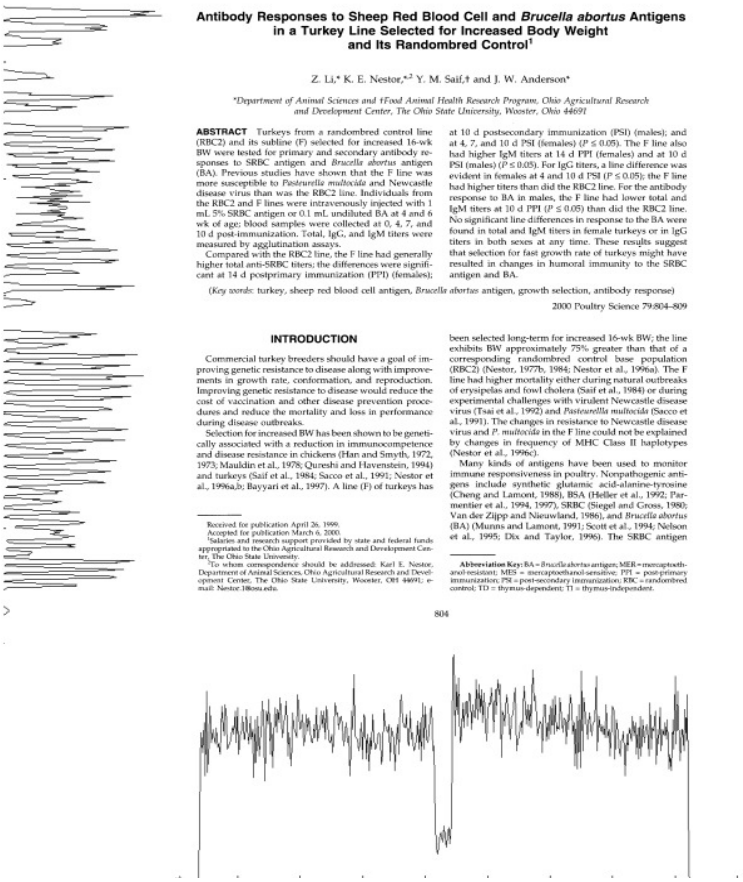
Those physical blocks are then identified as meaningful ***logical entities*** according to their functionalities of document processing, such as retrieval, modification, or transformation. Namboodiri [2003] defined the process of "assigning the logical layout labels to physical regions identified during physical layout analysis" as ***logical structure analysis***. Simply speaking, the logical structure is a mapping from the physical blocks in the document to their logical entities. Document images with logical structure information are now useful for machines to process documents.

2.2 Methods for Physical Layout Analysis

According to Namboodiri [2003], the algorithms of document layout analysis can be divided into ***top-down approaches*** and ***bottom-up approaches*** based on their order of processing. Top-down approaches start with the whole document image and repeatedly break it down into smaller homogeneous regions until each region is recognized as a primitive unit, such as a pixel, word, or graphics. Bottom-up approaches, on the other hand,

start with those primitive units and repeatedly group them into larger regions such us words, lines, or text-blocks.

Namboodiri [2003] stated the X-Y Cut algorithm is a typical top-down approach proceeding by splitting a document image into smaller regions using horizontal and vertical projection profiles. It starts dividing a document image based on valleys in their projection profiles. The algorithm repeats to project the regions of the current segment both on the horizontal and vertical axes until a stop criterion that determines the minimal unit of a region is reached. Other top-down approaches include shaped-directed cover algorithm and the white stream based segmentation.



Antibody Responses to Sheep Red Blood Cell and *Brucella abortus* Antigens in a Turkey Line Selected for Increased Body Weight and Its Randombred Control[1]

Z. Li,* K. E. Nestor,*[,2] Y. M. Saif,† and J. W. Anderson*

*Department of Animal Sciences and †Food Animal Health Research Program, Ohio Agricultural Research and Development Center, The Ohio State University, Wooster, Ohio 44691

ABSTRACT Turkeys from a randombred control line (RBC2) and its subline (F) selected for increased 16-wk BW were tested for primary and secondary antibody responses to SRBC antigen and *Brucella abortus* antigen (BA). Previous studies have shown that the F line was more susceptible to *Pasteurella multocida* and Newcastle disease virus than was the RBC2 line. Individuals from the RBC2 and F lines were intravenously injected with 1 mL 5% SRBC antigen or 0.1 mL undiluted BA at 4 and 6 wk of age; blood samples were collected at 0, 4, 7, and 10 d post-immunization. Total, IgG, and IgM titers were measured by agglutination assays.

Compared with the RBC2 line, the F line had generally higher total anti-SRBC titers; the differences were significant at 14 d postprimary immunization (PPI) (females); at 10 d postsecondary immunization (PSI) (males); and at 4, 7, and 10 d PSI (females) ($P \leq 0.05$). The F line also had higher IgM titers at 14 d PPI (females) and at 10 d PSI (males) ($P \leq 0.05$). For IgG titers, a line difference was evident in females at 4 and 10 d PSI ($P \leq 0.05$); the F line had higher titers than did the RBC2 line. For the antibody response to BA in males, the F line had lower total and IgM titers at 10 d PPI ($P \leq 0.05$) than did the RBC2 line. No significant line differences in response to the BA were found in total and IgM titers in female turkeys or in IgG titers in both sexes at any time. These results suggest that selection for fast growth rate of turkeys might have resulted in changes in humoral immunity to the SRBC antigen and BA.

(*Key words:* turkey, sheep red blood cell antigen, *Brucella abortus* antigen, growth selection, antibody response)

2000 Poultry Science 79:804–809

INTRODUCTION

Commercial turkey breeders should have a goal of improving genetic resistance to disease along with improvements in growth rate, conformation, and reproduction. Improving genetic resistance to disease would reduce the cost of vaccination and other disease prevention procedures and reduce the mortality and loss in performance during disease outbreaks.

Selection for increased BW has been shown to be genetically associated with a reduction in immunocompetence and disease resistance in chickens (Han and Smyth, 1972, 1973; Mauldin et al., 1978; Qureshi and Havenstein, 1994) and turkeys (Saif et al., 1984; Sacco et al., 1991; Nestor et al., 1996a,b; Bayyari et al., 1997). A line (F) of turkeys has been selected long-term for increased 16-wk BW; the line exhibits BW approximately 75% greater than that of a corresponding randombred control base population (RBC2) (Nestor, 1977b, 1984; Nestor et al., 1996a). The F line had higher mortality either during natural outbreaks of erysipelas and fowl cholera (Saif et al., 1984) or during experimental challenges with virulent Newcastle disease virus (Tsai et al., 1992) and *Pasteurella multocida* (Sacco et al., 1991). The changes in resistance to Newcastle disease virus and *P. multocida* in the F line could not be explained by changes in frequency of MHC Class II haplotypes (Nestor et al., 1996c).

Many kinds of antigens have been used to monitor immune responsiveness in poultry. Nonpathogenic antigens include synthetic glutamic acid-alanine-tyrosine (Cheng and Lamont, 1988), BSA (Heller et al., 1992; Parmentier et al., 1994, 1997), SRBC (Siegel and Gross, 1980; Van der Zijpp and Nieuwland, 1986), and *Brucella abortus* (BA) (Munns and Lamont, 1991; Scott et al., 1994; Nelson et al., 1995; Dix and Taylor, 1996). The SRBC antigen

Received for publication April 26, 1999.
Accepted for publication March 6, 2000.
[1]Salaries and research support provided by state and federal funds appropriated to the Ohio Agricultural Research and Development Center, The Ohio State University.
[2]To whom correspondence should be addressed: Karl E. Nestor, Department of Animal Sciences, Ohio Agricultural Research and Development Center, The Ohio State University, Wooster, OH 44691; e-mail: Nestor.1@osu.edu.

Abbreviation Key: BA = *Brucella abortus* antigen; MER = mercaptoethanol-resistant; MES = mercaptoethanol-sensitive; PPI = post-primary immunization; PSI = post-secondary immunization; RBC = randombred control; TD = thymus-dependent; TI = thymus-independent.

804

Figure 2: Horizontal and vertical projection profiles [Namboodiri, 2003]

Bottom-up approaches, such as the run length smoothing algorithm (RLSA), first define the basic unit in order to start the grouping process. It consists of four major steps: a horizontal smoothing, a vertical smoothing, a logical AND operation and an additional horizontal smoothing [Fisher et al., 1990]. In those steps, the distance between two adjacent units is calculated and compared with a threshold, either a horizontal threshold or vertical one. If the distance is less than the threshold, then two units are joined together. The vertical smoothed image is then logically ANDed with the horizontal smoothed one, and is horizontally smoothed one more time. The resulting image is the so-called RLSA image.

2.3 Methods for Logical Structure Analysis

Compared to studies on physical layout analysis with an aligned set of terms for their methodology, researchers dealing with logical structure analysis diverse in their selection of names for the methods they adopt. Even though some researchers have tried to classify the method adopted by previous research, they use different names for methods. For example, Lee et al. [2003] divided related work for logical structure analysis into the ***syntactic methods*** and the ***model-matching methods*** (see Table 4). Mao et al. [2003] stated that document logical structures are represented by models derived either from ***a set of rules*** or from ***formal grammars*** (see Table 5). Stehno and Retti [2003] categorised models representing the logical structure into three: ***rule-based models***, ***grammar-based models***, and ***models using statistical or probabilistic methods***.

Table 6 compares the classification schemes by the three different papers to each other. Although they did not give definitions for each method, by inspecting the approaches in each classification, it can be inferred that syntactic or grammatical methods regard the document as a sequence of repeated objects. For example, Lee et al. view a document as a sequence of headers and bodies [Lee et al., 2003], while Conway regards a document as a string or sentence to be parsed [Conway, 1993]. They create a grammar to describe the logical structure in terms of sequences of neighbouring blocks. By applying a certain parsing algorithm repeatedly, the logical structure is identified either in a top-down manner [Lee et al., 2003] or in a bottom-up manner [Conway, 1993].

On the other hand, model-matching methods or rule-based methods do not create a syntax or grammar to represent the logical structure. Instead, they encode the knowledge about mapping each physical block to the most likely logical entity in the form of rules and by applying the preset rules toward each physical block, each block can be specified with a logical entity label. For example, such a rule could say: “If a block is of type ‘large text’ and located at the beginning of the document, then it is a title”. By applying these rules to predict each physical block to be a logical entity these approaches build up the logical structure of the whole document. They also apply rules to regulate the process of logical structure detection.

Both the grammar and the knowledge established in the two respective methods are subject to a certain document type, such as newspaper pages, books, journal articles, or business letters. Therefore, it might not be possible to generate a universal domain model fitting all types of documents. ***Statistical or probabilistic methods*** uses a large volume of annotated data and adopts statistical pattern recognition algorithms to enable a logical structure detector to apply a corresponding grammar, based on documents it is fed.

Since there are no annotated data sets available in this project, the statistical methods are beyond consideration. Instead, we will adopt ***rule-based*** and ***syntactic approaches*** for

the logical structure analysis. The next two sub-sections inspect some specific models from these two categories.

Table 4: Logical structure analysis methods summarized by Lee [Lee et al, 2003]

Classification	Author	Year	Characteristics
Syntactic	Nagy et al. [6]	1992	Describe logical hierarchical structure based on the length and frequency of a pixel
	Krishnamoorthy et al. [7]	1993	
	Story et al. [15]	1992	Describe relative position of logical structure elements etc.
	Hu and Ingold [16]	1993	Allow error-tolerant parsing based on fuzzy logic
	Conway [17]	1993	Describe geometric characteristics and spatial relations using a page grammar
	Tateisi and Itoh [18]	1994	Search the lowest cost path from a graph which is a part of connecting relations between text lines
	Klein and Fankhauser [19]	1997	Propose a grammar based on an SGML DTD
	Klein and Abecker [20]	1999	
Model - matching	Dengel and Barth [12]	1988	Describe hierarchical structure in a form of decision tree using the relative size and location of regions
	Dengel et al. [13]	1992	
	Tsujimoto and Asada [11]	1992	Define four rules for transformation of a geometric structure tree into a logical structure tree
	Bayer and Walischewski [31]	1995	Describe geometric characteristic, frequency, and lexical information
	Farrow et al. [10]	1996	Based on rules about geometric characteristics and simple layout model
	Niyogi and Srihari [9]	1996	Describe knowledge and control process in a from of rules
	Lin et al. [29]	1997	Use general knowledge about content structure
	Rus and Summers [25]	1997	Create hierarchical structure using the size and the location of white spaces
	Summers [26]	1998	
	Kochi and Saito [30]	1998	Describe geometric characteristics of target objects
	Hitz et al. [27]	1999	Describe statistical probability distribution of hierarchical structure
	Worring and Smeulders [32]	1999	Based on geometric characteristics and OCR

Table 5: Logical structure analysis methods summarized by Mao [Mao et al, 2003]

Model	Authors	Key Idea	Logical Labels	Domain
Formal grammars	Conway [1993]	page grammar	title, heading, paragraph, figure	not mentioned
	Krishnamoorthy [1993]	page parsing, block grammar	title, author, abstract	journal pages
	Tateisi [1994]	stochastic grammars, physical zones available	headings, paragraph, list item	not mentioned
	Ishitani [1999]	emergent computation, rule based	headline, header, footer note, caption, program, formula, title, list	various documents
A set of rules	Tsujimoto [1990]	mapping a physical tree to a logical one	title, abstract, sub-title, paragraph, header, footer page number, caption	various documents
	Fisher1 [1991]	rule-based	section heading, figure, figure caption, page heading, page footings	not mentioned
	Niyogi [1995]	rule-based, knowledge-based	title, story, sub-story, photo, caption, graph	newspaper pages
	Summers [1995]	logical prototype, matching, physical zones available	paragraph, heading, list item	technical reports

Table 6: Summary of classifications of logical structure analysis methods

Classifier	Lee et al [2003]		Mao et al [2003]		Stehno et al [2003]		
Method / Research	Model-matching	Syntactic	A set of rules	Formal grammars	Rule Model	Grammar Model	Statistics Model
Tsujimoto [1992]	●		●				
Niyogi [1996]	●		●		●		
Fisher [1990]			●		●		
Nage [1992]		●		●		●	
Conway [1993]		●		●		●	
Tateisi [1994]		●		●			
Lee [2000]					●		
Altamura [1999]					●		
Cesarini [1999]						●	
Brugger [1997]							●
Ittner [1993]							●
Palmero [1999]							●

2.3.1 Rule-based Approaches

- **Niyogi & Srihari [1995], Knowledge-Based Derivation of Document Logical Structure**

Niyogi and Srihari [1995] established a knowledge-based system for the derivation of logical structure of newspaper pages. A computational model is developed consisting of a Knowledge base, Inference Engine and a Global Data Structure representing the syntax of a typical newspaper. Figure 3 illustrates the architecture of proposed system (DeLos).

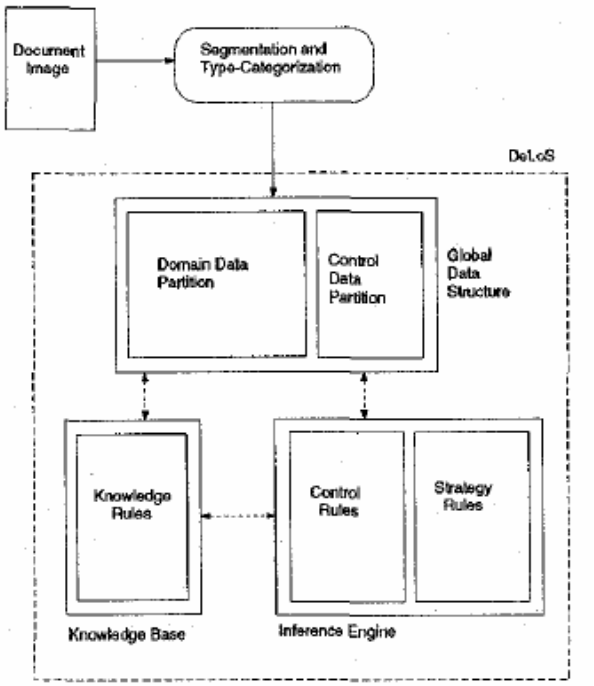


Figure 3: The DeLos system [Niyogi & Srihari, 1995]

The **Global Data Structure** stores the syntax of newspapers as shown in Table 7. It is used to specifying a logical entity label to a physical block by applying the rules in the knowledge base. The **Knowledge Base** contains all the rules in form of first-order predicates. These rules define the general characteristics of each logical entity in a newspaper as well as the relationships between such entities. All common characteristics of different logical entities and their geometrical constraints are encoded in the knowledge base. This knowledge is then used for block segmentation, block grouping, or text block ordering. An example of these knowledge rules is shown in Table 8 (a).

The **Inference Engine** contains two more levels of rules: control rules and strategy rules. Control rules regulate the invocation of a knowledge rule while strategy rules determine which control strategy is to be applied. The application starts with segmenting a document image using a bottom-up algorithm, then those segmented blocks are grouped, and finally the grouped blocks are imported intothe DeLos system and a logical tree structure is derived [Niyogi & Srihari, 1995; Mao et al., 2003]. The output of the system is a tree

representing the logical structure of the imported document. The system was tested on 44 US newspaper pages and the performance, in terms of block classification accuracy, block grouping accuracy, and read-ordering accuracy, is fairly high [Mao et al., 2003].

Table 7: Syntax of a typical newspaper [Niyogi & Srihari, 1995]

```
<document> : : = { <page> }
<page>          : : = { <block> }
<block>         : : = <large-text> | <medium-text> |
                      <small-text> | <line-drawing> |
                      <half-tone> | <boundary>
<boundary> : : = <horizontal-line> | <vertical-line> |
                      <line-rectangle>
```

(a) Physical structure

```
<document info> : : = { <unit> )
<unit>                  : : = <title> | <graphical area> |
                              <story> | <photoblock>
<photoblock>        : : = [ <title> | <photo> <caption>
<graphical area> : : = <page banner> | <horizontal band, |
                              <other graphics>
<story>                 : : = [ <sub-story> ] |
                              <title> <sub-title> |
                             { <text-para> } [ <photoblock> ]
                            [ [ <title> ] <chart> <caption> ]
                            [ [ <title> ] <table> <caption> ]
<sub-story>         : : = <story>
```

(b) Logical structure

Table 8: Example of rules in DeLos [Niyogi & Srihari, 1995]

```
IF a block Z is of type "large-text",
        OR IF it satisfies the following three conditions:
            (it is of type "medium-text",
                    AND it is below another block W,
                    AND block W is not of type "large-text"
                        or "medium-text" 1,
THEN block 2 is a major headline.
```

(a) Knowledge Rule

```
IF the grouping mode is on,
    AND a block has been selected,
THEN find all the immediate neighbors
          of the selected block.
```

(b) Control Rule

```
IF any partially grouped units remain,
THEN apply all unit-related control rules
          for each of these units
      until there are no more partial units.
```

(c) Strategy Rule

• Tsujimoto & Asada [1992], Major Components of Complete Text Reading System

Tsujimoto and Asada [1992] present a method to document analysis, document understanding, and character segmentation/recognition for a text reader system. The document analysis component extracts text lines from a document image and obtains the geometric structure tree (see Figure 4 b) as a hierarchy of physical blocks. The document understanding component then maps a pre-defined logical structure tree (see Figure 5) with the geometric structure tree using four transformation rules that deal with the tree nodes movement during the transformation. The character segmentation/recognition component extracts characters from a text line and recognises them based on the heuristics of character composition and recognition results.

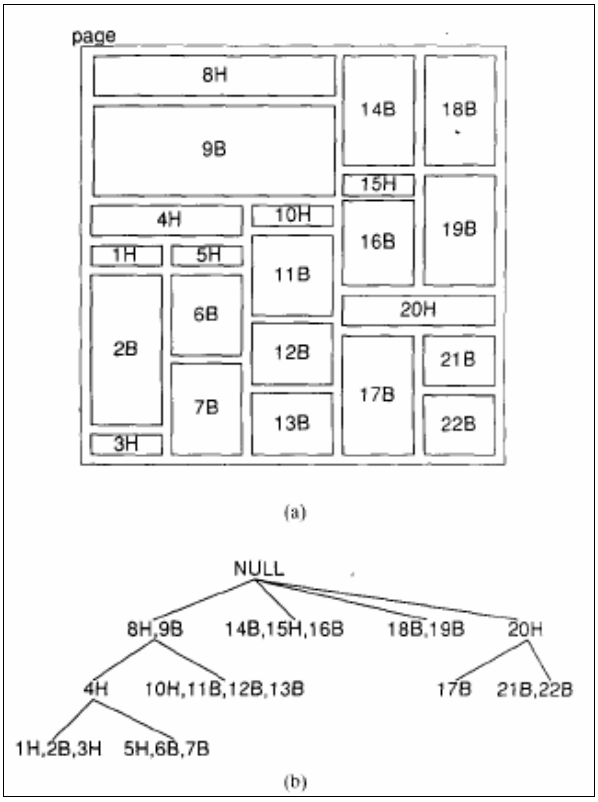


Figure 4: Geometric structure tree [Tsujimoto & Asada, 1992]

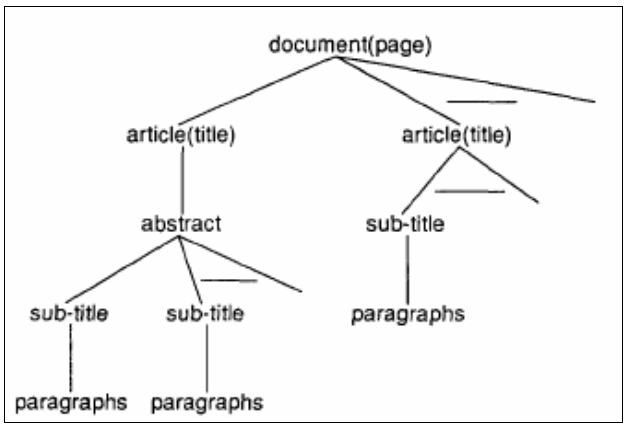


Figure 5: Logical structure tree [Tsujimoto & Asada, 1992]

The algorithm for the geometric to logical structure transformation is composed of four transformation rules that define the conditions under which an element in a node list is moved. These rules are shown in Table 9 and illustrated in Figure 6, where H indicates a head block, B indicates a body block, and S indicates that a block can be either body or head. Each node in the tree is sequentially numbered in the depth-first order [Tsujimoto & Asada, 1992]. Rule (a) is based on the observation that each line can only belong to a single paragraph, and Rule (b) is similar to Rule (a). Rule (c) extracts chapters of sections for a subtitle, and Rule (d) attaches a unique class (head/body) to each node [Tsujimoto & Asada, 1992]. Figure 7 shows an example of the transformation from the geometric structure of the document shown in Figure 6.

They tested the algorithm on 106 pages from different sources and obtained a logical structure recognition accuracy of 94/106 [Mao et al., 2003].

Table 9: Transformation Rules [Tsujimoto & Asada, 1992]

Rule (a): **If** a node (say A) is a terminal node, and the first element of node A is a body, and the preceding node (say B) in the depth-first indexing is a terminal node, **then** remove the first element from node A, and append it to the last element of node B.	**Rule (b):** **If** a node (say A) is a terminal node that is not connected to the root node, and the preceding node (say B) in the depth-first indexing is a terminal node, and the first element of node A is not NULL, and last element of node B is a head, **then** remove the first element from node A, and append it to the last element of node B.
Rule (c): **If** a node (say A) contains a head block, and it is not the first element of the node, **then** generate a younger sister node (say D), and remove the head-body sequence that begins with that head block and ends with the last element of node A, with daughters of node A, if any, and attach them to the younger sister node D.	**Rule (d):** **If** there is a head block sequence in a node, and it is the first part of the node, **then** generate a daughter node, and move the body sequence that follows the head sequence to the daughter node.

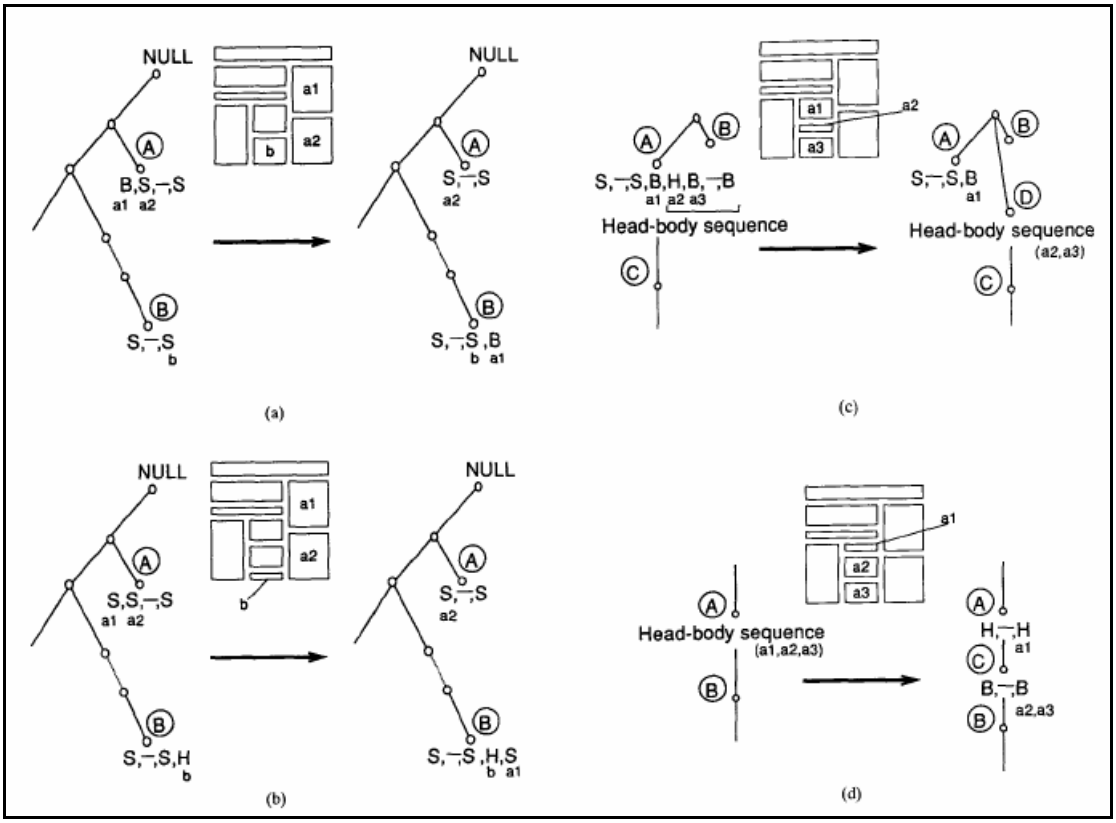


Figure 6: Transforming from a geometric into logical structure [Tsujimoto & Asada, 1992]

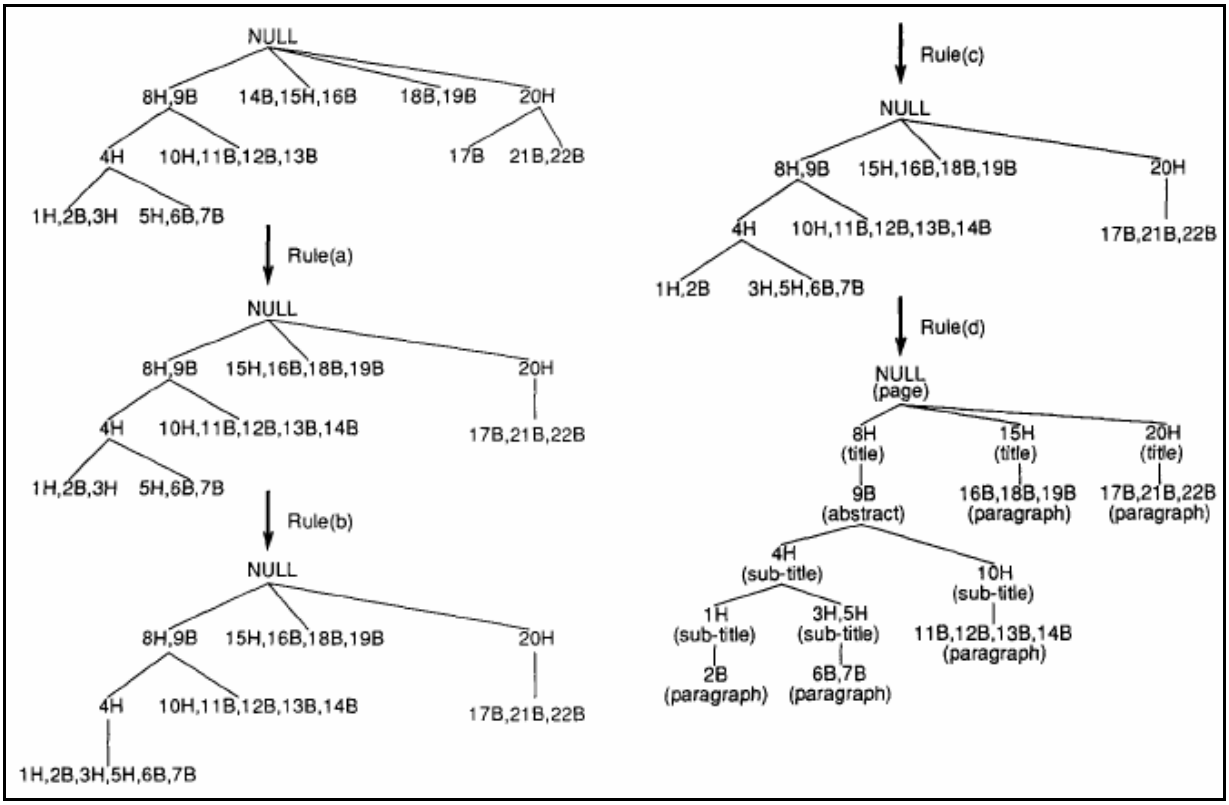


Figure 7: Transformation from a geometric structure in Figure 6 [Tsujimoto & Asada, 1992]

2.3.2 Syntactic Approaches

- **Conway, 1993, Page grammar and page parsing: A syntactic approach to document layout recognition**

Conway [1993] adopted page grammars and page parsing techniques to recognize logical structure from physical document layout. He described the physical layout of a certain type of documents as a page layout grammar similar to a context free string grammar. In other words, he viewed the page as a "sentence" to be parsed. In his system, the first step is to segment the page image using run length smoothing segmentation and produce segments corresponding to text lines and graphic objects. Then the page parser groups segments according to physical layout and produces a list of text and graphic items tagged as headings, paragraphs, figures, etc. in a reading order. Figure 8 illustrates the process of parsing.

The grammar is a set of rules similar to the rules in rule-based approaches mentioned earlier that specify how the logical entities are embedded in the whole document, but in a more syntactic manner. For example, in Conway's system he presented the layout of a typical title page like the following:

TitlePage → (**over** Title Author Organisation Body)

Body → (**leftside** Column Column)

Column → (**over** ParaBody ? (Paragraph I Figure)*

Paragraph → (**over** textline dirst-indented> ParaBody)

ParaBody → (**over** textline<aligned>')

Title → (**over** textline[large, bold]ccentred>+)

etc.

Each logical entity, such as TitlePage or Body, has a grammatical rule with the pattern like:

$$\mathbf{X \rightarrow (ruletype\ X_1 \ldots X_n)}$$

Where **ruletype** is one of the relations: **above**, **leftof**, **over**, **leftside**, and **closeto**. It means an entity **X** can be made up of a sequence of entities $\mathbf{X_1 \ldots X_n}$, which are linked by the relationship **ruletype**. Then the page parsing algorithm applied Kay's active chart parsing, which is mainly used for natural language processing, to parse the entire page. The active chart parsing is based on a data structure called an **edge**, which consists of four components:

<label start end remainder>

The **label** is a grammar symbol, **start** and **end** are positions in the sentence and **remainder** is a list of grammar symbols like the **label**.

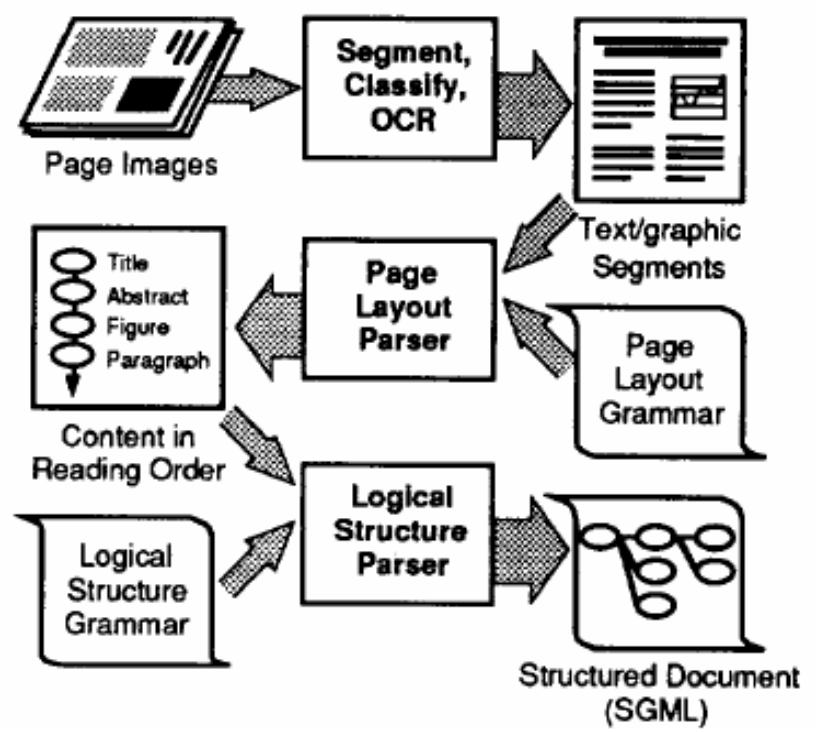


Figure 8: Processing stages [Conway, 1993]

An edge with an empty remainder is called **complete** while an edge with non-empty remainder is called an **active** edge. For example, **<VP 1 5 ()>** in Figure 9 represents the verb phrase "*had a little lamb*", which is complete, and **<VP 1 2 (NP)>** represents a partial verb phrase "*had*" which requires a noun phrase at position 2 to be complete, and therefore is an active edge.

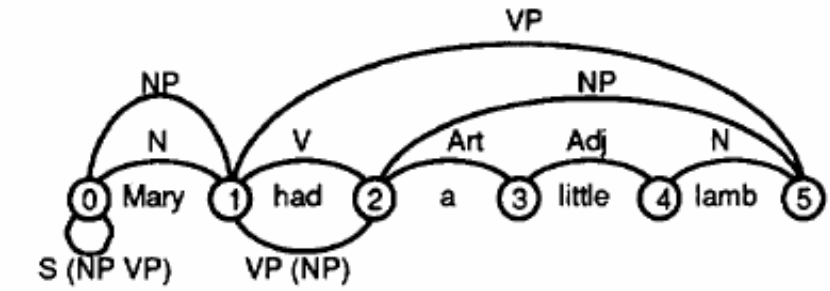


Figure 9: Edge from parsing an English sentence [Conway, 1993]

The basic operation on edges is to **continue** an active edge with a complete edge. If given an active edge **a = <A a_1 a_2 (X_1…X_n)>** and a complete edge **b = <B b_1 b_2 ()>**, then **b** can continue **a** if **X_1 = B** and **a_2 = b_1**. The resulting edge is **<A a_1 b_2 (X_2…X_n)>**. For example, in Figure 9, the edge **<VP 1 2 (NP)>** can be continued by **<NP 2 5>** resulting in a complete edge **<VP 1 5 ()>**.

The page parsing algorithm is similar to the string parsing algorithm described above, with a difference that the **page edge** data structure represents page blocks and the rule for continuing these page edges [Conway, 1993]. A page edge is defined as:

<label ruletype consumed remainder>

The **label** and **remainder** have the same meaning as in string edges. The ruletype specifies the relation that is required to hold between consecutive sub-edges and to be one of five relations mentioned earlier. The start and end components are replaced by consumed component which holds the sequence of sub-edges making up the completed portion of the edge [Conway, 1993]. Thus, the rule for continuing page edges is like:

a = <A rule$_a$ (a_1..a_i) (X_1…X_n)> (active)

b = <B rule$_b$ (b_1..b_j)> (active)

Then **b** can continue **a** if X_1 **= B** and **rule$_a$ (a_i , b)** holds. The resulting edge is then **a = <A rule$_a$ (a_1..a_i b) (X_2…X_n)>**.

Using page grammar rules and the edges definition, his system applied the chart parsing algorithm to parse pages and generates the logical structure of the document page.

2.4 Summary

In this section we have reviewed literature in the field of document analysis and understanding. Several methods are introduced both for physical layout analysis and logical structure analysis. Some previous works of logical structure analysis applying syntactic approaches as well as rule-based approaches are inspected. This project focuses on physical blocks grouping and logical structure analysis since it continues previous work which delivers an XML version with geometric information for each word in a PDF document. Thus, the extraction of physical blocks from a document image is not part of this project. However, grouping of the words into physical blocks is the first task to be accomplished, followed by mapping from physical layout to logical structure.

Due to the nature of the input file of this project, which provides rich geometric information but doesn't extract table contents and graphic object, a bottom-up algorithm such as run length smoothing algorithm (RLSA) will be adopted in the early stage of physical block grouping. A rule-based approach, matching each physical block to a preset logical entity model for journal articles in both top-down and bottom-up manner, will be applied in the later stage of detecting logical structure from the physical blocks format.

3 Material

XML (Extensible Markup Language) is a structural mark-up language that allows users to define their own mark-up elements, or tags. It aims to provide a human- and machine-readable data format for encoding and sharing information via the internet. This research is based on an XML version of the post-2000 portion of the ACL Anthology corpus which was derived by running Powley et al.'s (2008) software over the online PDF documents in the ACL Anthology corpus from the year 2000 and later. The left-hand side of Figure 10 is an example of the PDF documents from the ACL Anthology corpus and Figure 11 is the XML source with all physical features extracted from that PDF document. Although these PDF documents are different from document images generated by scanning mentioned in Section 2 since they contain specific physical information about the contained text, their logical structure still remain unknown to machines and thus need to be detected.

The derived XML source provides a rich set of physical features for each word roughly in the order of reading on a page, one page after another. Those physical features include the height and width of a document page as well as the font, font size, and the position of each word on that page. The physical meaning of attributes in the XML source of Figure 11 is illustrated in the right-hand side of Figure 10. The extraction sequence of the XML source is roughly in the order of reading, from top to bottom, left to right, and left column to right column, as shown in Figure 12. Simply speaking, the XML source is merely a raw file extracting each word sequentially from a PDF document and representing the text in a hierarchical structure using an XML format, which makes later processing easier. The XML source will be referred to as ***XML Source by Text*** in the remainder of this report for differentiation from other XML formats we use. One example is included in Appendix A.1.

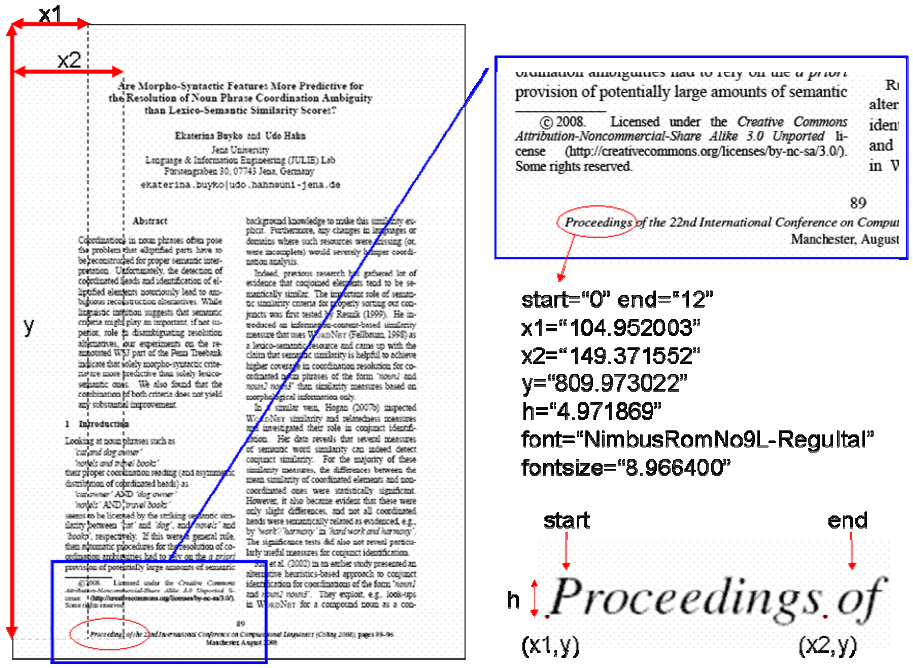


Figure 10: Physical features of a PDF document

```
<?xml version="1.0" encoding="UTF-8" ?>
- <document>
  - <page index="1" height="841.890015" width="595.276001">
    - <text start="0" end="12" x1="104.952003" x2="149.371552" y="809.973022" h="4.971869" font="NimbusRomNo9L-ReguItal"
      fontsize="8.966400">
        <![CDATA[ Proceedings   ]]>
      </text>
    - <text start="12" end="15" x1="151.613159" x2="158.589020" y="809.973022" h="4.971869" font="NimbusRomNo9L-ReguItal"
      fontsize="8.966400">
        <![CDATA[ of   ]]>
      </text>
    - <text start="15" end="19" x1="160.830627" x2="171.787567" y="809.973022" h="4.971869" font="NimbusRomNo9L-ReguItal"
      fontsize="8.966400">
        <![CDATA[ the   ]]>
      </text>
    - <text start="19" end="24" x1="174.029175" x2="191.961975" y="809.973022" h="4.971869" font="NimbusRomNo9L-ReguItal"
      fontsize="8.966400">
        <![CDATA[ 22nd   ]]>
      </text>
    - <text start="24" end="38" x1="194.203583" x2="241.528244" y="809.973022" h="4.971869" font="NimbusRomNo9L-ReguItal"
      fontsize="8.966400">
        <![CDATA[ International   ]]>
      </text>
```

Figure 11: An example of XML Source by Text

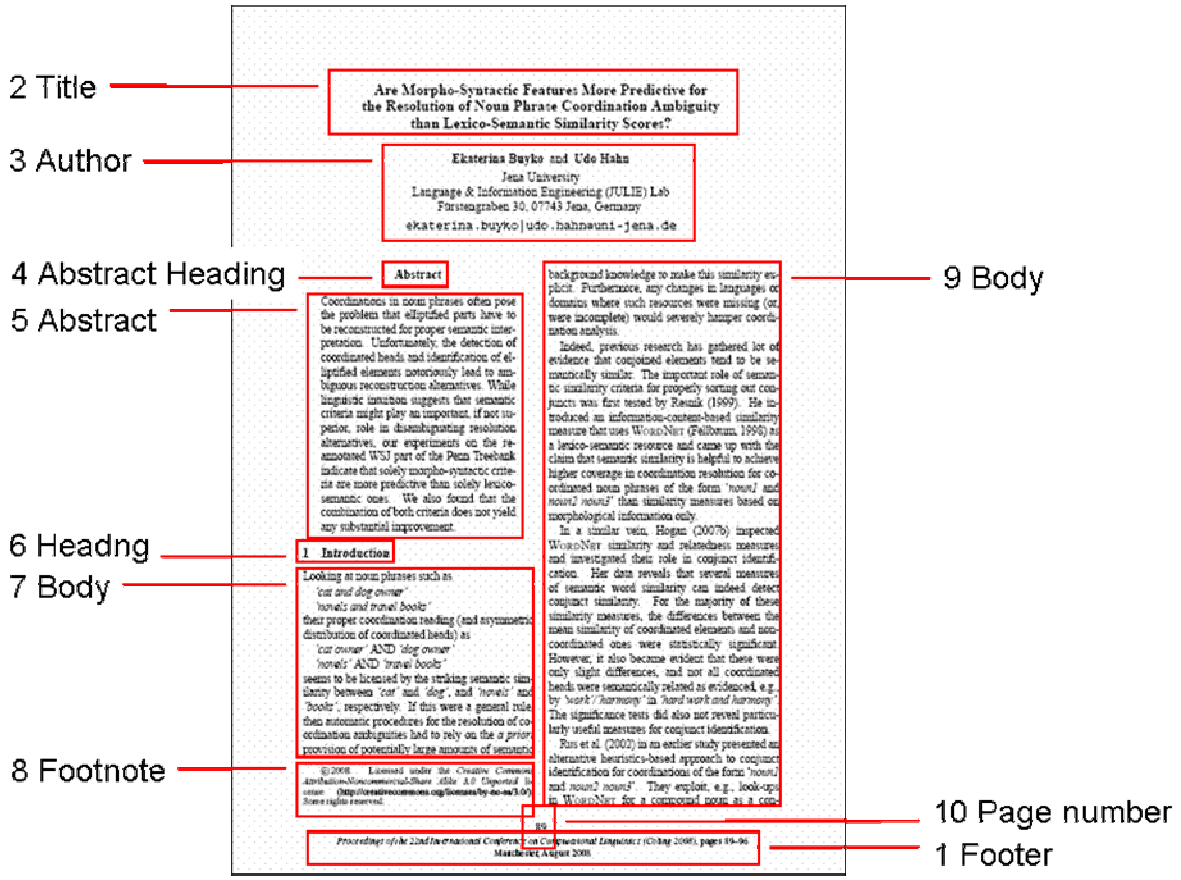


Figure 12: Extraction sequence of XML Source by Text

4 Methodology

This section describes the methodology we adopt for logical structure detection of academic articles. We introduce our two-phase detection strategy, followed by the description of the algorithms implementing the two phases, one for aggregating homogeneous blocks from *XML Source by Text* (Section 4.1) and one for annotating each block with a logical label (Section 4.2). In the end, four output files are introduced to provide a quick overview of the outcomes of this research.

Figure 13 illustrates the process of logical structure detection consisting of two phases. In Phase I, the *XML Source by Text* is read-in and words with same y-position are grouped into a line (1a. in Figure 13). This process produces a newly created XML file, referred to as *XML Source by Line*. The lines in the *XML Source by Line* are further aggregated into homogeneous blocks according to their physical features and the process produces another new XML file, *XML Physical Blocks* (1b. in Figure 13). This process applies an algorithm that will be detailed in Section 4.1. From *XML Physical Blocks* an HTML file with a line break between each block of texts is created by applying an XML Stylesheet (XSLT) on *XML Physical Blocks*. The HTML file is referred to as *HTML Physical Structure* and can be displayed in human-readable form in a web browser.

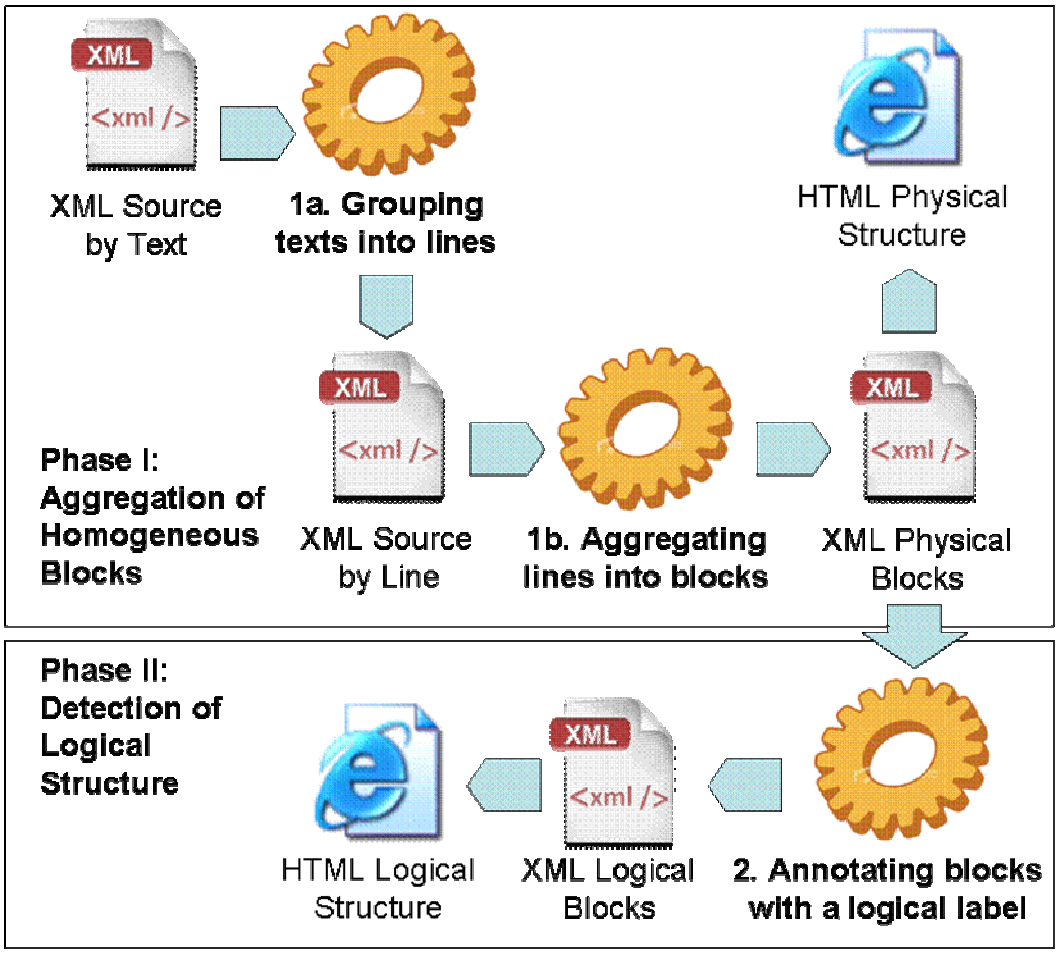


Figure 13: Process of logical structure detection

In Phase II, another algorithm is applied to the *XML Physical Blocks* file to predict the logical meaning of each block according to rules which are based on the knowledge of the format of academic articles in the ACL Anthology corpus. This process (2 in Figure 13) annotates each block with a logical label such as title, author, section-heading, or paragraph, and produces an *XML Logical Blocks* file, which is the final product of our research. It can be further used to produce an *HTML Logical Structure file* which facilitates the manual evaluation process at a later stage. Section 4.2 describes the algorithm and specific rules applied for detecting logical entities.

There are many advantages of separating the detection into two phases. First, separating the implementation of both algorithms can prevent the confusion of errors caused by the physical layout analysis or the logical structure detection. Second, both algorithms can be further refined and extended according to the results of an evaluation. The more precise the block aggregation is, the more accurate the detection of the logical structure will be. With the experience of testing throughout the development, we found this can greatly enhance the accuracy of final detection. Third, the modularisation using object-oriented technology makes it possible for the software to behave as a shell in the future detecting different types of documents by accommodating different layout knowledge for different document types.

4.1 Aggregation of Homogeneous Physical Blocks

The aggregation of homogeneous physical blocks before directly detecting the logical structure of a document is mimicking the process of a person visually taking in the structure of a document. When humans read a document, the attention is first drawn to the physical features of that document, instead of logical features. In other words, human readers first identify the homogeneous blocks of lines or texts according to their physical attributes, such as position, dominant font size or font style, and spacing between those blocks. Then they start to have a closer look at the blocks according to the message that each block transmits to them; for example, read the most upper block with the biggest font size on the first page of the document, which is assumed to be the title at the stage of block aggregation and confirmed as such after reading.

Our approach applies the same strategy and aggregates lines with homogeneous physical features into a block. The method we use is similar to the Run Length Smoothing Algorithm (RLSA) used for physical layout analysis. Our algorithm, as shown in Figure 14, assumes words in the same line in an *XML Source by Line* document belong to one homogeneous block and then reads in three lines at a time to determine which lines, with their words, could further belong to the same block. The algorithm only considers the *dominant font size* of each line, which is the most frequent font size among the words in a line, and the spacing between lines to define physical homogeneity.

As Figure 14 shows, first the dominant font sizes of the first three lines are considered. If

the dominant font sizes of three lines are not identical, cases AAB, ABB, A1BA2, and ABC, lines with the same dominant font size are aggregated into one block, and the rest into another or all lines are assigned to three different blocks. If the dominant font sizes of three lines are identical, then the y-spacing between lines is further examined. Lines with smaller spacing are aggregated into one block, and lines with a larger spacing split off into another block. If the spacing is the same, the three lines are aggregated into one block. In the next iteration of the algorithm, the last line of the previous iteration is read in again as the first line of the next three for a continuous aggregation.

Table 10 and 11 are the pseudo-codes of the aggregation algorithm. It processes each page element in XML Source by Line, where lines of words are either grouped into a new block (as ***createBlock*** in Table 10 and 11) or appended to the currently processing block (as ***appendToBlock*** in Table 11). The criteria applied to determine either to create or to append includes the font size combination of three lines for each round of reading (as shown in Figure 14), and the spacing conditions including three categories: 1) ***SAME_SPACING***, where the spacings between three lines are identical, 2) ***LARGER_THEN_SAMLLER***, where the spacing between the first two lines is larger than that between the last two lines, while ***SMALLER_THEN_LARGER***, vice versa, 3) ***NEGATIVE_SPACING_1,*** where the spacing between the first two lines is negative, which means the change of column occurs between those lines, while ***NEGATIVE_SPACING_2*** means the change occurs between the last two lines.

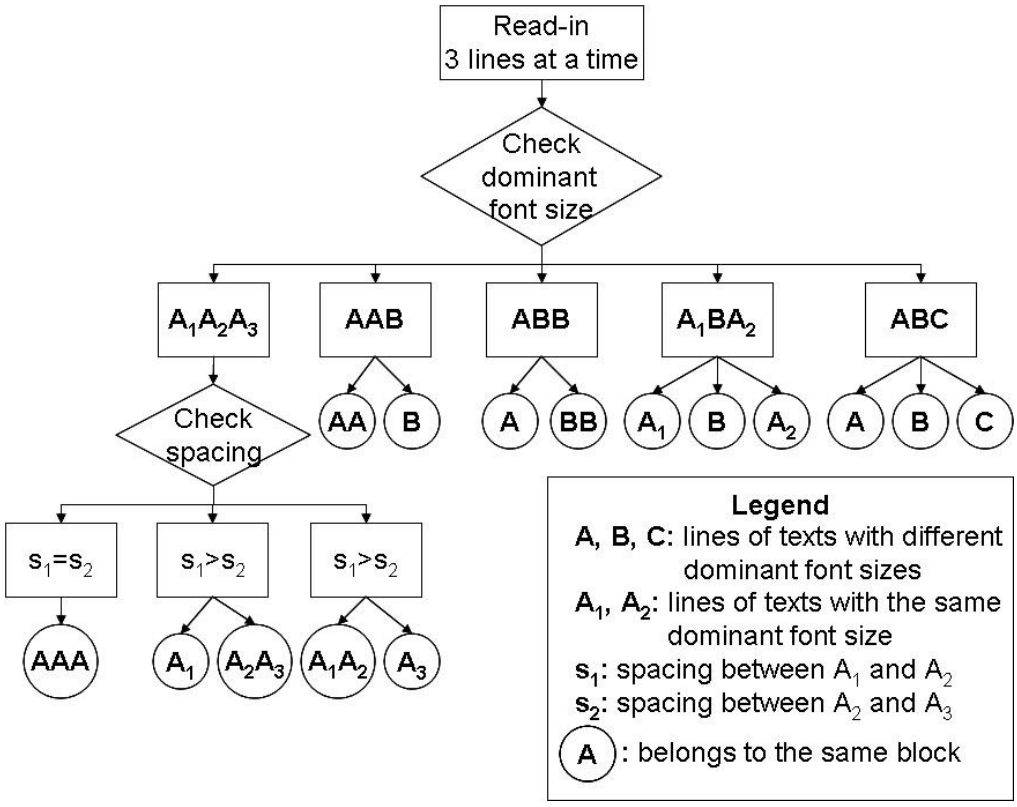


Figure 14: Algorithm for aggregating blocks

Table 10: Algorithm of aggregation (only for the first 3 lines)

```
FOR EACH page IN all pages
    read-in 3 lines (line1, line2, lin3)
    check {font_size_combination, spacings_condition} (line1, line2, lin3)
    previousSpacing = 0.0

    CASE font_size_combination = AAA:
        IF(spacings_condition == SAME_SPACINGS)
            createBlock (line1, line2, lin3)
            previousSpacing = spacing (line2, line3)
        ELSE IF(spacings_condition == NEGATIVE_SPACING_1 OR LARGER_THEN_SMALLER)
            createBlock (line1)
            createBlock (line2, lin3)
            previousSpacing = 0.0
        ELSE IF(spacings_condition == NEGATIVE_SPACING_2 OR SMALLER_THEN_LARGER)
            createBlock (line1, line2)
            createBlock (lin3)
            previousSpacing = 0.0
    CASE font_size_combination = AAB:
            createBlock (line1, line2)
            createBlock (lin3)
            previousSpacing = 0.0
    CASE font_size_combination = ABB:
            createBlock (line1)
            createBlock (line2, lin3)
            previousSpacing = 0.0
    CASE font_size_combination = ABA:
            createBlock (lin1)
            createBlock (lin2)
            createBlock (lin3)
            previousSpacing = 0.0
    go to the Algorithm in Table 11
go to next page (page id = page id + 1)
```

Table 11: Algorithm of aggregation (for the rest of lines)

```
read-in 3 lines (line1, line2, lin3)
check {font_size_combination, spacings_condition} (line1, line2, lin3)
CASE font_size_combination = AAA:
    IF (spacings_condition == SAME_SPACINGS)
        appendToBlock (line2, line3)
        previousSpacing = spacing (line2, line3)
    ELSE IF (spacings_condition == NEGATIVE_SPACING_1 OR LARGER_THEN_SMALLER)
        CASE NEGATIVE_SPACING_1:
            createBlock (line2, line3)
            previousSpacing = 0.0
        CASE LARGER_THEN_SMALLER:
            createBlock (line2)
            previousSpacing = 0.0
            go to next round (cursor = cursor +1)
    ELSE IF (spacings_condition == NEGATIVE_SPACING_2 OR SMALLER_THEN_LARGER)
        appendToBlock (line2)
        createBlock (line3)
CASE font_size_combination = AAB:
    IF (previousSpacing == 0.0 OR previousSpacing == spacing (line1, line2))
        appendToBlock (line2)
    ELSE
        createBlock (line2)
    createBlock (lines)
    previousSpacing = 0.0
CASE font_size_combination = ABB:
    IF (spacing (line2, line3) > the most frequent spacing)
        createBlock (line2)
        createBlock (line3)
    ELSE
        createBlock (line2, line3)
    previousSpacing = 0.0
CASE font_size_combination = ABA AND ABC:
    createBlock (line2)
    createBlock (line3)
    previousSpacing = 0.0
go to next round (cursor = cursor +2)
```

4.2 Detection of Logical Structure

Our research aims to detect the most important items of the logical structure, including the title, authors and affiliation, abstract heading, abstract, section headings, and body text by applying heuristic rules. These rules are based on the unique characteristics of each logical entity and compared with the statistics of the characteristic of the whole document. When choosing rules to be applied, we try to be as general as possible to accommodate the majority of document we observe from the corpus. Therefore, it is hardly possible to find a single set of rules that apply to all the variations in the format of the conference papers. This section first describes the detection sequence of the above-mentioned logical entities, followed by seven sub-sections discussing the details of the detection of them.

The detection of logical structure is based on the outcome from Phase I, which is *XML Physical Block*. This file contains blocks of words with homogeneous physical features on each page. The aim of the detection algorithm is to specify a logical label to each of blocks. The detection is then divided into two categories: unique entities and multiple-occurrence entities. Unique entities include title, abstract heading, and abstract, while multiple-occurrence entities include authors, affiliations, section-headings, page numbers, and paragraphs. Since unique entities only appear once and thus can be good benchmarks for the detection of other entities, they are detected first, followed by the detection of multiple-occurrence entities. The detection sequence of logical entities is as follows: 1) title, 2) abstract heading, 3) abstract, 4) affiliations, 5) authors, 6) page numbers, and 7) section headings.

4.2.1 Detection of Title

The detection of title in our research only considers the dominant font size and the position of a block. First of all, the title is always located on the first page. Then we observed the majority of development set from the corpus and found that font size of titles is almost the largest one through the entire document. To increase the detection accuracy, we also add one rule that the title should always be on the upper half of the first page. Table 12 illustrates this simple algorithm.

Table 12: Algorithm of detecting the title

```
FOR EACH block IN first page
    IF ( the block's dominant font size == the largest font size AND block is on upper page)
        annotateLogicalType (block, "title")
        stop
go to next block (block id = block id + 1)
```

4.2.2 Detection of Abstract Heading and Abstract

The detection of abstract heading is as straight-forward as that of the title. The algorithm shown in Table 13 first considers a block from the first page if its dominant font size is larger than the most frequent font size, which is the font size of context, and the content equals to "abstract" as well as its location is on the upper page, then the block is annotated as the abstract heading. Since there are still quite a few documents whose font size of abstract heading is as large as that of context, the algorithm continues to search again without requiring the font size condition.

Table 13: Algorithm of detecting the abstract heading

```
FOR EACH block IN first page
    IF ( the block's dominant font size > the most frequent font size AND the content equals to "abstract"
        AND block is on upper page)
        annotateLogicalType (block, "abstract-heading")
        stop
go to next block (block id = block id + 1)

IF NOT found
    FOR EACH block IN first page
        IF (the content equals to "abstract" AND block is on upper page AND number of line ==1)
        annotateLogicalType (block, "abstract-heading")
        stop
    go to next block (block id = block id + 1)
```

The detection of abstract comes after the abstract heading, which assumes the abstract is the block next to the abstract heading. This can only be true when the aggregation of homogeneous block on abstract is correct. In some cases, we found the abstract is divided into two or more block due to the variation of font size and spacing occurred in those abstract. More considerations need to be taken into account to fix this problem in the future. For example, it can be applied a rule saying that those blocks between abstract heading and the first section heading can be abstract. However, we recommend revising the aggregation algorithm to minimize the possibility of this error.

4.2.3 Detection of Authors and Affiliations

There are three major types of the block of authors and affiliations in the corpus as shown in Figure 15. To adapt this variation, the detection of authors and affiliations comes in a fixed order. First, the algorithm of detecting affiliations assumes those blocks between title and abstract heading are mixture candidates of authors and affiliation, and annotated those blocks with affiliation. Another algorithm continues to detect the authors using an exclusion

set of string, including keywords of organization, such as “department”, “center”, “university”, “laboratory”, “of”, etc., country names as shown in the address, and certain characters appeared in the e-mail address, such as “@”, “{“, or ”}”. The contents of those blocks annotated with affiliation are examined to check against the exclusion set of string. Those blocks without matching any item in the exclusion set are then annotated with author. Table 14 and 15 show the pseudo-codes of these two algorithms.

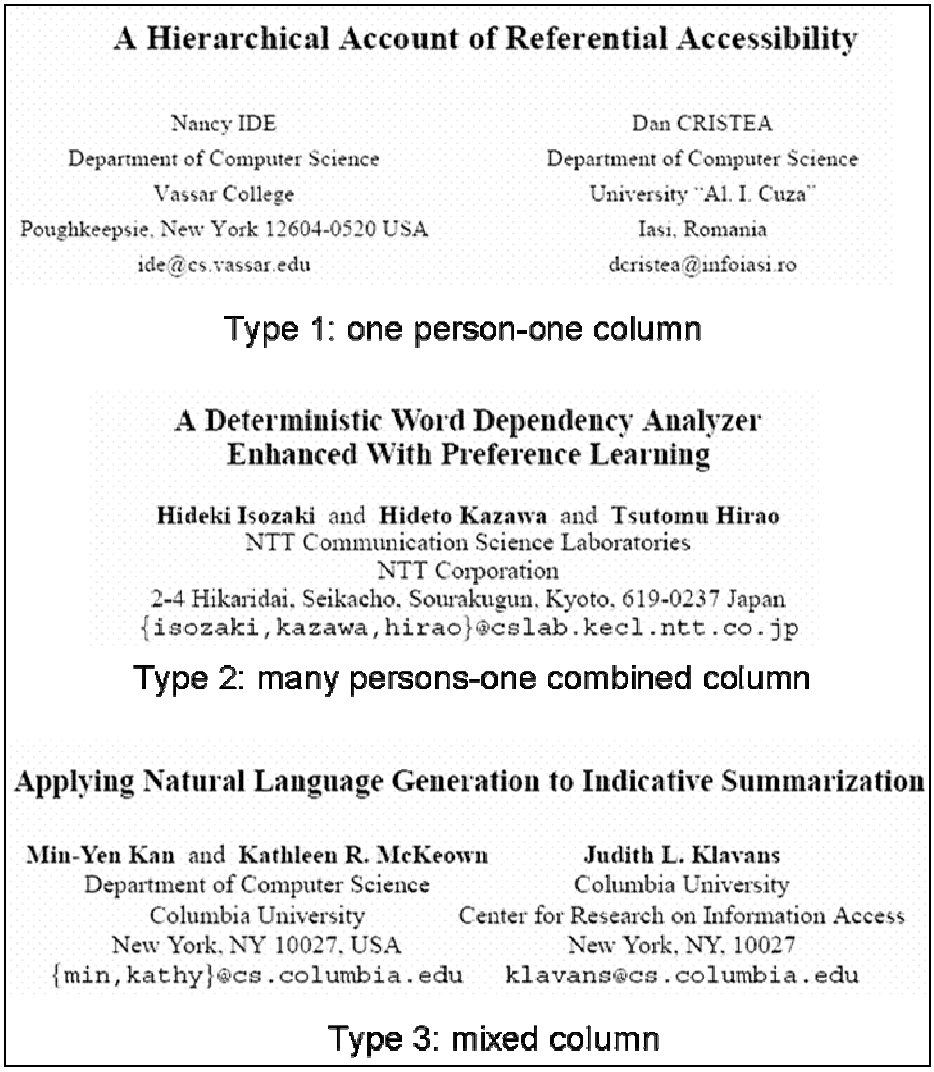


Figure 15: Types of block of authors and affiliations

Table 14: Algorithm of detecting affiliations

```
find block id of the title (title_id) and block id of the abstract heading (abstract_heading_id)
FOR EACH block IN first page
    IF (the block id falls between title_id AND abstract_heading_id)
        annotateLogicalType (block, “affiliation”)
        IF (the block id == abstract_heading_id)
            stop
go to next block (block id = block id + 1)
```

Table 15: Algorithm of detecting authors

```
find block id of the title (title_id) and block id of the abstract heading (abstract_heading_id)
FOR EACH block IN first page
    IF (the block id falls between title_id AND abstract_heading_id)
        IF (block's contents do not contain any element in the exclusion set)
            annotateLogicalType (block, "author")
        IF (the block id == abstract_heading_id)
                stop
go to next block (block id = block id + 1)

exclusion set: { "university", "center", "department", "research", "laboratory", "computer", "science",
"computing", "computational" ,"division" ,"information", , "group", "communication", "of", "@", "{", "}" }
```

4.2.4 Detection of Page Numbers

The detection of page numbers considers several physical features. First, the dominant font size is larger than the smallest font size, which excludes footnotes and footers whose font sizes are usually the smallest ones. Second, the block always falls on the lower page. Third, the count of words equals to one and the block's content should be numeric. For most cases in the ACL Anthology corpus, the page number is centered, and thus the algorithm assumes the block of page number should fall between the page's center with 20 pixels in both sides. Table 16 shows the algorithm of detecting page numbers.

Table 16: Algorithm of detecting page numbers

```
FOR EACH page IN all pages
    FOR EACH block IN page
        IF (the dominant font size > the smallest font size AND the block is on the lower page
            AND the x-coordinate of the block's center falls in the range of page's center ± 20.0
            AND the count of words in block == 1 AND block's content is numeric)
            annotateLogicalType (block, "page-number")
    go to next block (block id = block id + 1)
go to next page (page id = page id + 1)
```

4.2.5 Detection of Section Headings

The detection of section headings relies on the dominant font size and their common characteristic of starting with a number. The algorithm also considers those section headings that do not start with a number. Table 17 shows the pseudo-codes of the algorithm. It assumes the dominant font size is larger than the most frequent font size, which is the font size of the context. This assumption obviously does not match a minority of document whose font size of section heading is as large as that of context. Therefore,

there should be other rules to address this flaw in the future. The algorithm then examines the first word of the block. If the first word is numeric or the block contents contain any of keywords such as introduction, conclusion, reference, and acknowledge, then the block could be a section heading.

Table 17: Algorithm of detecting section headings

```
FOR EACH page IN all pages
    FOR EACH block IN page
        find the first word of the block (first_word)
        IF (the dominant font size of block > the most frequent font size AND
            (the first_word is numeric OR the block contents contain any of element in heading set))
            annotateLogicalType (block, “section-heading”)
    go to next block (block id = block id + 1)
go to next page (page id = page id + 1)

exclusion set: { "introduction", "conclusion”, “reference”, “acknowledge” }
```

4.3 Output Files

There are four different output files generated by our algorithms, two of which are in XML format and the other two, HTML. This section describes the structure for each output file format as well as its purpose to provide an overview of outcomes by this research. In this section, ***<text>***, ***<line>***, and **<page>** refer to mark-up elements or tags in those XML files mentioned above.

4.3.1 XML Source by Line

XML Source by Line is derived from *XML Source by Text* by grouping the **<text>** elements with the same attribute values for the y-coordinate together. **<text>** elements in the same group are appended under a newly created tag **<line>**, which is inserted under the existing **<page>** tag, with extra attributes, such as *dominantFont, dominantFontsize, left, right, words,* and *y-position.* The values of the above attributes are summary information of each line expressing collectively physical features or statistics of words in the same line. For example, the attribute *dominantFont* specifies the most frequent font among the **<text>** elements in that line, while the attribute *words* specifies the number of **<text>** elements and attributes *left* and *right* specify the horizontal margins for that line. Figure 16 is an example of *XML Source by Line* derived from the example *XML Source by Text* shown in Figure 11. One example is included in Appendix A.2.

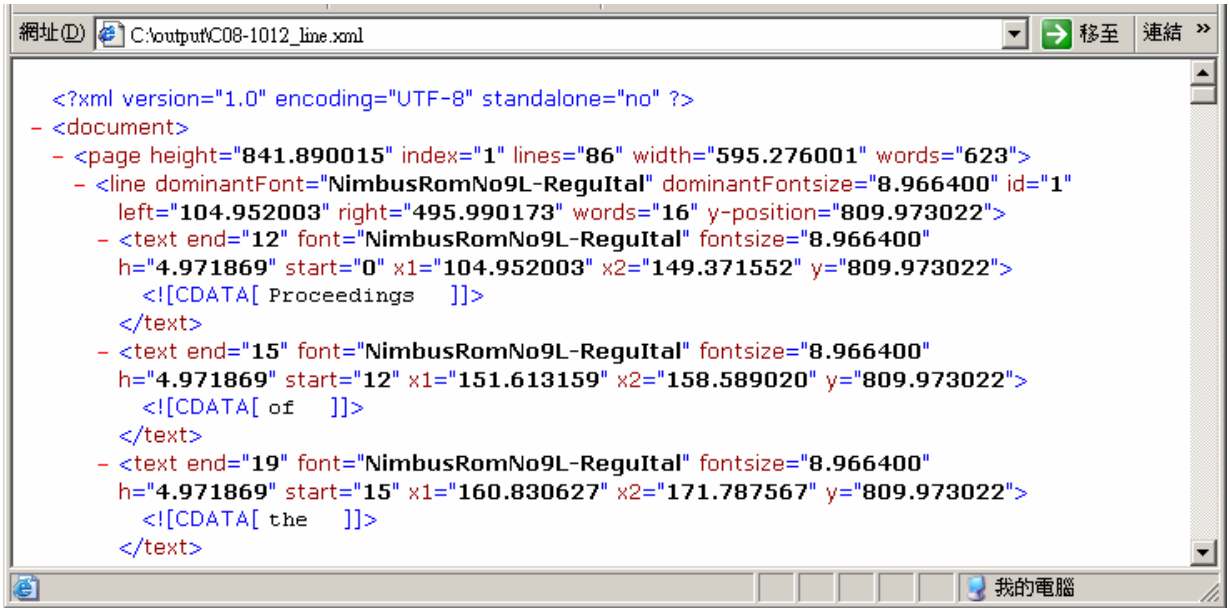


Figure 16: An example of XML Source by Line

4.3.2 XML Physical Blocks

XML Physical Blocks is derived from the *XML Source by Line* by applying the algorithm of aggregation introduced in Section 4.1. Figure 17 is an example of *XML Physical Blocks* derived from the example *XML Source by Line* shown in Figure 16. In this version, a newly created tag **<block>** is inserted under the existing tag **<page>** containing some **<line>** elements according to the algorithm, with several attributes, such as *marginLeft*, *marginRight*, *marginTop*, *marginBottom*, and lines. Similar to the attributes of **<line>** for its **<text>** elements, these attributes specify the summary information about these **<line>** elements aggregated as a block. One example is included in Appendix A.3.

4.3.3 XML Logical Blocks

XML Logical Blocks is the same file format as *XML Physical Blocks*, but annotated with logical labels according to the algorithm discussed in Section 4.2, by adding a new attribute *logicalType* to the tag **<block>**. Figure 18 is an example of *XML Logical Blocks* files annotated with a logical label *title* from the example of *XML Physical Blocks* shown in Figure 17. Except for this attribute appeared in most of **<block>** element, the rest of structure is the same as *XML Physical Blocks*. One example is included in Appendix A.3.

The two XML files introduced in this section and in Section 4.3.2 are the major outcomes of our research. With the annotated information for each block of texts, these files can be processed by machines for more creative uses, such as search or in-depth analysis.

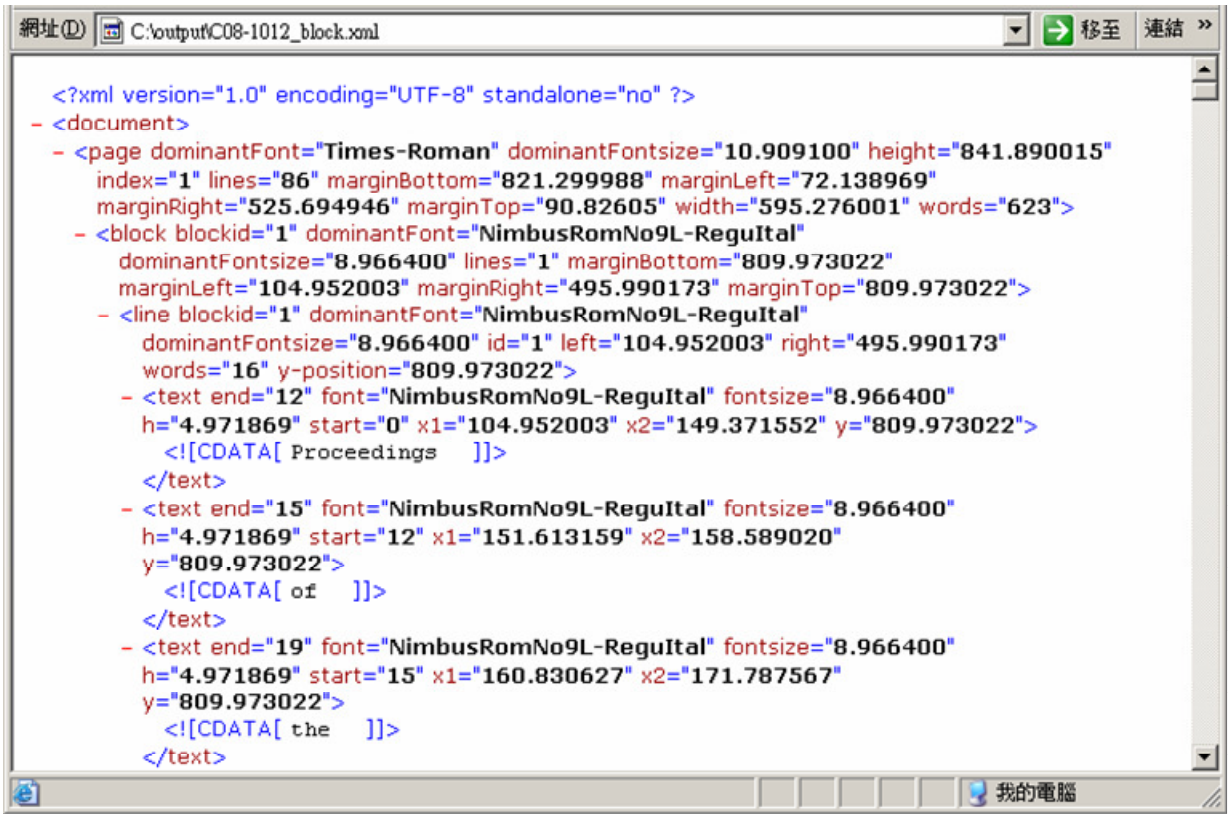


Figure 17: An example of XML Physical Blocks

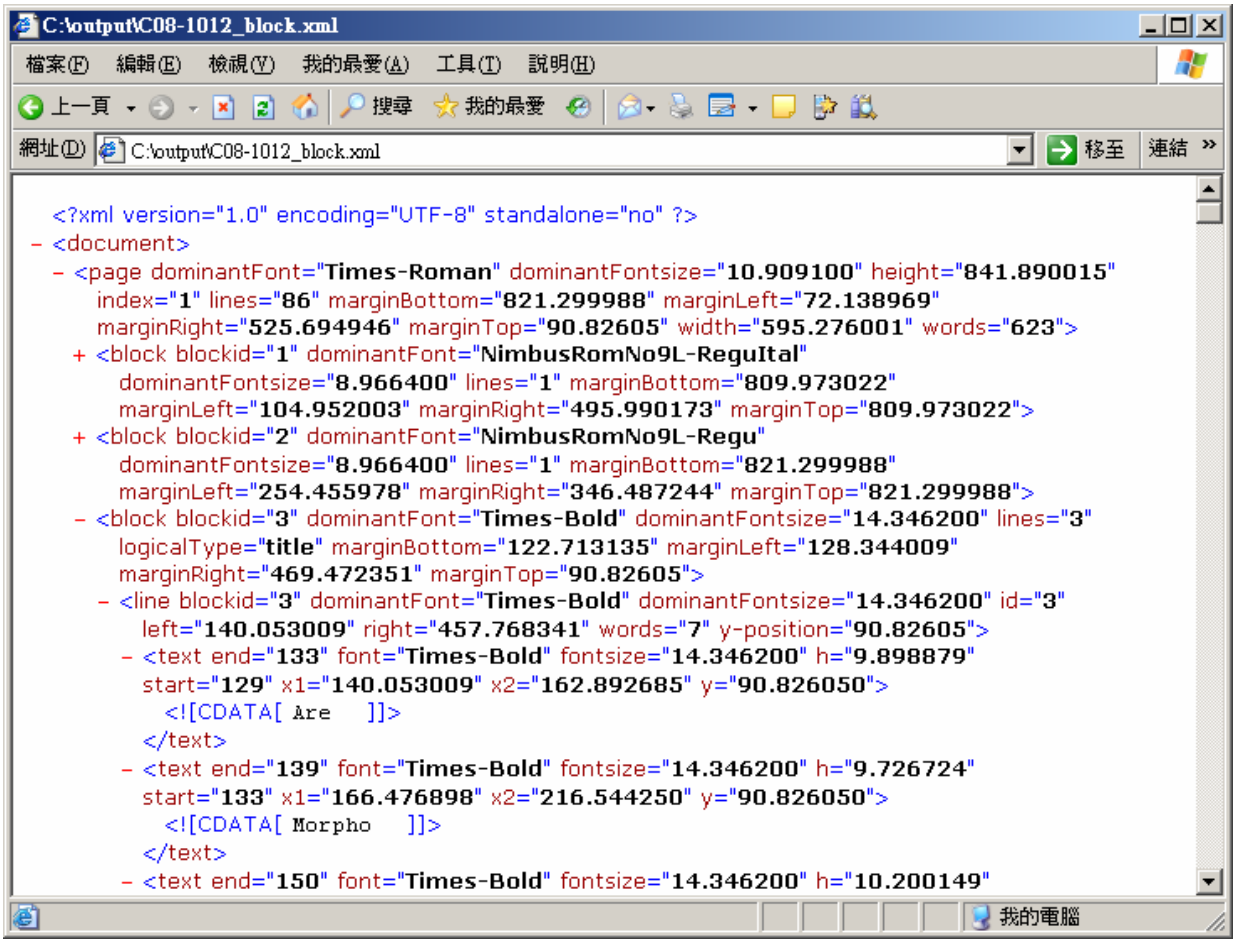


Figure 18: An example of XML Logical Blocks

4.3.4 HTML Physical Structure

This research also produces two other HTML file for human evaluation of the two major XML formats. The first file is *HTML Physical Structure*, transformed by applying an XML Stylesheet (XSLT) on *XML Physical Blocks*. It shows the result of block aggregation in the web browser with a line break between each block of texts. Figure 19 is an example of *HTML Physical Structure* derived from the example *XML Physical Blocks* shown in Figure 17. From this view of aggregation results, it is straightforward to check how successful the physical block aggregation algorithm is doing during development.The XML Stylesheet to transform *XML Physical Blocks* into HTML *Physical Structure* is included in Appendix A.4.

4.3.5 HTML Logical Structure

By applying another XSLT stylesheet on the *XML Logical Blocks* file from Figure 18, *HTML Logical Structure*, as shown in Figure 20, is created. It gives the developer with a clear view of the logical structure detection output. The stylesheet gives each logical entity a distinct format shown in the web browser, such as the centered bold title, centered authors, affiliations and abstract heading, and table of contents of section headings as hyperlinks. The XML Stylesheet to transform *XML Logical Blocks* into HTML *Logical Structure* is included in Appendix A.5.This file is produced mainly to assist the manual evaluation process discussed in Section 6.

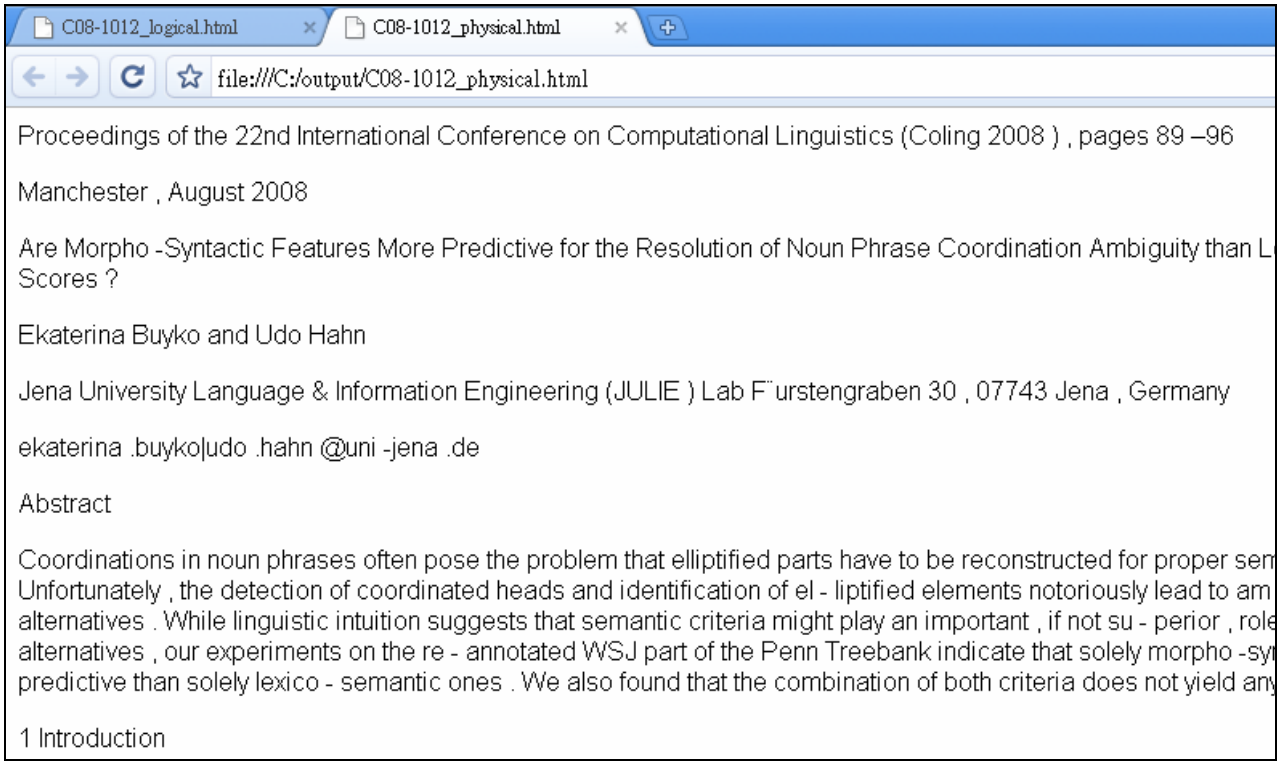


Figure 19: An example of HTML Physical Structure

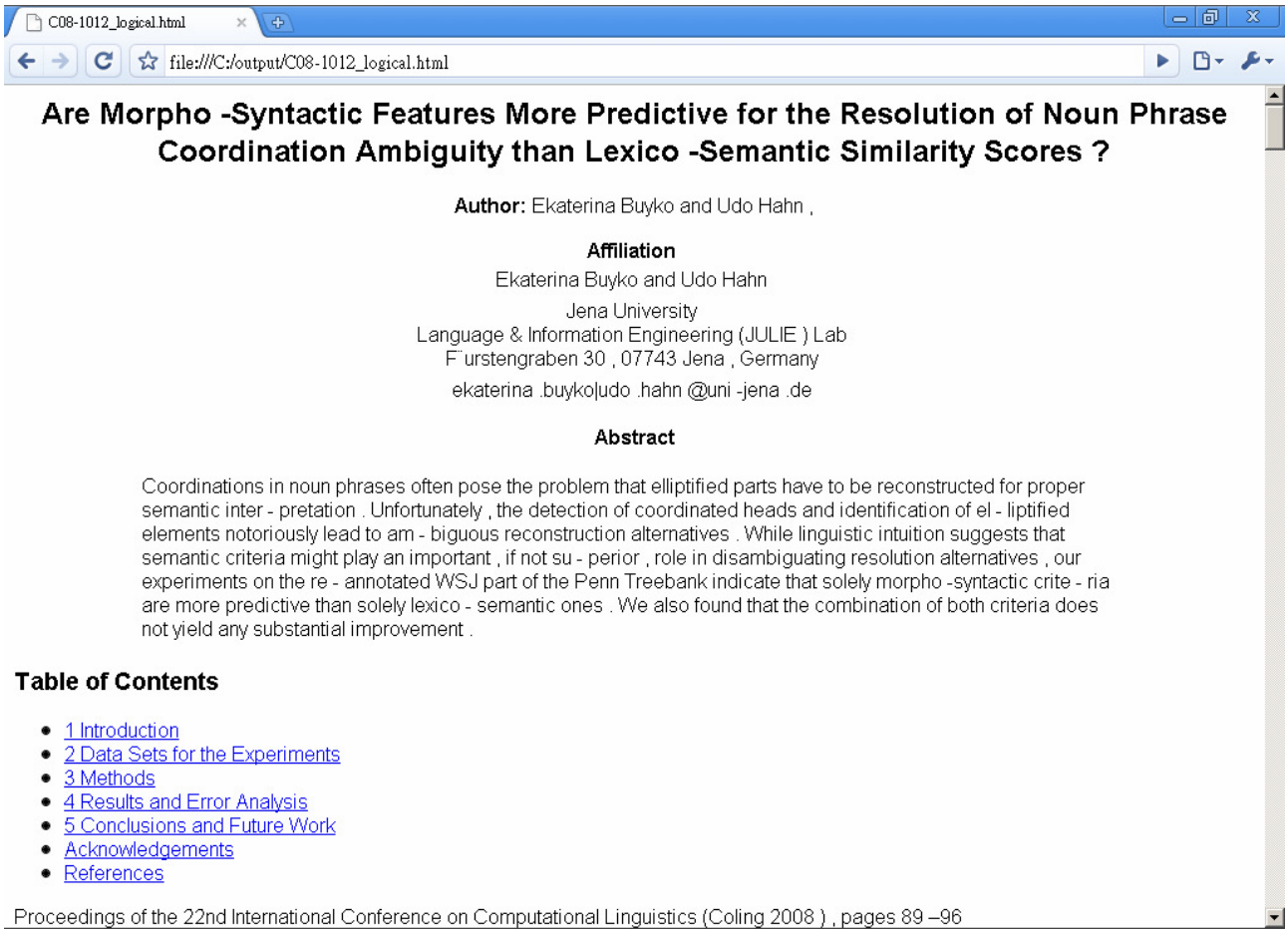


Figure 20: An example of HTML Logical Structure

5 Implementation

5.1 System Architecture

Inferring Document Structure System (IDSS) implemented the above-mentioned algorithms using Java (JDK1.6) on Windows 2003 Server. To obtain the flexibility of change and software extensibility, the design of system architecture is based on the concept of decoupling and modularization. Figure 21 is the class diagram of the system, including two Java interfaces and 15 major classes. Table 18 lists the functionalities for all interfaces and classes.

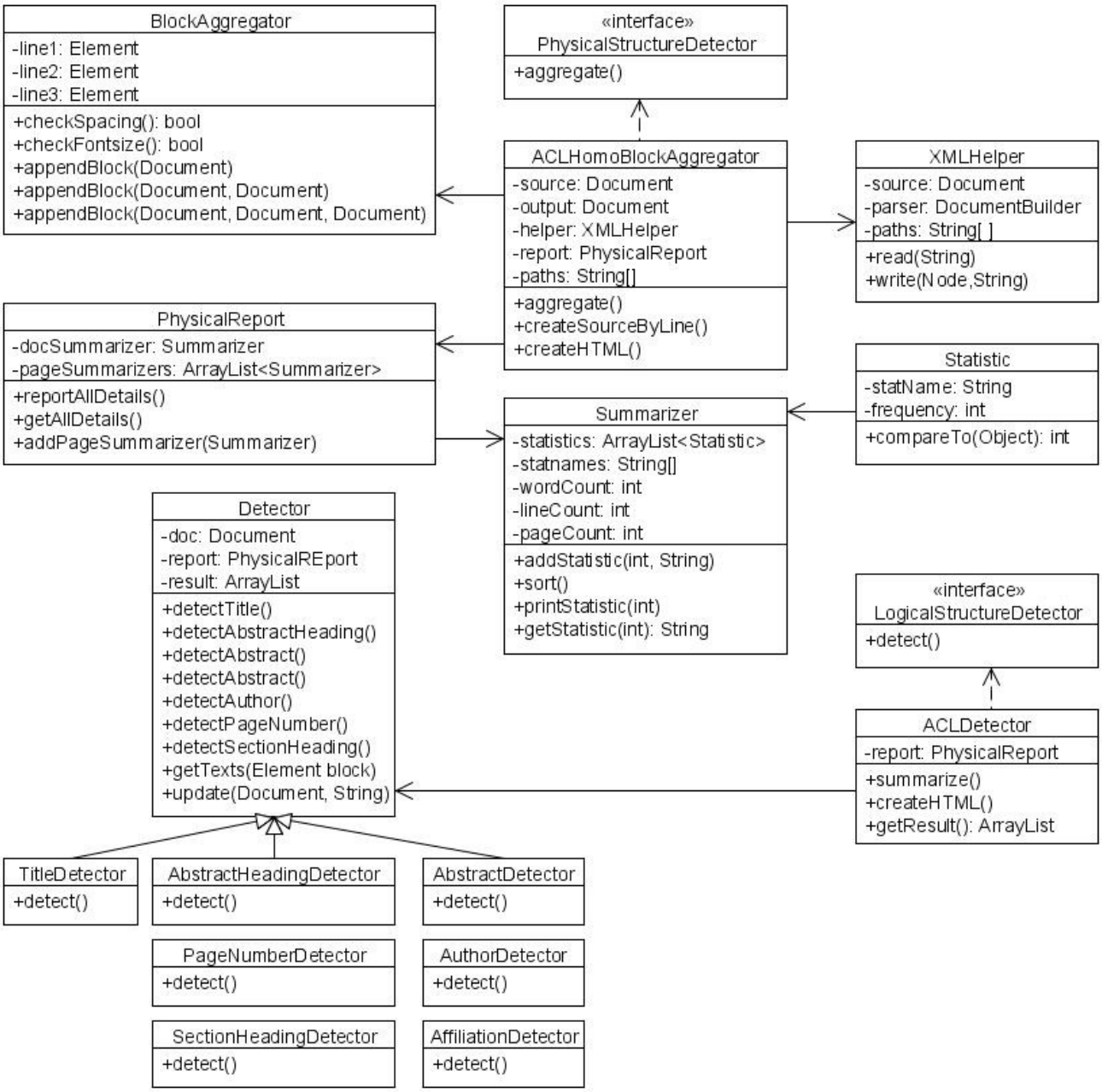


Figure 21: Class diagram of the Inferring Document Structure System (IDSS)

Table 18: Algorithm of detecting section headings

Class/Interface Name	Functionality
PhysicalStructureDetector*	Interface for aggregating homogeneous blocks
LogicalStructureDetector*	Interface for detecting logical structure
ACLHomoBlockAggregator	An implementation of PhysicalStructureDetector specific to aggregate homogeneous blocks from ACL Anthology corpus. The algorithm of aggregation is defined in this class. It provides methods to summarize the statistics of physical feature from XML Source by Text, and to create XML Source by Line, XML Physical Blocks, and HTML Physical Structure.
ACLDetector	An implementation of LogicalStructureDetector specific to detect logical structure from ACL Anthology corpus. It provides methods to summarize the statistics of physical feature from XML Physical Blocks, to detect logical structure with assistance from class Detector, and to create HTML Logical Structure.
BlockAggregator	A helper class to assist class ACLHomoBlockAggregator in checking font sizes and spacing of imported line elements as well as create new blocks containing associated line elements according to the algorithm of aggregation.
Summarizer	A helper class to assist classes ACLHomoBlockAggregator and ACLDetector in summarizing the statistics of physical feature from XML Source by Text and XML Physical Blocks.
PhysicalReport	A report class to report summary information of a processing document derived from classes ACLHomoBlockAggregator and ACLDetector, such as the most frequent font size, the largest/smallest font size, the most frequent spacing, and the number of lines, words for each page as well as for the entire document. Many detections of logical structure rely on this summary information.
Statistic	A data wrapper class to hold the frequency for each physical feature, in a pair of {feature, frequency}, such as the font size, font, and spacing.
XMLHelper	A helper class to assist in manipulating XML parsing and modification.
Detector	A helper class to assist class ACLDetector in detection of all logical entities including the title, abstract heading, abstract, affiliation, author, page number, and section heading. It fulfills the modularization and decoupling to maintain the flexibility and extensibility.

Class/Interface Name	Functionality
TitleDetector	It defines the algorithm of detecting the title (see Table 12)
AbstractHeadingDetector	It defines the algorithm of detecting the abstract heading (see Table 13)
AbstractDetector	It defines the algorithm of detecting the abstract
AffiliationDetector	It defines the algorithm of detecting affiliations (see Table 14)
AuthorDetector	It defines the algorithm of detecting the author (see Table 15)
PageNumberDetector	It defines the algorithm of detecting page numbers (see Table 16)
SectionHeadingDetector	It defines the algorithm of detecting section headings (see Table 17)
AutoRun	The main access entry of the system with a user interface for testing functionalities and evaluating performance of aggregation and detection.
ConfigFrame	A user interface for configuring system parameters such as the source path, output path, and the number of documents randomly selected for evaluation.

*: Java Interface

5.2 User Interface

Figure 22 is the main user interface for testing functionalities and evaluating performance of aggregation and detection. It separates each functionality in an independent button, such as analyze document (it creates XML Source by Text), aggregate blocks (it create XML Physical Blocks), view physical result (it creates HTML Physical Structure), detect layout (it annotates logical entities to generate XML Logical Blocks), and view logical structure (it creates HTML Logical Structure). It also provides a batch functionality to execute all steps in the button Batch Detection toward a single document specified in the text box of upper screen.

The user interface also provides a functionality of evaluation. By clicking the button Detection Evaluation, the system randomly selects a fixed number of documents from the appointed directory containing all the XML Source by Text files. Figure 23 is the user interface for specifying system parameters such as the source path, output path, and the number of documents randomly selected for evaluation. One example of detection summary with only two selected documents is included in Appendix A.6.

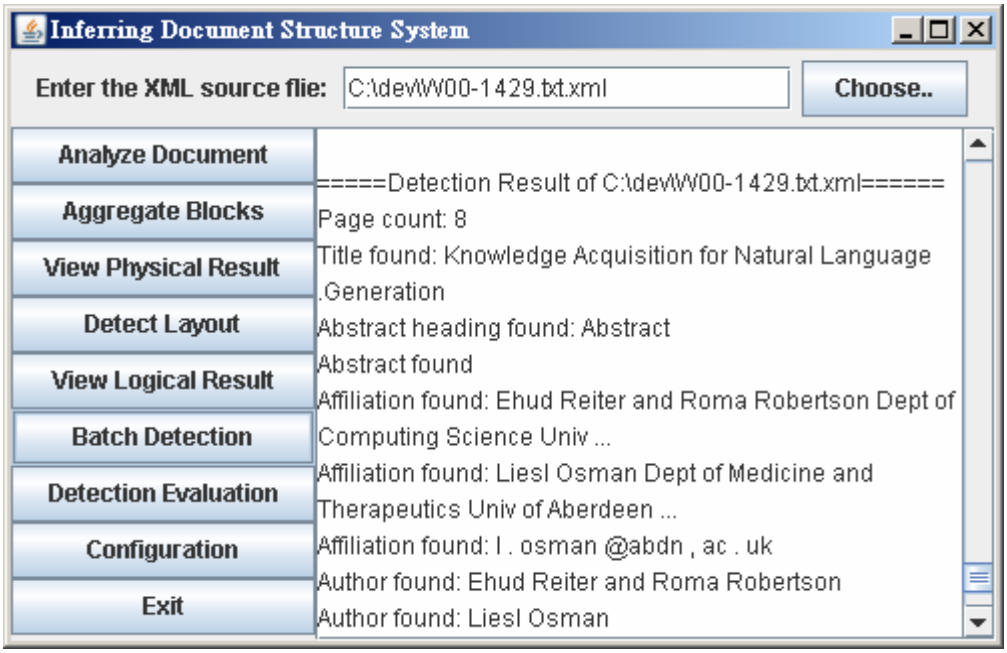


Figure 22: Main frame of IDSS

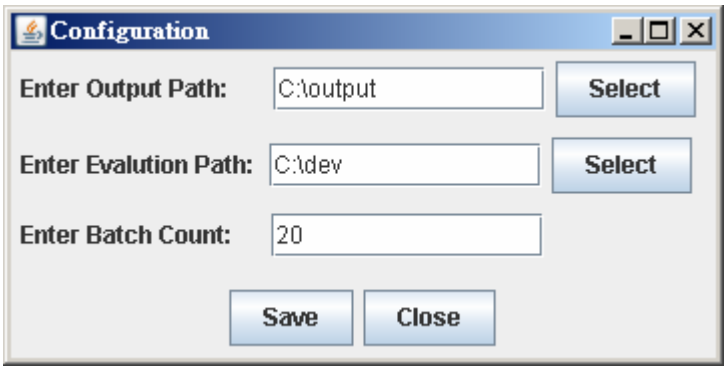


Figure 23: Configuration frame of IDSS

6 Conclusion

6.1 Results

This research uses the ACL Anthology corpus both for development and testing. Around 10 percent of the corpus was selected as the development set, accounting for 572 academic papers roughly evenly distributed over 13 conferences and 1 journal since 2000. Another 10 percent of unseen corpus is selected as the test set for a final evaluation of detection accuracy. Due to constraints on time and resources in this project, we were not able to perform a full evaluation on the test set. Instead, a preliminary evaluation was be done by manually comparing the logical structure of the final HTML files to the original PDF document for 40 randomly selected articles neither used for development nor part of the test set. Table 19 summarizes the detection results for title, author and affiliation, abstract heading, abstract, section heading, and page number for these 40 documents. For the time being, the author and affiliation are detected as one block due to their large variation in format.

From the summary, we can see the system obtains fairly high accuracy when detecting title (97.5%), abstract heading (90%), and abstract (90%). The accuracies for authors-affiliation, page numbers, and section headings are lower. Generally speaking, the accuracy of detection is satisfactory considering the limited implementation time.

Table 19: Summary of detection results out of 40 randomly selected documents

Error Type	Error Found	Accuracy of Detection
Incorrect title or missing title	1	97.5% (39/40)
Incorrect Abstract heading or Missing Abstract heading	4	90.0% (36/40)
Incorrect Abstract or Missing Abstract	4	90.0% (36/40)
Incorrect Affiliation(s) or Missing Affiliation(s)	11	72.5% (29/40)
Missing >50% of Page number(s) or Erroneous Page number(s) found	15	62.5% (25/40)
Missing >50% Section heading(s) or Erroneous Section heading(s) found	11	72.5% (29/40)

6.2 Error Analysis

When observing the details of detection results and looking at the original XML sources and PDF documents, we found several causes for the detection errors which can be solved in the near future as well as some defects due to the nature of format variation.

For example, the failure to detect section heading or sub-section headings can be improved by considering the length of lines and spacing before and after blocks. The detection of page numbers can be improved by calculating their positions and taking into account the total number of <page> tags. Furthermore, one abstract heading was groups into the same

block as its abstract text, which resulted from incorrect aggregation in Phase I. This can be solved by refining the aggregation algorithm to separate them as different homogeneous blocks.

Some erroneous detections of section-headings or page numbers mainly resulted from noise in the XML source files, such as incomplete table content and mathematic formula containing numbers and random characters. Rules dealing with noise can be introduced in order to obtain a higher accuracy here. However, this could also be resolved by improving Powley et al.'s [2008] extraction process from the original PDF documents. At this stage, we regarded the loss in accuracy due to these erroneous detections of noise as inevitable.

6.3 Future Work

Both algorithms for physical block aggregation and for logical structure detection need to be further refined until they obtain as high detection accuracy as possible for the 572 documents of the development set.

In the near future, the separation of author and affiliation, more accurate detections of section-headings, sub-section heading, and paragraph texts need to be achieved as mentioned in Section 4.2. Following this, noise such as table contents and mathematical formula should also be detected as such and removed or handled separately.

References

Powley, B., Dale, R. and Anisimoff I., 2009. Enriching a Document Collection by Integrating Information Extraction and PDF Annotation. *Proceedings of Document Recognition and Retrieval*.

Powley, B. and Dale, R. 2007. High Accuracy Citation Extraction and Named Entity Recognition. *2007 IEEE International Conference on Natural Language Processing and Knowledge Engineering*.

Powley, B. and Dale, R. 2007. Evidence-Based Information Extraction for High Accuracy Citation and Author Name Identification. *Proceedings of RIAO 2007: the 8th Conference on Large-Scale Semantic Access to Content.*

Conway, A. 1993. Page grammar and page parsing: A syntactic approach to document layout recognition, *Document Analysis and Recognition, Proceedings of the Second International Conference*, 761-764.

Fisher, L., Hinds, C. and D'Amato, P. 1990. A rule-based system for document image segmentation, *Pattern Recognition, Proceedings of the 10th International Conference*, 1: 16-21

Lee, K., Choy Y. and Cho S. 2000. Geometric structure analysis of document images: a knowledge-based approach, *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 22(11): 1224-1240.

Lee, K., Choy Y. and Cho S. 2003. Logical Structure Analysis and Generation for Structured Documents: A Syntactic Approach, *Knowledge and Data Engineering, IEEE Transactions*, 15(5): 1277-1294.

Mao, S., Rosenfeld, A. and Kanungo, T. 2003. *Document Structure Analysis Algorithms: A Literature Survey*, IBM Almaden Research Center, San Jose, USA.

Nagy, G., Seth, S. and Viswanathan, M. 1992. A prototype document image analysis system for technical journals, *Computer*, 25(7): 10–22.

Namboodiri A. and Jain A. 2007. Document Structure and Layout Analysis, in *Digital Document Processing: Major Directions and Recent Advances*, Springer-Verlag, London, 29-48.

Niyogi, D. and Srihari S. 1995. Knowledge-Based Derivation of Document Logical Structure, *Document Analysis and Recognition, Proceedings of the Third International Conference*, 1: 472-475.

Stehno, B. and Retti, G. 2003. Modeling the logical structure of books and journals using augmented transition network grammars, *Journal of Documentation*, 59(2): 69-83.

Tateisi, Y. and Itoh, N. 1994. Using stochastic syntactic analysis for extracting a logical structure from a document image, *Pattern Recognition, Conference B: Computer Vision and Image Processing., Proceedings of the 12th IAPR International Conference*, 2: 391-394.

Tsujimoto, S. and Asada, H. 1992. *Major Components of Complete Text Reading System, Proceedings of the IEEE*, 80(7): 1133-1149.

Appendix

A.1: XML Source by Text

Example Filename: W00-1429.txt..xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document>
<page height="842.000000" index="1" lines="96" width="596.000000" words="845">
    <text end="9" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="0" x1="140.160004" x2="213.271408" y="116.479980">
        <![CDATA[Knowledge ]]>
    </text>
    <text end="21" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="9" x1="213.271408" x2="297.851746" y="116.479980">
        <![CDATA[ Acquisition ]]>
    </text>
    <text end="25" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="21" x1="297.851746" x2="323.072205" y="116.479980">
        <![CDATA[ for ]]>
    </text>
    <text end="33" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="25" x1="323.072205" x2="381.467438" y="116.479980">
        <![CDATA[ Natural ]]>
    </text>
    <text end="42" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.795200"
                start="33" x1="381.467438" x2="453.484741" y="116.479980">
        <![CDATA[ Language ]]>
    </text>
    <text end="53" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.795200"
                start="42" x1="453.484741" x2="532.885132" y="116.479980">
        <![CDATA[.Generation ]]>
    </text>
</page>
</document>
```

A.2: XML Source by Line

Example Filename: W00-1429_line.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document>
<page height="842.000000" index="1" lines="96" width="596.000000" words="845">
    <line dominantFont="TimesNewRoman,Bold" dominantFontsize="15.600000" id="1"
        left="140.160004" right="532.885132" words="6" y-position="116.47998">
        <text end="9" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="0" x1="140.160004" x2="213.271408" y="116.479980">
            <![CDATA[Knowledge ]]>
        </text>
        <text end="21" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="9" x1="213.271408" x2="297.851746" y="116.479980">
            <![CDATA[ Acquisition ]]>
        </text>
        <text end="25" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="21" x1="297.851746" x2="323.072205" y="116.479980">
            <![CDATA[ for ]]>
        </text>
        <text end="33" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="25" x1="323.072205" x2="381.467438" y="116.479980">
            <![CDATA[ Natural ]]>
        </text>
        <text end="42" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.795200"
                start="33" x1="381.467438" x2="453.484741" y="116.479980">
            <![CDATA[ Language ]]>
        </text>
        <text end="53" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.795200"
                start="42" x1="453.484741" x2="532.885132" y="116.479980">
            <![CDATA[.Generation ]]>
        </text>
    </line>
</page>
</document>
```

A.3: XML Physical Blocks and Logical Blocks

Example Filename: W00-1429_block.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document>
<page dominantFont="TimesNewRoman" dominantFontsize="9.600000" height="842.000000"
    index="1" lines="96" marginBottom="801.679993" marginLeft="108.0"
    marginRight="568.398804" marginTop="116.47998" width="596.000000" words="845">
    <block blockid="1" dominantFont="TimesNewRoman,Bold" dominantFontsize="15.600000"
            lines="1" logicalType="title" marginBottom="116.47998" marginLeft="140.160004"
            marginRight="532.885132" marginTop="116.47998">
        <line blockid="1" dominantFont="TimesNewRoman,Bold" dominantFontsize="15.60000"
            id="1" left="140.160004" right="532.885132" words="6" y-position="116.47998">
            <text end="9" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="0" x1="140.160004" x2="213.271408" y="116.479980">
                    <![CDATA[Knowledge ]]></text>
            <text end="21" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="9" x1="213.271408" x2="297.851746" y="116.479980">
                    <![CDATA[ Acquisition ]]></text>
            <text end="25" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="21" x1="297.851746" x2="323.072205" y="116.479980">
                    <![CDATA[ for ]]></text>
            <text end="33" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.764001"
                start="25" x1="323.072205" x2="381.467438" y="116.479980">
                    <![CDATA[ Natural ]]></text>
            <text end="42" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.795200"
                start="33" x1="381.467438" x2="453.484741" y="116.479980">
                    <![CDATA[ Language ]]></text>
            <text end="53" font="TimesNewRoman,Bold" fontsize="15.600000" h="10.795200"
                start="42" x1="453.484741" x2="532.885132" y="116.479980">
                    <![CDATA[.Generation ]]></text>
        </line>
    </block>
</page>
</document>
```

A.4: HTML Physical Stylesheet

Filename: physical.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
            <html>
                <body>
                    <xsl:for-each select="/document/page/block">
                    <p>
                        <xsl:for-each select="./line">
                            <xsl:for-each select="./text">
                                <xsl:value-of select="."/>
                            </xsl:for-each>
                        </xsl:for-each>
                    </p>
                    </xsl:for-each>
                </body>
            </html>
        </xsl:template>
    </xsl:stylesheet>
```

A.5: HTML Logical Stylesheet

Filename: logical.xsl

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
    <html><body>
        <h2><center>
        <xsl:for-each select="document/page/block[@logicalType='title']">
            <xsl:for-each select="./line">
                <xsl:for-each select="./text">
                        <xsl:value-of select="."/>
                    </xsl:for-each>
            </xsl:for-each>
        </xsl:for-each>
        </center>    </h2>

        <center> <p><b>Author: </b>
        <xsl:apply-templates select="//block/line" mode="author"/>
        </p></center>

        <center><b>Affiliation</b><br/></center>
        <xsl:apply-templates select="//block" mode="affiliation"/>

        <xsl:apply-templates select="//block" mode="abstract-heading"/>
        <xsl:apply-templates select="//block" mode="abstract"/>

        <h3>Table of Contents</h3>
            <ul>
                <xsl:apply-templates select="//block" mode="toc"/>
            </ul>

        <p>
            <xsl:apply-templates select="//block" mode="body"/>
        </p>
    </body></html>
</xsl:template>
```

```
<xsl:template match="block/line[@logicalType='author']" mode="author">
    <xsl:for-each select="./text">
        <xsl:value-of select="."/>
    </xsl:for-each>,
</xsl:template>

<xsl:template match="block[@logicalType='affiliation']" mode="affiliation">
    <center><table width="90%"><tr><td align="center">
        <xsl:for-each select="./line">
            <xsl:for-each select="./text">
                <xsl:value-of select="."/>
            </xsl:for-each><br/>
        </xsl:for-each>
    </td></tr></table></center>
</xsl:template>

<xsl:template match="block[@logicalType='abstract-heading']" mode="abstract-heading">
    <center><p id="{ generate-id(.) }"><b>
    <xsl:for-each select="./line">
        <xsl:for-each select="./text">
            <xsl:value-of select="."/>
        </xsl:for-each>
    </xsl:for-each>
    </b></p></center>
</xsl:template>

<xsl:template match="block[@logicalType='abstract']" mode="abstract">
    <center><table width="80%"><tr><td>
        <xsl:for-each select="./line">
            <xsl:for-each select="./text">
                <xsl:value-of select="."/>
            </xsl:for-each>
        </xsl:for-each>
    </td></tr></table></center>
</xsl:template>
```

```
    <xsl:template match="block[@logicalType='section-heading']" mode="toc">
        <li>
            <a href="#{generate-id(.)}"><xsl:apply-templates/></a>
        </li>
    </xsl:template>

    <xsl:template match="block[@logicalType='section-heading']" mode="body">
        <p id="{ generate-id(.) }"><b>
            <xsl:for-each select="./line">
                <xsl:for-each select="./text">
                    <xsl:value-of select="."/>
                </xsl:for-each>
            </xsl:for-each>
        </b></p>
    </xsl:template>

    <xsl:template match="block[@logicalType='page-number']" mode="body" />
    <xsl:template match="block[@logicalType='title']" mode="body" />
    <xsl:template match="block[@logicalType='affiliation']" mode="body" />
    <xsl:template match="block[@logicalType='abstract-heading']" mode="body" />
    <xsl:template match="block[@logicalType='abstract']" mode="body" />

    <xsl:template match="block" mode="body">
        <p>
            <xsl:for-each select="./line">
                <xsl:for-each select="./text">
                    <xsl:value-of select="."/>
                </xsl:for-each>
            </xsl:for-each>
        </p>
    </xsl:template>

    <xsl:template match="*" mode="author" />
    <xsl:template match="*" mode="toc" />
    <xsl:template match="*" mode="affiliation" />
    <xsl:template match="*" mode="abstract-heading" />
    <xsl:template match="*" mode="abstract" />
    <xsl:template match="*" mode="body" />
</xsl:stylesheet>
```

A.6: Summary of Detection

Filename: summary090604_124810.txt

=====Detection Result of W05-0502.txt.xml======
Page count: 10
Title found: Simulating Language Change in the Presence of Non -Idealized Syntax ...
Abstract heading found: Abstract
Abstract found
Affiliation found: W . Garrett Mitchener
Affiliation found: Mathematics Department Duke University Box 90320 Durham ...
Affiliation found: wgm @math .duke .edu
Author found: W . Garrett Mitchener
Page number found: 10
Page number found: 11
Page number found: 12
Page number found: 13
Page number found: 14
Page number found: 15
Page number found: 16
Page number found: 17
Page number found: 18
Page number found: 19
Section heading found: 1 Introduction
Section heading found: 2 Linguistic specifics of the simulation
Section heading found: 3 Adaptation for Markov chain analysis
Section heading found: 4 Tweaking
Section heading found: 5 Results
Section heading found: 6 Discussion and conclusion
Section heading found: References

=====Detection Result of D07-1071.txt.xml======
Page count: 10
Title found: Online Learning of Relaxed CCG Grammars for Parsing to Logical ...
Abstract heading found: Abstract
Abstract found
Affiliation found: Luke S . Zettlemoyer and Michael Collins
Affiliation found: MIT CSAIL
Affiliation found: lsz @csail .mit .edu ,mcollins @csail .mit .edu
Author found: Luke S . Zettlemoyer and Michael Collins
Author found: MIT CSAIL
Page number found: 678
Page number found: 679
Page number found: 680
Page number found: 681
Page number found: 682
Page number found: 683
Page number found: 684
Page number found: 685
Page number found: 686
Page number found: 687
Section heading found: 1 Introduction
Section heading found: 2 Background
Section heading found: 3 Parsing Extensions : Combinators
Section heading found: 4 Learning
Section heading found: 5 Related Work
Section heading found: 6 Experiments
Section heading found: 7 Discussion
Section heading found: References