Structure-Preserving Pipelines for Digital Libraries

Massimo Poesio
University of Essex, UK and
Università di Trento, Italy

Eduard Barbu
Egon W. Stemle
Università di Trento, Italy

Christian Girardi
FBK-irst, Trento, Italy
cgirardi@fbk.eu

{massimo.poesio,eduard.barbu,egon.stemle}
@unitn.it

Abstract

Most existing HLT pipelines assume the input is pure text or, at most, HTML and either ignore (logical) document structure or remove it. We argue that identifying the structure of documents is essential in digital library and other types of applications, and show that it is relatively straightforward to extend existing pipelines to achieve ones in which the structure of a document is preserved.

1 Introduction

Many off-the-shelf Human Language Technology (HLT) pipelines are now freely available (examples include LingPipe,[1] OpenNLP,[2] GATE[3] (Cunningham et al., 2002), TextPro[4] (Pianta et al., 2008)), and although they support a variety of document formats as input, actual processing (mostly) takes no advantage of structural information, i.e. structural information is not used, or stripped off during preprocessing. Such processing can be considered safe, e.g. in case of news wire snippets, when processing does not need to be aware of sentence or paragraph boundaries, or of text being part of a table or a figure caption. However, when processing large documents, section or chapter boundaries may be considered an important segmentation to use, and when working with the type of data typically found in digital libraries or historical archives, such as whole books, exhibition catalogs, scientific articles, contracts we should keep the structure. At least three types of problems can be observed when trying to use a standard HLT pipeline for documents whose structure cannot be easily ignored:

- techniques for extracting content from plain text do not work on, say, bibliographic references, or lists;
- simply removing the parts of a document that do not contain plain text may not be the right thing to do for all applications, as sometimes the information contained in them may also be useful (e.g., keywords are often useful for classification, bibliographic references are useful in a variety of applications) or even the most important parts of a text (e.g., in topic classification information provided by titles and other types of document structure is often the most important part of a document);
- even for parts of a document that still can be considered as containing basically text—e.g., titles—knowing that we are dealing with what we will call here **non-paragraph** text can be useful to achieve good - or improve - performance as e.g., the syntactic conventions used in those type of document elements may be different - e.g., the syntax of NPs in titles can be pretty different from that in other sections of text.

In this paper we summarize several years of work on developing structure-preserving pipelines for different applications. We discuss the incorporation of

[1] http://alias-i.com/lingpipe/
[2] http://incubator.apache.org/opennlp/
[3] http://http://gate.ac.uk/
[4] http://textpro.fbk.eu/

Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities, pages 54–62,
Portland, OR, USA, 24 June 2011.

document structure parsers both in pipelines which the information is passed in BOI format (Ramshaw and Marcus, 1995), such as the TEXTPRO pipeline (Pianta et al., 2008), and in pipelines based on a standoff XML (Ide, 1998). We also present several distinct applications that require preserving document structure.

The structure of the paper is as follows. We first discuss the notion of document structure and previous work in extracting it. We then introduce our architecture for a structure-preserving pipeline. Next, we discuss two pipelines based on this general architecture. A discussion follows.

2 The Logical Structure of a Document

Documents have at least two types of structure[5]. The term **geometrical**, or **layout**, structure, refers to the structuring of a document according to its visual appearance, its graphical representation (pages, columns, etc). The **logical** structure (Luong et al., 2011) refers instead to the content's organization to fulfill an intended overall communicative purpose (title, author list, chapter, section, bibliography, etc). Both of these structures can be represented as trees; however, these two tree structures may not be mutually compatible (i.e. representable within a single tree structure with non-overlapping structural elements): e.g. a single page may contain the end of one section and the beginning of the next, or a paragraph may just span part of a page or column. In this paper we will be exclusively concerned with logical structure.

2.1 Proposals concerning logical structure

Early on the separation of presentation and content, i.e. of layout and logical structure, was promoted by the early adopters of computers within the typesetting community; well-known, still widely used, systems include the LaTeXmeta-package for electronic typesetting. The importance of separating document logical structure from document content for electronic document processing and for the document creators lead to the ISO 8613-1:1989(E) specification where *logical structure* is defined as the result of dividing and subdividing the content of a document into increasingly smaller parts, on the basis of the human-perceptible meaning of the content, for example, into chapters, sections, subsections, and paragraphs. The influential ISO 8879:1986 Standard Generalized Markup Language (SGML) specification fostered document format definitions like the Open Document Architecture (ODA) and interchange format, CCITT T.411-T.424 / ISO 8613.

Even though the latter format never gained wide-spread support, its technological ideas influenced many of today's formats, like HTML and CSS as well as, the Extensible Markup Language (XML), today's successor of SGML. Today, the ISO 26300:2006 Open Document Format for Office Applications (ODF), and the ISO 29500:2008 Office Open XML (OOXML) format are the important XML-based document file formats.

For the work on digital libraries the Text Encoding Initiative (TEI)[6],most notably, developed guidelines specifying encoding methods for machine-readable texts. They have been widely used, e.g. by libraries, museums, and publishers.

The most common logical elements in such proposals—chapters, sections, paragraphs, footnotes, etc.—can all be found in HTML, LaTeX, or any other modern text processor. It should be pointed out however that many modern types of documents found on the Web do not fit this pattern: e.g. blog posts with comments, and the structure of reply threads and inner-linkings to other comments cannot be captured; or much of wikipedia's non-paragraph text. (For an in depth comparison and discussion of logical formats, and formal characterizations thereof we suggest (Power et al., 2003; Summers, 1998).)

2.2 Extracting logical structure

Two families of methods have been developed to extract document structure. Older systems tend to follow the **template-matching** paradigm. In this approach the assignment of the categories to parts of the string is done by matching a sequence of hand crafted templates against the input string *S*. An instance of this kind of systems is DeLos (Derivation of Logical Structure) (Niyogi and Srihari, 1995) which uses control rules, strategy rules and knowl-

[5] other structure types include e.g. (hyper)links, cross-references, citations, temporal and spatial relationships

[6] `http://www.tei-c.org`

edge rules to derive the logical document structure from a scanned image of the document. A more elaborate procedure for the same task is employed by Ishitani (Ishitani, 1999). He uses rules to classify the text lines derived from scanned document image and then employs a set of heuristics to assign the classified lines to logical document components. The template based approach is also used by the ParaTools, a set of Perl modules for parsing reference strings (Jewell, 2000). The drawback of the template based approaches is that they are usually not portable to new domains and are not flexible enough to accommodate errors. Domain adaptation requires the devising of new rules many of them from scratch. Further the scanned documents or the text content extracted from PDF have errors which are not easily dealt with by template based systems.

Newer systems use supervised machine learning techniques which are much more flexible but require training data. Extracting document structure is an instance of (hierarchical) **sequence labeling**, a well known problem which naturally arises in diverse fields like speech recognition, digital signal processing or bioinformatics. Two kinds of machine learning techniques are most commonly used for this problem: Hidden Markov Models (HMM) and Conditional Random Fields (CRF). A system for parsing reference strings based on HMMs was developed in (Hetzner, 2008) for the California Digital Library. The system implements a first order HMM where the set of states of the model are represented by the categories in C; the alphabet is hand built and tailored for the task and the probabilities in the probability matrix are derived empirically. The system obtains an average F_1 measure of 93 for the Cora dataset. A better performance for sequence labeling is obtained if CRF replaces the traditional HMM. The reason for this is that CRF systems better tolerate errors and they have good performance even when richer features are not available. A system which uses CRF and a series of post-processing rules for both document logical structure identification and reference string parsing is ParsCit (Councill et al., 2008). ParsCit comprises three sub-modules: SectLabel and ParseHead for document logical structure identification and ParsCit for reference string parsing. The system is built on top of the well known CRF++ package.

The linguistic surface level, i.e. the linear order of words, sentences, and paragraphs, and the hierarchical, tree-like, logical structure also lends itself to parsing-like methods for the structure analysis. However, the complexity of fostering, maintaining, and augmenting document structure grammars is challenging, and the notorious uncertainty of the input demands for the whole set of stochastic techniques the field has to offer – this comes at a high computing price; cf. e.g.,(Lee et al., 2003; Mao et al., 2003). It is therefore not surprising that high-throughput internet sites like CiteSeerX[7] use a flat text classifier (Day et al., 2007).[8]

3 Digital Libraries and Document Structure Preservation

Our first example of application in which document structure preservation is essential are digital libraries (Witten et al., 2003). In a digital library setting, HLT techniques can be used for a variety of purposes, ranging from indexing the documents in the library for search to classifying them to automatically extracting metadata. It is therefore becoming more and more common for HLT techniques to be incorporated in document management platforms and used to support a librarian when he / she enters a new document in the library. Clearly, it would be beneficial if such a pipeline could identify the logical structure of the documents being entered, and preserve it: this information could be used by the document management platform to, for instance, suggest the librarian the most important keywords, find the text to be indexed or even summarized, and produce citations lists, possibly to be compared with the digital library's list of citations to decide whether to add them.

We are in the process of developing a Portal for Research in the Humanities (Portale Ricerca Umanistica-PRU). This digital library will eventually include research articles about the Trentino region from Archeology, History, and History of Art. So far, the pipeline to be discussed next has been used to include in the library texts from the Italian archeology journal *Preistoria Alpina*. One of our goals was to develop a pipeline that could be used

[7] `http://citeseerx.ist.psu.edu/`

[8] Still, especially multimedia documents with their possible temporal and spatial relationships might need more sophisticated methods.

whenever a librarian uploads an article in this digital library, to identify title, authors, abstract, keywords, content, and bibliographic references from the article. The implemented portal already incorporates information extraction techniques that are used to identify in the 'content' part of the output of the pipeline temporal expressions, locations, and entities such as archeological sites, cultures, and artifacts. This information is used to allow spatial, temporal, and entity-based access to articles.

We are in the process of enriching the portal so that title and author information are also used to automatically produce a bibliographical card for the article that will be entered in the PRU Library Catalog, and bibliographical references are processed in order to link the article to related articles and to the catalog as well. The next step will be to modify the pipeline (in particular, to modify the Named Entity Recognition component) to include in the library articles from other areas of research in the Humanities, starting with History. There are also plans to make it possible for authors themselves to insert their research articles and books in the Portal, as done e.g., in the Semantics Archive.[9].

We believe the functionalities offered by this portal are or will become pretty standard in digital libraries, and therefore that the proposals discussed in this paper could find an application beyond the use in our Portal. We will also see below that a document structure-sensitive pipeline can find other applications.

4 Turning an Existing Pipeline into One that Extracts and Preserves Document Structure

Most freely available HLT pipelines simply eliminate markup during the initial phases of processing in order to eliminate parts of the document structure that cannot be easily processed by their modules (e.g., bibliographic references), but this is not appropriate for the Portal described in the previous section, where different parts of the output of the pipeline need to be processed in different ways. On the other end, it was not really feasible to develop a completely new pipeline from scratch. The approach we pursued in this work was to take an existing pipeline and turn it into one which extracts and outputs document structure. In this Section we discuss the approach we followed. In the next Section we discuss the first pipeline we developed according to this approach; then we discuss how the approach was adopted for other purposes, as well.

Incorporating a document structure extractor in a pipeline requires the solution of two basic problems: where to insert the module, and how to pass on document structure information. Concerning the first issue, we decided to insert the document structure parser after tokenization but before sentence processing. In regards to the second issue, there are at present three main formats for exchanging information between elements of an HLT pipeline:

- inline, where each module inserts information in a pre-defined fashion into the file received as input;
- tabular format as done in CONLL, where tokens occupy the first column and each new layer of information is annotated in a separate new column, using the so-called IOB format to represent bracketing (Ramshaw and Marcus, 1995);
- standoff format, where new layers of information are stored in separate files.

The two main formats used by modern HLT pipelines are tabular format, and inline or standoff XML format. Even though we will illustrate the problem of preserving document structure in a pipeline of the former type the PRU pipeline itself supports tabular format and inline XML (TEI compliant).

The solution we adopted, illustrated in Figure 1, involves using **sentence headers** to preserve document structure information. In most pipelines using a tabular interchange information, the output of a module consists of a number of sentences each of which consists of

- a **header**: a series of lines with a hash character # at the beginning;
- a set of tab-delimited lines representing tokens and token annotations;
- an empty EOF line.

[9] http://semanticsarchive.net/

```
# FILE: 11
# PART: id1
# SECTION: title
# FIELDS: token   tokenstart  sentence  pos     lemma         entity    nerType
Spondylus         0            -         SPN     Spondylus     O         B-SITE
gaederopus        10           -         YF      gaederopus    O         O
,                 20           -         XPW     ,             O         O
gioiello          22           -         SS      gioiello      O         O
dell'             31           -         E       dell'         O         O
Europa            36           -         SPN     europa        B-GPE     B-SITE
preistorica       43           -         AS      preistorico   O         O
.                 55           <eos>     XPS     full_stop     O         O
# FILE: 11
# PART: id2
# SECTION: author
# FIELDS: token   tokenstart  sentence  pos     lemma         entity    nerType
MARIA             0            -         SPN     maria         B-PER     O
A                 6            -         E       a             I-PER     O
BORRELLO          8            -         SPN     BORRELLO      I-PER     O
&                 17           -         XPO     &             O         O
.                 19           <eos>     XPS     full_stop     O         O

(TEI compliant inline XML snippet:)
<text>
  <body>
    <div type="section" xml:lang="it">
      [...]
      <p id="p2" type="author">
        <s id="p2s1"><name key="PER1" type="person">MARIA A BORRELLO</name>&.</s>
      </p>
    </div>
  </body>
</text>
```

Figure 1: Using sentence headers to preserve document structure information. For illustration, the TEI compliant inline XML snippet of the second sentence has been added.

The header in such pipelines normally specifies only the file id (constant through the file), the number of the sentence within the file, and the columns (see Figure 1). This format however can also be used to pass on document structure information provided that the pipeline modules ignore all lines beginning with a hash, as these lines can then be used to provide additional meta information. We introduce an additional tag, SECTION, with the following meaning: a line beginning with `# SECTION:` specifies the position in the document structure of the following sentence. Thus for instance, in Figure 1, the line

```
# SECTION: title
```

specifies that the following sentence is a title.

5 An Pipeline for Research Articles in Archeology

The pipeline currently in use in the PRU Portal we are developing is based on the strategy just discussed. In this Section We discuss the pipeline in more detail.

5.1 Modules

The pipeline for processing archaeological articles integrates three main modules: a module for recovering the logical structure of the documents, a module for Italian and English POS tagging and a general Name Entity Recognizer and finally, a Gazetteer Based Name Entity Recognizer. The architecture of the system is presented in figure 2. Each module except the first one takes as input the output of the previous module in the sequence.

1. **Text Extraction**. This module extracts the text from PDF documents. Text extraction from PDF is a notoriously challenging task. We experimented with many software packages and obtained the best results with *pdftotext*. This is a component of XPDF, an open source viewer for PDF documents. *pdftotext* allows the extraction of the text content of PDF documents in a variety of encodings. The main drawback of the text extractor is that it does not always preserve the original text order.

2. **Language Identification**. The archeology repository contains articles written in one of the two languages: Italian or English. This module uses the TextCat language guesser[10] for guessing the language of sentences. The language identification task is complicated by the fact that some articles contain text in both languages: for example, an article written in English may have an Italian abstract and/or an Italian conclusion.

3. **Logical Structure Identification**. This module extracts the logical structure of a document. For example, it identifies important parts like the title, the authors, the main headers, tables or figures. For this task we train the SectLabel component of ParsCit on the articles in the archeology repository. Details on the training process, the tag set and the performance of the module are provided in section 5.2.

4. **Linguistic Processing**. A set of modules in the pipeline then perform linguistic processing on specific parts of the document (the Bibliography Section is excluded for example). First English or Italian POS is carried out as appropriate, followed by English or Italian NER. NER adaptation techniques have been developed to identify non-standard types entities that are important in the domain, such as *Archeological Sites* and *Archeological Cultures*. (This work is discussed elsewhere.)

5. **Reference Parsing**. This module relies on the output of ParsCit software to update the Archeology Database Bibliography table with the parsed references for each article. First, each parsed reference is corrected in an automatic post processing step. Then, the module checks, using a simple heuristic, if the entry already exists in the table and updates the table, if appropriate, with the new record.

Finally, the documents processed by the pipeline are indexed using the Lucene search engine.

5.2 Training the Logical Document Structure Identifier

As mentioned in Section 5, we use ParsCit to find the logical structure of the documents in the archeology

[10] `http://odur.let.rug.nl/~vannoord/TextCat/`

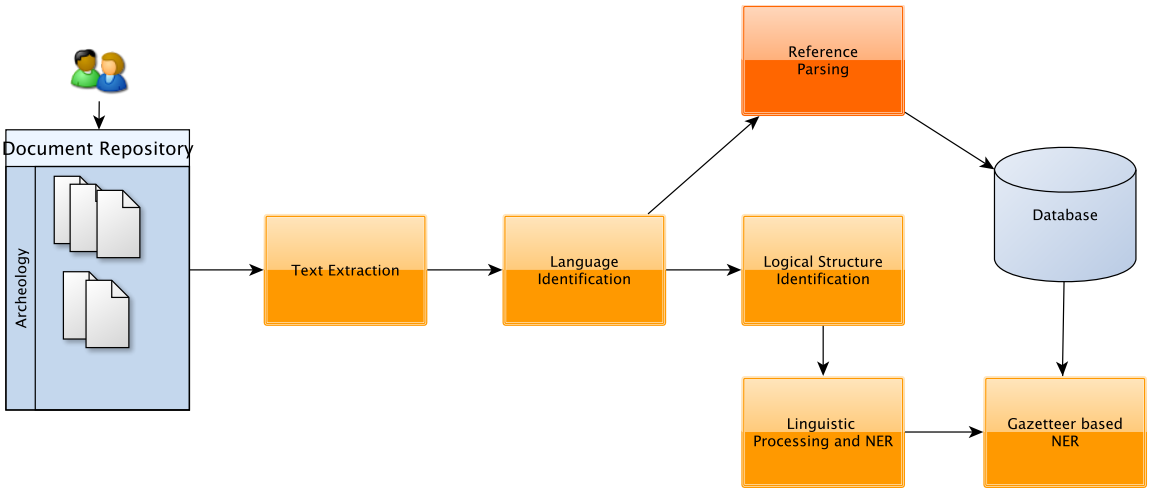

Figure 2: The pipeline of the system for PDF article processing in the Archeology Domain

domain. ParsCit comes with general CRF trained models; unfortunately, they do not perform well on archeology documents. There are some particularities of archeology repository articles that require the retraining of the models. First, as said before, the text extracted from PDF is not perfect. Second, the archeology articles contain many figures with bilingual captions. Third, the articles have portions of the texts in both languages: Italian and English. To improve the parsing performance two models are trained: the first model should capture the logical documents structure for those documents that have Italian as main language but might contain portions in English (like the abstract or summary). The second model is trained with documents that have English as main language but might contain fragments in Italian (like abstract or summary).

The document structure annotation was performed by a student in the archeology department, and was checked by one of the authors. In total 55 documents have been annotated (35 with Italian as main language, 20 with English as main Language). The tagset used for the annotation was specifically devised for archeology articles. However, as it can be seen below most of the devised tags can also be found in general scientific articles. In Table 1 we present the tag set used for annotation. The column "Tag Count" gives the number of each tag in the annotated documents.

In general the meaning of the tags is self-explanatory with the possible exception of the tag *VolumeInfo*, which reports information for volume the article is part of. An annotation example using this tag is: "<VolumeInfo> Preistoria Alpina v. 38 (2002) Trento 2002 ISSN 0393-0157 </VolumeInfo>". The volume information can be further processed by extracting the volume number, the year of the issue and the International Standard Serial Number (ISSN). To asses the performance of the trained models we performed a five fold cross-validation. The results are reported in the table 2 and are obtained for each tag using the F_1 measure (1):

$$F_1 = \frac{2 \times P \times R}{P + R} \quad (1)$$

The results obtained for the Archeology articles are in line with those obtained by the authors of ParsCit and reported in (Luong et al., 2011). The tag categories for which the performance of the system is bad are the multilingual tags (e.g. ItalianAbstract or Italian Summary in articles where the main language is English). We will address this issue in the future by adapting the language identifier to label multilingual documents. We also noticed that many mis-tagged titles, notes or section headers are split on multiple lines after the text extraction stage. The system performance might be further improved if a pre-processing step immediately after the text extraction is introduced.

Tag	Tag Count
ItalianFigureCaption	456
ItalianBodyText	347
EnglishFigureCaption	313
SectionHeader	248
EnglishTableCaption	58
ItalianTableCaption	58
Author	71
AuthorEmail	71
AuthorAddress	65
SubsectionHeader	50
VolumeInfo	57
Bibliography	55
English Summary	31
ItalianKeywords	35
EnglishKeywords	35
Title	55
ItalianSummary	29
ItalianAbstract	10
Table	25
EnglishAbstract	13
Note	18

Table 1: The tag set used for Archeology Article Annotation.

Tag	F_1
ItalianFigureCaption	70
ItalianBodyText	90
EnglishFigureCaption	71
SectionHeader	90
EnglishTableCaption	70
ItalianTableCaption	75
Author	72
AuthorEmail	75
AuthorAddress	73
SubsectionHeader	65
VolumeInfo	85
Bibliography	98
English Summary	40
ItalianKeywords	55
EnglishKeywords	56
Title	73
ItalianSummary	40
ItalianAbstract	50
Table	67
EnglishAbstract	50
Note	70

Table 2: The Precision and Recall for the trained models.

6 Additional Applications for Structure-Sensitive Pipelines

The pipeline discussed above can be used for a variety of other types of documents–archeology documents from other collections, or documents from other domains–by simply replacing the document structure extractor. We also found however that the pipeline is useful for a variety of other text-analysis tasks. We briefly discuss these in turn.

6.1 Blogs and Microblogging platforms

Content creation platforms like blogs, microblogs, community QA sites, forums, etc., contain user generated data. This data may be emotional, opinionated, personal, and sentimental, and as such, makes it interesting for sentiment analysis, opinion retrieval, and mood detection. In their survey on opinion mining and sentiment analysis Pang and Lee (2008) report that logical structure can be used to utilize the relationships between different units of content, in order to achieve a more accurate labeling; e.g. the relationships between discourse participants in discussions on controversial topics when responding are more likely to be antagonistic than to be reinforcing, or the way of quoting–a user can refer to another post by quoting part of it or by addressing the other user by name or user ID–in posts on political debates hints at the perceived opposite end of the political spectrum of the quoted user.

We are in the process of creating an annotated corpus of blogs; the pipeline discussed in this paper was easily adapted to pre-process this type of data as well.

6.2 HTML pages

In the IR literature it has often been observed that certain parts of document structure contain information that is particularly useful for document retrieval. For instance, Kruschwitz (2003) automatically builds domain models – simple trees of related terms – from documents marked up in HTML to assist users during search tasks by performing automatic query refinements, and improves users' experi-

ence for browsing the document collection. He uses term counts in different **markup contexts** like non-paragraph text and running text, and markups like bold, italic, underline to identify concepts and the corresponding shallow trees. However, this domain-independent method is suited for all types of data with logical structure annotation and similar data sources can be found in many places, e.g. corporate intranets, electronic archives, etc.

6.3 Processing Wikipedia pages

Wikipedia, as a publicly available web knowledge base, has been leveraged for semantic information in much work, including from our lab. Wikipedia articles consist mostly of free text, but also contain different types of structured information, e.g. infoboxes, categorization and geo information, links to other articles, to other wiki projects, and to external Web pages. Preserving this information is therefore useful for a variety of projects.

7 Discussion and Conclusions

The main point of this paper is to argue that the field should switch to structure-sensitive pipelines. These are particularly crucial in digital library applications, but novel type of documents require them as well. We showed that such extension can be achieved rather painlessly even in tabular-based pipelines provided they allow for meta-lines.

References

Isaac G. Councill, C. Lee Giles, and Min-Yen Kan. 2008. Parscit: An open-source crf reference string parsing package. In *Proceedings of the Language Resources and Evaluation Conference (LREC 08)*, May.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*.

Min-Yuh Day, Richard Tzong-Han Tsai, Cheng-Lung Sung, Chiu-Chen Hsieh, Cheng-Wei Lee, Shih-Hung Wu, Kun-Pin Wu, Chorng-Shyong Ong, and Wen-Lian Hsu. 2007. Reference metadata extraction using a hierarchical knowledge representation framework. *Decision Support Systems*, 43(1):152–167, February.

Erik Hetzner. 2008. A simple method for citation metadata extraction using hidden markov models. In *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '08, pages 280–284, New York, NY, USA. ACM.

Nancy Ide. 1998. Corpus encoding standard: SGML guidelines for encoding linguistic corpora. In *Proceedings of LREC*, pages 463–70, Granada.

Yasuto Ishitani. 1999. Logical structure analysis of document images based on emergent computation. In *Proceedings of International Conference on Document Analysis and Recognition*.

Michael Jewell. 2000. Paracite: An overview.

Udo Kruschwitz. 2003. An Adaptable Search System for Collections of Partially Structured Documents. *IEEE Intelligent Systems*, 18(4):44–52, July.

Kyong-Ho Lee, Yoon-Chul Choy, and Sung-Bae Cho. 2003. Logical structure analysis and generation for structured documents: a syntactic approach. *IEEE transactions on knowledge and data engineering*, 15(5):1277–1294, September.

Minh-Thang Luong, Thuy Dung Nguyen, and Min-Yen Kan. 2011. Logical structure recovery in scholarly articles with rich document feature. *Journal of Digital Library Systems. Forthcoming*.

Song Mao, Azriel Rosenfeld, and Tapas Kanungo. 2003. Document Structure Analysis Algorithms: A Literature Survey.

Debashish Niyogi and Sargur N. Srihari. 1995. Knowledge-based derivation of document logical structure. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 472–475.

Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, January.

Emanuele Pianta, Christian Girardi, and Roberto Zanoli. 2008. The TextPro tool suite. In *LREC, 6th edition of the Language Resources and Evaluation Conference*, Marrakech (Marocco).

Richard Power, Donia Scott, and Nadjet Bouayad-Agha. 2003. Document Structure. *Computational Linguistics*, 29(2):211–260, June.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using tranformation-based learning. In *Proceedings of Third ACL Workshop on Very Large Corpora*, pages 82–94.

Kristen M. Summers. 1998. *Automatic discovery of logical document structure*. Ph.D. thesis, Cornell University.

Ian H. Witten, David Bainbridge, and David M. Nichols. 2003. *How to build a digital library*. Morgan Kaufmann.