Logical Structure Recovery in Scholarly Articles with Rich Document Features

Minh-Thang Luong, Thuy Dung Nguyen and Min-Yen Kan[*]
National University of Singapore, Singapore

ABSTRACT

Scholarly digital libraries increasingly provide analytics to information within documents themselves. This includes information about the logical document structure of use to downstream components, such as search, navigation and summarization. We describe SectLabel, a module that further develops existing software to detect the logical structure of a document from existing PDF files, using the formalism of conditional random fields. While previous work has assumed access only to the raw text representation of the document, a key aspect of our work is to integrate the use of a richer representation of the document that includes features from optical character recognition (OCR), such as font size and text position. Our experiments reveal that using such rich features improves logical structure detection by a significant 9 F_1 points, over a suitable baseline, motivating the use of richer document representations in other digital library applications.

Keywords: ParsCit, Metadata Extraction, Logical Structure Discovery, Conditional Random Fields, Rich Document Features

INTRODUCTION

The pace of scholarly exploration, publication and dissemination grows faster every year, reaching unprecedented levels. To support this level of innovation, scholars increasingly rely on open-access mechanisms and digital libraries, portals and aggregators to disseminate their findings (Brown, 2009). While there is controversy over which of the trends of search engines, open access, preprint and self-archiving have most influenced the growth of scientific discovery, the consensus is that these batteries of methods have bettered the dissemination of scholarly materials. Now, an arguable bottleneck in the scientific process is in the processing, sensemaking and utilization of scholarly discoveries for the next iteration. Scholars are still largely confined to printing, reading and annotating the papers of their interest offline, without the help or guidance of a digital library to organize and collect their thoughts.

We believe a key component of a strategy to address this gap is in building applications that take advantage of the logical structure and semantic information within the documents themselves. Even within the limited domain of computer science, searching for competing methodologies to solve a problem, analyzing empirical results in tables, finding example figures to use in a presentation, or determining which datasets have been used to evaluate an approach,

[*] Corresponding author

are all comparative tasks that researchers do on a regular basis. Unfortunately, currently these can only be done manually, without aid from any computing infrastructure.

To support such analytics is not trivial and requires groundwork. One important subtask that is common to all of the above problems is to obtain the logical structure of the scholarly document. We paraphrase Mao, Rosenfeld, & Kanungo's (2003) earlier definition and define a document's *logical structure* as "a hierarchy of logical components, such as (for example) titles, authors, affiliations, abstracts, sections, etc." Note that the logical structure we seek is more comprehensive than what most other published systems identify. Namely, we identify not only metadata such as title, authors, abstract and parsing references, but also the logical structure of the internals of the document – sections, subsections, figures, tables, equations, footnotes and captions.

In this paper, we present SectLabel, an open source system to solve two related subtasks in logical structure discovery: 1) logical structure classification, and 2) generic section classification. In the first task, we consider a scholarly document as an ordered collection of text lines, and need to label each text line in a document with a semantic category, representing its logical role. In the second task, we take the headers of each section of text in a paper as evidence to deduce a generic logical purpose of the section.

We accomplish our implementation by extending an existing, freely available platform for reference string parsing, ParsCit[i]. ParsCit uses the machine learning methodology of conditional random fields (CRF), a model that blends sequential labeling techniques with pointwise entropy-based classification. We extend the use of CRFs in ParsCit to also provide logical structure discovery through the addition of the SectLabel module.

A further reality of document processing is that inputs come in many forms of markup: from richly annotated XML representations of OCR output to noisy, raw text dumps provided by copy and paste operations. Robustness is thus highly desirable, where the tool does not fail but where output quality may gracefully degrade as the input quality becomes poorer.

We summarize our contributions as follows:

1. We present the first logical structure discovery tool that expressly caters for scientific documents. Unlike previous scholarly document processing systems, it attempts logical structure discovery over the span of the entire paper at a fine-grained (per line) level.
2. We infer the generic logical purpose of each major section of text, mapping specific sections names to generic ones (*i.e.* "5. Text Features" → Methodology). This promotes comparative viewing of sections with identical purpose across articles.
3. Our implemented system handles both rich formatted input from an optical character recognition system, as well as from plain text dumps of scholarly articles. We evaluate both modes of input and conclusively show that rich input enhances classification performance, especially for certain logical structure classes.
4. We have created gold-standard training data for these tasks and made this dataset public at the original ParsCit site to encourage others to perform comparative evaluation.

We first discuss related work with an emphasis on work originating in the document analysis and digital library communities. Then, we formalize the two tasks and discuss the learning methodology of conditional random fields in more details. The section on System Architecture gives an overview of our system architecture; the subsequent discussion on Classification Categories explains different classification categories used. We discuss plain text features first, followed by a discussion of the rich input features that we distill in the case of OCR input. We quantify our system's performance in experiments, discuss our system's relative merit and discuss current and future work in final portions of the paper.

RELATED WORK

Logical structure analysis has a long research history that has been surveyed in multiple prior works (see Mao *et al.*, (2003) as an example). To keep the discussion relevant to our work, we limit our discussion briefly to two specific aspects: 1) the use of the conditional random field (CRF) specifically in digital library area, and 2) logical structure analysis where the main focus has been the use of OCR information.

The Use of Conditional Random Fields in Digital Libraries

We draw attention to a class of problems, called the *sequence labeling task*, which attempts to assign labels to a sequence of observations. Our tasks of logical structure analysis and generic section labeling can be viewed as instances of such problems. Such labeling problems are common and arise in many fields, including bio-informatics, computational linguistics, and speech recognition (Durbin *et al.*, 1998; Manning & Schütze, 1999; Rabiner & Juang, 1993). There have been many learning models designed to tackle the sequence labeling task, of which the Conditional Random Field (Lafferty, McCallum, & Pereira, 2001) is a recent and effective formalism.

In the context of digital libraries, CRFs have already been widely used in numerous applications, most notably metadata extraction tasks, being described as "the most difficult task performed by an automated digital library system for research papers" (Councill *et al.*, 2006). The choice of such a learning framework is justified in Peng and McCallum's (2004) work, which demonstrated the performance gains when substituting CRF for Hidden Markov Models (Seymore, McCallum and Rosenfeld, 1999; Takasu, 2003), or Support Vector Machines (Han *et al.*, 2003). Examples of systems using CRFs include CiteSeerX (Councill *et al.*, 2006) – a search engine for scientific literature, ArnetMiner (Tang *et al.*, 2008) – an academic social network search system, and ParsCit (Councill, Giles and Kan, 2008) – a reference string parsing software package, which is also incorporated in CiteSeerX.

Logical Structure Analysis

Of all the surveyed works on logical structural analysis, Kim, Le and Thomas' (2001) work is most related to ours in utilizing OCR information for analysis. They first categorize a paper into a layout template, and then apply OCR features such as the bounding box, font size/attributes together with word list tables to perform classification on biomedical journals. Similarly, Klink, Dengel and Kieninger (2000) use geometric information, text patterns and font information given by OCR output to classify business letters and journal papers. These works, however, rely on hand-crafted rules and heuristic matching.

At a more fined-grained level, works on mathematical expression recognition have been a research focus of the document processing community for many years (Suzuki *et al.*, 2003; Fujiyoshi, Suzuki and Uchida, 2009) using OCR primitive features. These works also utilize heuristically tuned systems whose rulesets need to be reconstructed for its application to a new problem area.

Statistical approaches for inferring logical document structure address the shortcomings of rule based approaches and have been a more recent focus of research. Belaïd and Rangoni (2008) have used neural networks to classify a similar set of categories. The LATISI project[ii] has made inroads towards building per journal / venue classifiers for logical blocks of text using a memory based classifier.

These works come the closest to our goal of a system for segmenting a document, but stop short of completing the classification into a set of fine-grained categories that cater for document metadata, logical structure and construct (*i.e.*, definitions, formulas and theorems) categories. Furthermore, many works up to now have focused on using only textual features or spatial and font (i.e., OCR-based) based evidence, but have not adequately demonstrated systems that use both in a synergistic way. A necessary development is a robust logical document structure inference system that can handle both rich input (page image information), but still be able to perform inference on impoverished input (plain text) with degraded performance. In the remainder of this paper, we detail the construction and evaluation of our SectLabel system, which aims to fill this important gap.

METHODOLOGY

Problem Formulation and Learning Model

Scholarly articles can be viewed as consisting of multiple lines, generally organized in several logical flow patterns such as a header with title, author and affiliation lines; followed by abstract, introduction lines; then body text; ending with a conclusion and lines making up its references. As such, our problem is naturally modeled as a formal sequence labeling task with the input text as a sequence of multiple lines $L = \{l_1, l_2, \ldots l_n\}$. Each element l_i needs to be assigned a correct label from a set of classes $C = \{c_1, c_2, \ldots c_m\}$. Evidence used to classify a line l_i not only comes from the features of that line itself, but also from the previous classifications of $l_1 \ldots l_{i-1}$. We make an assumption that each document line contains homogenous text belonging to only one category, *i.e.* a unique class, hence, making our problem an instance of the hard categorization task.

We employ the CRF learning model which is often expressed using the following simplified form:

$$p(y \mid x, \lambda) = \frac{1}{Z(x)} \exp(\sum_j \lambda_j f_j(y, x)) \qquad (1)$$

Equation 1 represents the probability of a particular sequence *y* given an observation sequence *x*. In this simplified form, $f_j(y, x)$ is used to represent either a state function $s(y_i, x, i)$ or a transition function $t(y_{i-1}, y_i, x, i)$. λ_j are the feature weights to be set by training and $Z(x)$ is the *partition function*, used to guarantee that the resulting $p(y|x,\lambda)$s are proper probabilities. State and transition functions are defined in terms of binary features as illustrated below using our problem context:

$$\mathrm{b(x,i)} = \begin{cases} 1 & \text{if } 1^{\text{st}} \text{ word of line } \mathrm{x_i} = \text{"}\textit{University}\text{"} \\ 0 & \text{otherwise} \end{cases}$$

Given *b*(*x*, *i*), the state and transition functions will then be of the form:

$$\mathrm{s(y_i, x, i)} = \begin{cases} \mathrm{b(x,i)} & \text{if label } \mathrm{y_i} = \text{affiliation} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{t(y_{i-1}, y_i, x, i)} = \begin{cases} \mathrm{b(x,i)} & \text{if } \mathrm{y_{i-1}} = \text{author} \,\&\, \mathrm{y_i} = \text{affiliation} \\ 0 & \text{otherwise} \end{cases}$$

In our implementation, we use the open-source CRF++ package[iii] to handle CRF specifics of inducing and applying the learned models. With this formalism, the focus of our problem becomes in feature engineering; that is, the design and selection of binary features *b*(*x*, *i*). Our input into CRF++ will be of the form "$value_1$ … $value_m$ $category_i$" for each line l_i ($i = \overline{1,n}$), where $category_i$ is the true class at training time and which is to be inferred at test time. $value_1$ … $value_m$ are feature values corresponding to a fixed set of *m* feature types. CRF++ automatically converts feature values into binary features *b*(*x*, *i*) by means of a feature template file, providing a clean and neat way to later incorporate our text and OCR features (see later section on "Raw Text Features and Rich Document Representation").

System Architecture

Our system consists of two parts: a primary component, *Logical Structure* (LS), and a subordinating part, namely *Generic Section* (GS). The LS component takes in full-text papers as input, while the GS component only deals with header lines. We detail the system pipeline through two stages – training and testing – and illustrate these processes in Figure 1.

At training time, we run our OCR software (discussed later) through all input papers to obtain raw text data as well as XML layout information. If only a stream of plain text is provided, only the raw text is used as input. The *LS feature extractor* utilizes both raw text and layout information, when available, while the *GS feature extractor* takes in only text headers from the raw text. Both the LS and GS extracted features, together with the manually labeled data, go through the CRF trainer to produce the corresponding learned models.

Unseen data during testing is represented as set of lines $\{l_1, l_2, \ldots l_n\}$ with additional XML layout information. The LS feature extraction processes these data, and then passes them to the LS classifier to label each line l_i with one of 23 structure categories c_i such as title, author, or header. Most labels $\{c_1, c_2, \ldots c_n\}$ at this step are final; only ones classified as headers need further processing. The counterpart GS modules further consider lines l_i where c_i=header to perform additional classification, relabeling l_i into one of 13 generic section headers c'_i such as introduction, abstract, or methodology. Those newly-labeled logical headers are then incorporated into the original output to form the final output.

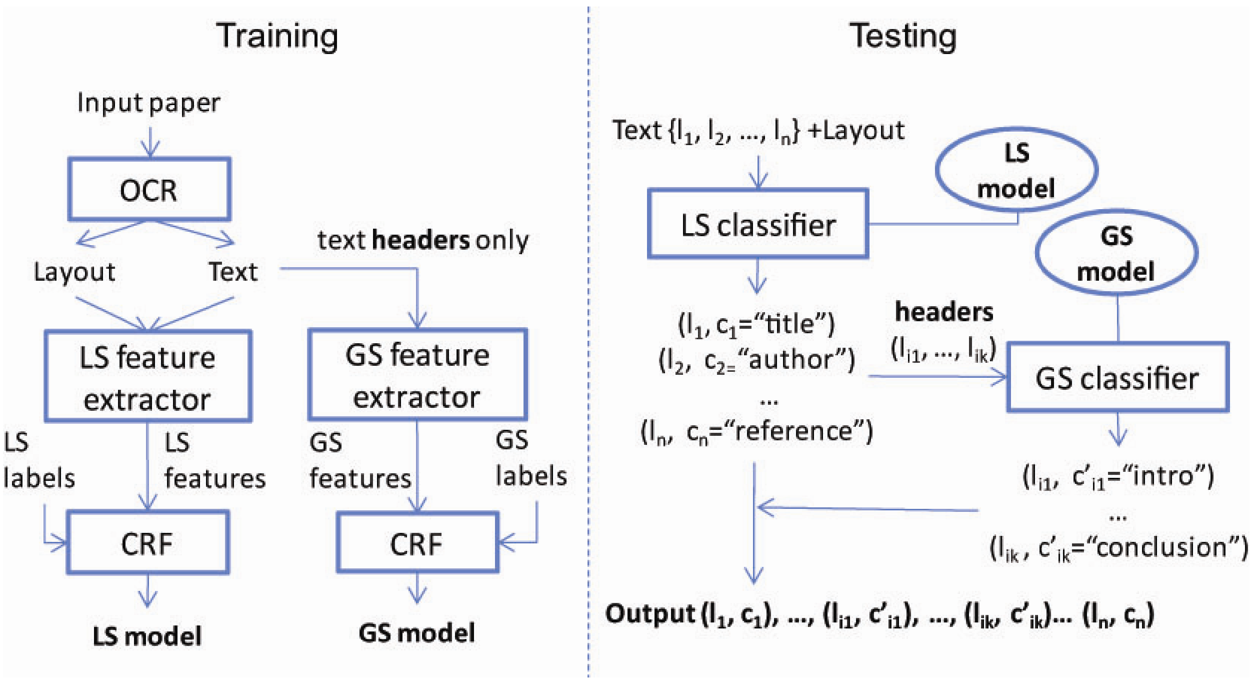
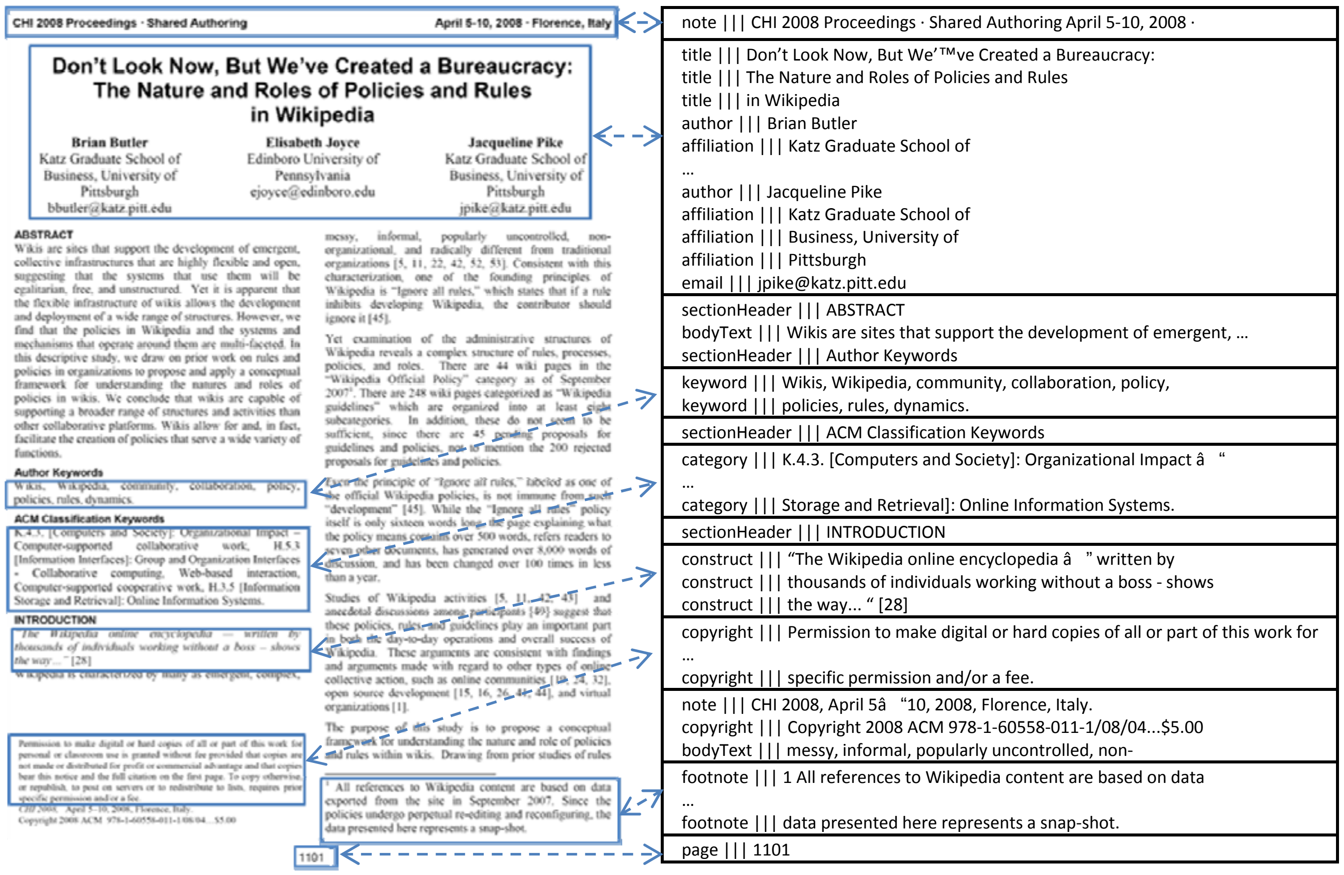


Figure 1: System architecture, showing training and testing phases. For clarity, LS classifier represents both the LS feature extractor and classifier; similarly for the GS classifier.

Classification Categories

In the logical structure LS subtask, we propose to assign each line of text to one of a set of 23 categories: address, affiliation, author, bodyText, categories, construct, copyright, email, equation, figure, figureCaption, footnote, keywords, listItem, note, page, reference, sectionHeader, subsectionHeader, subsubsectionHeader, table, tableCaption, and title. The names of most categories are largely self-descriptive, so we only discuss the exceptions. Note refers to additional text at the top or bottom of a page that is not a footnote or endnote. In the corpora that we have used to create SectLabel, such notes generally capture conference details. Categories and keywords denote paper metadata that describe the content. In the case of scholarly documents that follow the Association for Computing Machinery (ACM) style, we consider "general terms" as keywords. Lastly, we use construct broadly to define a block of texts that are separated from the main text visually. These include mathematical expressions such as definitions, lemmas, proofs, propositions, or corollaries; or as illustrated by our example in Figure 2, a quotation or saying which is distinct from the main text.

In the generic section subtask, we use a set of 13 categories to characterize a scholarly document’s sections: abstract, categories, general terms, keywords, introduction, background, related work, methodology, evaluation, discussion, conclusions, acknowledgments, and references. These categories are frequently used in many papers, and should be self-explanatory.



```
note ||| CHI 2008 Proceedings · Shared Authoring April 5-10, 2008 ·
title ||| Don’t Look Now, But We’™ve Created a Bureaucracy:
title ||| The Nature and Roles of Policies and Rules
title ||| in Wikipedia
author ||| Brian Butler
affiliation ||| Katz Graduate School of
...
author ||| Jacqueline Pike
affiliation ||| Katz Graduate School of
affiliation ||| Business, University of
affiliation ||| Pittsburgh
email ||| jpike@katz.pitt.edu
sectionHeader ||| ABSTRACT
bodyText ||| Wikis are sites that support the development of emergent, ...
sectionHeader ||| Author Keywords
keyword ||| Wikis, Wikipedia, community, collaboration, policy,
keyword ||| policies, rules, dynamics.
sectionHeader ||| ACM Classification Keywords
category ||| K.4.3. [Computers and Society]: Organizational Impact â “
...
category ||| Storage and Retrieval]: Online Information Systems.
sectionHeader ||| INTRODUCTION
construct ||| “The Wikipedia online encyclopedia â ” written by
construct ||| thousands of individuals working without a boss - shows
construct ||| the way... “ [28]
copyright ||| Permission to make digital or hard copies of all or part of this work for
...
copyright ||| specific permission and/or a fee.
note ||| CHI 2008, April 5â “10, 2008, Florence, Italy.
copyright ||| Copyright 2008 ACM 978-1-60558-011-1/08/04...$5.00
bodyText ||| messy, informal, popularly uncontrolled, non-
footnote ||| 1 All references to Wikipedia content are based on data
...
footnote ||| data presented here represents a snap-shot.
page ||| 1101
```

Figure 2: Examples of different LS categories. Each labeled line is in the format "label ||| line content".

RAW TEXT FEATURES

We now detail the features used in our model that are based solely on the raw text. These features form the minimal set of features used to classify lines of a document when no rich OCR features are provided, as in the case where the input is limited to plain text. Note that we use a different set of features for the logical structure and generic section classifiers.

Logical Structure Classifier

The text features of our LS classifier are best described by two distinct categories: *token-level* features that pertain to a single token, and *line-level* features that capture some aspect of a text line as a whole. For token-level features, we only extract features for the first *n* tokens in each line (set experimentally to *n*=4 in internal tests). As the original ParsCit uses a subset of the features used in our SectLabel module, we use the original ParsCit feature set as a baseline feature set for comparison.

A key difference that distinguishes the two tasks is that SectLabel classifies *entire lines* rather than individual tokens, as ParsCit does. As such, we add per-line features to capture information related to the text line as a whole[iv]. We describe the feature groups that we used in this component.

1. **Location** – Encodes the relative position of each line within a document (discretized into *n*=8 bins experimentally). This feature helps to separate different groups of labels: headers (*e.g.*, title, author, affiliation), body (*e.g.*, bodyText, figure, table, equation), and ending parts such as references.
2. **Number** – Detects the occurrence of patterns specific to hierarchies. These include subsectionHeaders ("1.1"), subsubSectionHeaders ("1.1.1"), ontologies (such as ACM categories, *e.g.*, "H.1.1"), footnotes (for both ordinary notes (1This work …" as well as URLs "1http…").
3. **Punctuation** – Checks if the line consists of email addresses or web links (potentially corresponding to references, footnotes or notes). This class of features includes values that classify lines that end with some type of bracket numbering, such as "(1)". Such values are good indicators of equations and other constructs.
4. **Length** – Measures the length of each text line in terms of tokens. Values for this feature are limited to 1token, 2token, 3token, 4token, 5+token. This feature is helpful to identify the majority of lines that should be considered as bodyText.

Generic Section Classifier

Academic publications tend to follow a consistent structure, *i.e.*, the majority in this group of publications contains the following generic sections: abstract, introduction, related work, methodology, evaluation, conclusions, acknowledgments and references. However, their actual section headers may not be the same; methodology is rarely labeled as such and may span multiple sections of a document. Hence, the section header classifier is built to automatically associate each section header with its corresponding generic header. We can thus deduce the logical purpose of each section in a scholarly document.

In implementing the section header classifier, we derive a small set of features from the header itself, such as its position as well as its content. Note that our implementation eschews features that could be derived outside of the headers themselves, such as the content of the corresponding sections – such a model would have larger memory requirements and is more computationally expensive to train and may not lead to increased accuracy. Thus, we have stuck to a simple model.

1. **Position –** Encodes the absolute and relative positions of headers within documents. *Absolute position* uses ordinal position of the header in a document, where as *relative position* normalizes ordinal position by the total number of sections in the whole document (discretized into *n*=10 bins, experimentally). Since scholarly documents share a consistent structure, abstracts or introductions usually come as the first two sections of the document, while methodology sections tend to appear in the middle, we believe both types of positions boost performance.

2. **First and Second Words** – Models the individual tokens of the header. We use the first and second words of each header as explicit features for our classifier, to better differentiate those that usually are expressed in a standard form. For example, abstract sections most likely have the header "Abstract", while related work usually manifests as "Related Work", "Previous Work" or "Literature Review".
3. **Whole Header** – Models the header as a whole, using concatenated header as a single feature. This acts as a memoization of all headers in the training data. As many headers are formulaic, this can be helpful.

RICH DOCUMENT REPRESENTATION

Raw text features alone produce acceptable performance for many of the document structure categories. But for others, it fails dramatically. For example, when section headers are not marked by initial numbering, such as in ACM Conference on Human Factors in Computing Systems (CHI) conference proceedings, there is little evidence in the raw text that indicates that a header is present. This is also true of other header-like categories – including title, author, and affiliation – which are often confused by the CRF models utilizing only raw text features. This is probably because they often occur at the beginning of a document with similar capitalization patterns and lengths.

However, to the human reader, these semantic categories are visually different, due to their positioning, font and size. We hypothesize that if we can provide the learner with such information, performance for these critical semantic categories would improve. In fact with PDF and other richly formatted text documents, we do have a much richer form of representation to text. Spatial layout, page breaks, and text font variations provide an orthogonal channel of information that can be used to derive logical structure.

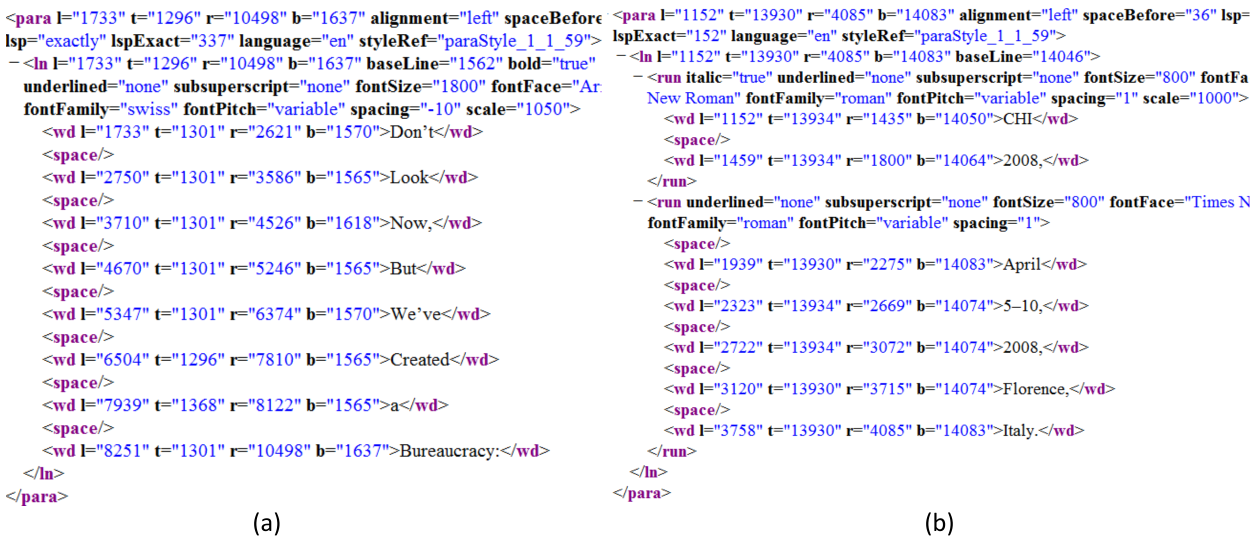
While digital formats such as Microsoft Word, HTML and PDF all convey this information, inferring structure directly on such document formats is difficult. If one takes this route, there are a multitude of different formats to interpret and separate modules must be created for each. Instead, we choose to go to the lowest common denominator and deal with documents as consecutive page images. This alternative route allows us to handle any document type that can be scanned or printed. We then run an optical character recognition (OCR) engine on the results to obtain rich format information.

In this section, we justify our choice of OCR software together with the extraction process of rich document features from the OCR output. We then categorize such rich information into *stationary* features (conditioned on a single label) and *differential* ones (conditioned on a pair of consecutive labels) with respect to the CRF framework.

OCR Processing

Today's OCR engines retrieve the raw text of documents quite reliably, if restricted to well-known languages and font families. OCR engines can also provide rich representations that recover the font, spacing and spatial layout of elements on the page and reconstruct the natural reading order of the elements. Nuance OmniPage (version 16) outputs such information in an accessible XML format, as seen in Figure 3. The information includes the coordinates of paragraphs (*para*), lines (*ln*) and words (*wd*) within a page (*l=left*, *t=top*, *r=right*, *b=bottom*), alignment, font size, font face, or format (*bold*, *italic*, *underlined*). While several commercial OCR packages are able to produce similar output, we chose OmniPage based on previous positive experience reported by other research groups.

The XML representation, while rich, needs to be linearized to become features for the CRF framework. We first extract the individual text lines from the XML output, and augment the line with its pertinent rich spatial and font features. During the feature extraction process, we note that text lines do not necessarily follow homogeneous formatting, *e.g.* when there are occurrences of special words highlighted. In such case, OmniPage often subdivides words in the same line (*ln*) into chunks (*run*). As in Figure 3b, the first run contains italic information for the phrase "Chi 2008," whereas the second run does not. When only a single word is different, per-word formatting attributes are directly embedded within the word (*wd*) tag. We handle these different levels of representations in OmniPage output to correctly identify dominant formatting features of each text line, *e.g.* whether the whole line is considered italic.



```
<para l="1733" t="1296" r="10498" b="1637" alignment="left" spaceBefore
lsp="exactly" lspExact="337" language="en" styleRef="paraStyle_1_1_59">
 - <ln l="1733" t="1296" r="10498" b="1637" baseLine="1562" bold="true"
   underlined="none" subsuperscript="none" fontSize="1800" fontFace="Ar
   fontFamily="swiss" fontPitch="variable" spacing="-10" scale="1050">
     <wd l="1733" t="1301" r="2621" b="1570">Don't</wd>
     <space/>
     <wd l="2750" t="1301" r="3586" b="1565">Look</wd>
     <space/>
     <wd l="3710" t="1301" r="4526" b="1618">Now,</wd>
     <space/>
     <wd l="4670" t="1301" r="5246" b="1565">But</wd>
     <space/>
     <wd l="5347" t="1301" r="6374" b="1570">We've</wd>
     <space/>
     <wd l="6504" t="1296" r="7810" b="1565">Created</wd>
     <space/>
     <wd l="7939" t="1368" r="8122" b="1565">a</wd>
     <space/>
     <wd l="8251" t="1301" r="10498" b="1637">Bureaucracy:</wd>
   </ln>
</para>
```

(a)

```
<para l="1152" t="13930" r="4085" b="14083" alignment="left" spaceBefore="36" lsp=
lspExact="152" language="en" styleRef="paraStyle_1_1_59">
 - <ln l="1152" t="13930" r="4085" b="14083" baseLine="14046">
   - <run italic="true" underlined="none" subsuperscript="none" fontSize="800" fontFa
     New Roman" fontFamily="roman" fontPitch="variable" spacing="1" scale="1000">
       <wd l="1152" t="13934" r="1435" b="14050">CHI</wd>
       <space/>
       <wd l="1459" t="13934" r="1800" b="14064">2008,</wd>
     </run>
   - <run underlined="none" subsuperscript="none" fontSize="800" fontFace="Times N
     fontFamily="roman" fontPitch="variable" spacing="1">
       <space/>
       <wd l="1939" t="13930" r="2275" b="14083">April</wd>
       <space/>
       <wd l="2323" t="13934" r="2669" b="14074">5–10,</wd>
       <space/>
       <wd l="2722" t="13934" r="3072" b="14074">2008,</wd>
       <space/>
       <wd l="3120" t="13930" r="3715" b="14074">Florence,</wd>
       <space/>
       <wd l="3758" t="13930" r="4085" b="14083">Italy.</wd>
     </run>
   </ln>
</para>
```

(b)

Figure 3: Example of the OCR XML output from OmniPage which forms the input to SectLabel. This XML excerpt encodes the OCR information for: (a) the first title line of the example paper's title page shown in Figure 2, and (b) the conference detail line within the copyright block of that same paper.

As a result of the linearization process, sample training vectors to the CRF model look like the following: (Figure 3a) "Don't-Look-Now,-But-We've-Created-a-Bureaucracy. Loc_0 Align_left FontSize_largest Bold_yes Italic_no Picture_no Table_no Bullet_no", and (Figure 3b) "CHI-2008,-April-5–10,-2008,-Florence,-Italy. Loc_7 Align_left FontSize_small Bold_no Italic_no Picture_no Table_no Bullet_no". The two feature vectors reflect the key differences between the title and the conference detail lines: the former is at the top of the page (*Loc_0*), employing the largest font size, and is bolded; whereas the latter is at the bottom of the page (*Loc_7*) in small font. These differences will assist the machine learner in inferring the correct labels, and such linear representation of rich document features provides a flexible way to either concatenate them to the raw text feature vector, or use them alone.

Stationary Features

We describe a set of rich OCR-derived features extracted directly from the OmniPage output. We term these features "stationary" as they are meant to reflect the state of the current line of interest, *e.g.* whether it is bolded or italicized, as opposed to differential features that capture state changes between two consecutive lines, discussed in the next section. These features are conditioned on a single output label, and realized under the CRF++ package by means of "unigram" features. We group these features as follows:

1. **Location** – Reflects the position of a text line within a page. We discretize the vertical coordinates reported by OmniPage into *n* bins (*n*=8 experimentally). We stress that this feature is the relative location within *each page*, and is distinct from the raw-text *Location* feature, which measures position with respect to the *entire document*. We expect to further differentiate labels such as note and footnote, which occur at the top and bottom of a page.

2. **Format** – Encodes salient font information, in the form of *FontSize*, *Bold*, and *Italic*. Knowing whether a text line is bolded and italicized distinguishes them from ordinary bodyText. Furthermore, they help identifying different levels of *headers*, especially for those paper styles using formatting instead of numerical hierarchies to denote headers.

   We pay special attention in transforming the literal *FontSize* information output in the XML. The actual font sizes are not as important compared with how frequent they occur, since font sizes vary among papers. It is the *relative font size* that we wish to model, thus requiring us to normalize font sizes. We achieve this first by finding the most frequent size in the document, and tagging this as the base. Larger font sizes than normal often imply certain categories such as title or header. We assign values starting from the largest as 0, then decreasing by 1 for each smaller size (*e.g.*, -1), until the base size is reached. Since smaller font sizes than the base are not indicative enough, and often due to OCR mistakes, we label them all as "smaller". This also helps avoid the problem of data sparseness. An example of our font labels are *smaller, smaller, base, -2, -1, 0,* which might be generated in the case the font sizes 6, 8, 10 (base font size), 12, 20 and 32 points all occur in a document. We later demonstrate and validate the effectiveness of our relative model.

3. **Object** – Captures special line attributes. We extract *Bullet*, *Picture* and *Table* attributes directly from the XML output. These features help to indicate if a text line is part of a paragraph or text block that has been specially identified in the XML output with one of these formatting attributes. While it may seem easy to categorize lines with the presence of these attributes, it is not always the case. Authors may employ such attributes for both logical and stylistic reasons. Tabular text can mix with figure or bullet points, or a figure (such as a flowchart) can contain bullet points that are recognized. Our experimental results demonstrate that using these features alone does not yield much improvement, but that they work better in conjunction with other features.

Differential Features

Though useful in contrasting lines of different labels, stationary features are not explicitly designed to capture text blocks that span multiple lines that use the same format. It is, however, crucial that these lines under the same text block, such as figure or captions, are labeled consistently. With current stationary features, there is no direct information for the machine learner to infer if two consecutive lines are of the same format. To explicitly model this, we also encode differential features that capture state changes between consecutive lines. These features are not extracted directly from OmniPage output, but rather synthesized from the values of stationary features in consecutive lines. In the CRF formalism, our differential features are conditioned on a pair of consecutive output labels and described in CRF++ as “bigram” features.

1. **Format** – We base formatting features on five distinct sources of information – *FontSize, Bold, Italic, FontFace* and *Alignment* – to explicitly mark if the current line has the same format as the previous. The feature will take the value “format_same” if all the five properties match and “format_new” for all other cases.

   Font face and alignment properties are extracted from OmniPage output similar to font size, bold, and italic ones discussed before. Font face values vary across documents, while alignment values take on one of five different values – *none*, *left*, *center*, *right* and *justified.* We experimented font face (normalized) and alignment as stationary features, but did not obtain any performance gains. However, as differential features, they yield a positive improvement, as detailed later in our experiments.

2. **Paragraph** – We process OmniPage output to identify blocks of text lines spanned under its XML output paragraph tag *para* (See Figure 3). The first line in each text block is assigned the feature value “para_new”, whereas the remainder take the value “para_same”. OmniPage sometimes groups multiple lines in the header section into a single paragraph, such as author, affiliation, and email. This is undesirable as we want to label the same for text lines within a paragraph, or a text block. Thus, we heuristically detect abstract or introduction header lines, and consider each line before this threshold of the document as a single paragraph receiving a value of “para_header”.

EVALUATION

The purpose of our evaluation is to answer the following questions:

1. How well does the logical structure classifier perform when using the baseline features from ParsCit?
2. How does performance change when adding the new raw text and rich document features?
3. How do the categories compare in classification difficulty and how does their classification accuracy vary with different feature sets?
4. With respect to the generic section classifier, does a shift from using a maximum entropy model as used in the previous work (Nguyen and Kan, 2007; discussed below) to a CRF model improve performance?
5. How do the individual features contribute to the overall performance of its feature group?

Let us first describe the datasets used to evaluate both the logical structure and generic section modules. We then detail on our evaluation metrics, and report overall performance for the LS and GS classifiers.

Logical structure (40 documents)				**Generic section** (211 documents)	
Category	#	Category	#	Category	#
address	64	note	148	abstract	210
affiliation	108	page	347	categories	165
author	66	reference	3,970	general terms	142
bodyText	25,062	sHeader	463	keywords	209
category	73	ssHeader	323	introduction	210
construct	234	sssHeader	78	background	28
copyright	186	table	1,098	related work	105
email	64	tableCap	228	methodology	608
equation	835	title	68	evaluation	151
figure	2,175			discussions	36
figureCap	472			conclusions	189
footnote	364			acknowledgements	102
keyword	68			references	211
listItem	1,308	**Total**	37,802	**Total**	2,366

Table 1: Logical structure and Generic section category instance counts. Note that sectionHeader*,* subsectionHeader *and* subsubsectionHeader *are abbreviated.*

Our *logical structure* dataset comprises of 40 scientific papers in the field of computer science. We try to cover a diverse collection of layouts and formats by including papers that originate from conferences that use different style guidelines for both the document body and references. The corpus includes 20 ACM papers spanning various years and venues, 10 papers from the 2009 Proceedings of the Association for Computational Linguistics Annual Meeting, and 10 papers from the 2008 proceedings of the ACM Conference on Human Factors in Computing Systems. The first author of this paper manually assigned categories to each line of these papers using the 23 logical section categories.

For the *generic section* dataset, we reuse the dataset of 211 ACM papers used in Nguyen and Kan (2007), which also worked on the same task of generic section detection. They used a different feature set and experimented with a maximum entropy learning framework, which is a pointwise learning model, different from the sequence labeling model embodied by CRF. We used this set so that we can compare results directly with this previous work. We have manually extracted headers of these papers, and assigned the same 13 generic section categories.

Both datasets have been made available on the original ParsCit website to spur future research on automatic logical structure analysis. Demographics on the corpus and counts of different categories are presented in Table 1.

Evaluation Metrics

Let TP denote the number of correctly classified text lines (true positive); similarly, FN for false negatives, FP for false positives, and TN for true negatives. We report both overall and category-specific results.

For category-specific performance, we use F_1 *measure* as our metric of interest, defined as $\frac{2xPxR}{P+R}$ where P is Precision = $\frac{TP}{TP+FP}$, and R is Recall = $\frac{TP}{TP+FN}$. Due to the skewness of the dataset (as bodyText lines are the majority of lines), we do not report line accuracy[v], as it would misleadingly imply very good performance; a simple baseline that does no work and reports all lines as bodyText would trivially get 66% accuracy.

For assessing overall performance, we measure both macro and micro F_1 to provide a comprehensive evaluation. Macro F_1 weighs each category equally, computed as an average of all category-specific F_1. Micro F_1 weighs each instance (line) equally[vi].

Results

We perform 10-fold stratified cross validations for both the logical structure (LS) and generic section (GS) classifiers. We first report the final performance of different systems using full set of features, and leave the analysis as well as evaluation of different feature types to the next section.

LS Evaluation

To answer the first two questions, we trained and tested a CRF for logical structure classification using different feature sets. Let us define three feature combinations: LS_{PC} – the baseline using only features found in the original ParsCit distribution, LS_{PC+RT} – which adds the raw text features, and $LS_{PC+RT+RD}$ – that further adds rich document features derived from the OCR output. Table 2 compares performance among these three systems for each category, and also reports overall micro- and macro-averaged performance.

Table 2 shows that the baseline LS_{PC} reproducing the original ParsCit (PC) feature set performs fairly well, garnering 68 F_1. However, it is clear that by considering additional raw text (RT) features tailored for logical structure classification and further incorporating rich document (RD) features has a significant impact: LS_{PC+RT} and $LS_{PC+RT+RD}$ monotonically increase macro F_1 to 76 and 85 points, respectively. Micro averaged results show smaller improvements, indicating that improvements were largely to minority classes.

This is confirmed when reviewing the category-specific performance to answer our third question. $LS_{PC+RT+RD}$ and LS_{PC+RT} demonstrate improvement over LS_{PC} with differences of greater than 10 F_1 points for many categories. With minor performance degradation in copyright category, $LS_{PC+RT+RD}$ greatly improves the majority of categories, out of which 13 categories have improvements greater than 5 F_1 points. Furthermore, $LS_{PC+RT+RD}$ shows especially improved performance with rich document features for the categories footnote and subsubsectionHeader, yielding a substantial improvement of 28.09 and 40.37 F_1 points, respectively, over the LS_{PC+RT} model.

	LS_{PC}	***LS_{PC+RT}***	***$LS_{PC+RT+RD}$***
Macro F_1	68.37	75.64	84.72
Micro F_1	90.01	91.03	93.38
address	66.67	80.00	**85.48**$^{+5.48}$
affiliation	76.76	90.57	**92.82**
author	71.93	90.91	**97.74**$^{+6.83}$
bodyText	95.37	95.82	**96.97**
category	66.67	82.96	**85.71**
construct	7.86	13.82	**33.11**$^{+19.29}$
copyright	94.79	**95.37**	95.11
email	80.34	96.12	**97.64**
equation	56.76	58.76	**72.01**$^{13.25}$
figure	72.71	76.87	**79.93**
figureCaption	63.05	62.21	**76.91**$^{+14.7}$
footnote	31.58	41.49	**69.58**$^{+28.09}$
keyword	58.82	62.61	**74.02**$^{+11.41}$
listItem	57.47	62.33	**71.21**$^{+8.88}$
note	95.10	95.53	**96.22**
page	91.41	95.39	**97.84**
reference	99.26	99.50	99.50
sectionHeader	88.27	90.22	**93.51**
subsectionHeader	70.31	75.97	**91.39**$^{+15.42}$
subsubsectionHeader	19.64	41.32	**81.69**$^{+40.37}$
table	73.21	72.26	**79.59**$^{+7,33}$
tableCaption	63.89	65.56	**80.69**$^{15.13}$
title	70.69	94.03	**100**$^{+5.97}$

Table 2: Comparative performances among LS_{PC}, LS_{PC+RT} and $LS_{PC+RT+RD}$ CRF models for logical structure classification. Category-specific performance given in F_1. Results in bold mark the best system for each category. Superscripts indicate large improvements in F_1 (> 5 points) between first and second ranked systems.

GS Evaluation

For the generic section classifier, Table 3 compares the performance between GS_{maxent}, a maximum entropy based system reported in (Nguyen and Kan, 2007), and our CRF-based system, GS_{crf}.

These results answer Question 4. Overall, our generic section classifier using CRF model achieves macro F_1 of 90.87 and micro F_1 of 95.82. This outperforms those of the maximum entropy classifier from previous work with scores of 87.71 and 93.28, respectively. At the per-category level, with the exception of background, GS_{crf} betters GS_{maxent} in all categories, some of which contain large improvements such as introduction, methodology, evaluation, discussions, conclusions, and acknowledgements.

	GS_{maxent}	GS_{crf}
Macro F_1	87.71	90.87
Micro F_1	93.28	95.82
Abstract	99.53	**100**
categories	100	100
general terms	99.65	**100**
Keywords	99.52	**99.76**
introduction	97.87	**99.29**$^{+1.42}$
background	**60.00**$^{+3.59}$	56.41
methodology	90.74	**93.76**$^{+3.02}$
evaluation	78.00	**83.21**$^{+5.21}$
relate work	93.33	**93.40**
discussions	38.46	**59.65**$^{+21.19}$
conclusions	87.96	**96.34**$^{+8.38}$
acknowledgements	96.23	**99.51**$^{+3.28}$
references	99.53	**100**

Table 3: Comparative performance between a maximum entropy based system (figures reproduced from (Nguyen and Kan, 2007)) and our CRF-based system for the task of generic section classification. Results in bold mark the better system for each category; superscripts indicate large improvement in F_1 (> 1 point).

FURTHER ANALYSIS

To answer our final question, we return to logical structure classification. We provide further detailed assessment of $LS_{PC+RT+RD}$ by evaluating the effectiveness of individual raw text and rich document features. We then we categorize several types of errors made by $LS_{PC+RT+RD}$ in our discussion and error analysis before concluding with a brief discussion of GS_{crf}.

Impact of Raw Text Features

We carried out an ablation test to understand the contribution of each of the 4 line-level text features used in LS_{PC+RT}. Results in Table 4 indicate that removing any of the features degrades performance, implying that all of the individual features contribute to the final composite performance. The most influential text feature is *position*; without it, overall performance drastically worsens.

Feature	Macro F_1	Micro F_1
Full	**75.64**	**91.03**
- length	74.73	90.57
- punct	75.85	90.72
- num	75.53	90.97
- pos	71.28	90.64

Table 4: Ablation test results for LS_{PC+RT}.

Impact of Rich Document Features

We are interested in finding out how each of the rich document feature groups influences performance. We add each of the *stationary* feature groups – position, format and object – into LS_{PC+RT} separately to assess their performance impact. We then incrementally add format, object, and lastly, the *differential* feature groups to obtain the final $LS_{PC+RT+RD}$ model.

System	Macro F_1	Micro F_1
LS_{text}	**75.64**	**91.03**
+ position	77.80	91.48
+ format	78.20	90.73
+ object	77.02	91.88
+ position, format	78.59	91.19
+ position, format, object	81.71	92.44
+ position, format, object, differential (RD)	**84.72**	**93.38**

Table 5: Performance with rich document (RD) features. The bottom model corresponds to $LS_{PC+RT+RD}$.

Individually, all stationary feature groups contribute positively to performance. Specifically, format (font) features contribute the most to macro average, while object features influence micro average most. Closer inspection of category-specific performance reveals that format features contribute to performance gains for a wider spectrum of categories, many of which are related to paper metadata and section headers; whereas object features contribute to just a few. However, those categories improved by object features already contain a large number of training data, which explains why format yields a larger macro F_1 improvement compared to micro F_1. Combining position and format features show consistent improvements in both metrics. Using all the three group of stationary features yields an absolute performance of 81.71 macro F_1 and 92.44 micro F_1.

Further adding differential features achieves a significant improvement of 3.01 macro F_1 and 0.94 micro F_1. Detailed inspection shows that 21 out of 23 are enhanced, 7 of which have performance gains greater than 3 F_1 points. As expected, major improvements are accounted by categories which often occur in text blocks: construct (+13.52 F_1), figureCaption (+13.49 F_1), and tableCaption (+12.64 F_1).

Stationary feature analysis

We also performed a subtractive analysis by dropping one feature at a time from the full stationary feature set, in order to evaluate their roles individually. Table 6 indicates that all stationary features, except *Picture*, contribute towards the final performance in both macro and micro averages. Removing picture feature degrades macro F_1, but slightly increases micro accuracy. Our close inspection reveals that while discarding picture feature affects many categories; it does, in contrast with other features, enhance the bodyText classification by 0.02 F_1 point. Since bodyText possesses a large number of samples, such a small increase in bodyText macro F_1 accounts primarily for the slight increase in the overall micro F_1 accuracy.

According to the ablation results, *location*, *bold* and *table* are the most effective features in each feature group. In particular, the location feature demonstrates their special usefulness for footnote as removing it severely degrades classification performance of footnote by 22.01 F_1. The bold feature, when being left out, affects mainly tableCaption (-3.16 F_1), subsectionHeader (-6.28 F1), and subsubsectionHeader (-6.5 F_1). Lastly, discarding table feature reduces performance of table category by 6.01 F_1, while removing bullet feature mostly influences listItem (-9.74 F_1).

Feature	Macro F_1	Micro F_1
LS_{text} + location, format, object	**81.71**	**92.44**
- Location	79.52	91.63
- Bold	80.04	91.70
- Italic	81.08	92.12
- Font	80.65	92.13
- Bullet	80.77	91.91
- Picture	81.28	92.51
-Table	80.73	91.81

Table 6: Ablation test for stationary features.

Differential feature analysis

We evaluate the effectiveness of differential features, *format* and *paragraph*, by means of the same incremental method. From the model that includes text and OmniPage stationary features (LS_{text} + location, format, object), we add differential features, format and paragraph, separately. This setting allows us to experiment with different subsets of information sources used to construct the format differential features. These sources include: font size (S), font face (F), bold (B), italic (I), and alignment (A). Once the best setting for format differential feature is known, we test the final model consisting of both format and paragraph differential features.

Feature	Macro F_1	Micro F_1
LS_{text} + location, format, object	**81.71**	**92.44**
+ format_S	82.30	92.30
+ format_SF	82.89	92.80
+ format_SFBI	82.56	92.80
+ format_SFBIA	83.29	92.95
+ paragraph	83.87	93.06
+ format_SFBIA, paragraph (RD)	**84.72**	**93.38**

Table 7: Performance with differential features. The bottom model corresponds to $LS_{PC+RT+RD}$.

Results in Table 7 suggest that the *format* differential feature performs best when utilizing all information sources. Incrementally, adding sources S, F, and A enhances the performance; whereas adding B and I over S and F does not demonstrate any gain. However, our experiment with SFA alone results in inferior performance as compared to the full composite SFBIA, suggesting that B, and I are essential sources to obtain the best combination with performance of 83.29 macro F_1 and 92.95 micro F_1.

The paragraph differential feature alone demonstrates even better performance at 83.87 macro F_1, and 93.06 micro F_1. When considering both types of differential features, we consistently improve performance in both macro and micro averages, obtaining our best system at 84.72 F_1 and 93.38 micro F_1.

Rich Document Feature Distribution

We have utilized so far a large pool of rich document features to greatly improve the performance. While differential features are from our own synthesis, stationary features are extracted directly from the OCR output. A natural set of questions to ask is: To what extent are we taking advantage of the OCR results? How noisy are the OCR results? Does the use of *bullet*, *picture* and *table* features make it trivial to recognize categories like listItem, figure, and table? We answer these by providing the statistics in Table 8, which compares OmniPage output with the manual annotations to tally the number of lines having a particular feature value and being annotated with a specific category.

	Alignment					**Bold**	**Italic**	**Bullet**	**Picture**	**Table**
	none	justified	left	right	center					
address	18	0	1	0	45	0	0	0	0	0
affiliation	18	0	7	0	83	0	0	0	0	0
author	11	0	11	0	44	18	0	0	0	0
bodyText	18853	5464	575	83	96	30	**78**	**227**	**99**	4
category	34	11	28	0	0	1	2	0	0	0
construct	116	62	42	1	6	11	**59**	0	0	0
copyright	40	125	22	0	1	0	1	0	0	0
email	8	0	10	0	46	1	1	0	0	0
equation	45	81	485	113	111	1	**53**	4	**349**	**34**
figure	557	361	1095	57	105	2	**41**	16	**1588**	**177**
figureCap	132	133	94	4	109	**118**	0	0	31	0
footnote	190	47	112	13	2	0	0	3	2	0
keyword	38	0	30	0	0	0	0	0	0	0
listItem	455	522	321	2	8	11	17	**929**	11	0
note	10	4	116	0	18	0	10	0	**93**	0
page	5	0	332	0	10	0	0	0	**336**	0
reference	2978	481	474	5	28	0	**368**	**2354**	5	0
sHeader	65	214	170	1	13	**347**	1	21	3	1
ssHeader	86	178	56	0	3	**255**	3	0	1	0
sssHeader	20	47	9	0	2	7	**59**	0	0	0
table	62	9	1015	2	10	15	5	0	**96**	**912**
tableCap	105	44	44	8	27	**72**	0	0	0	1
title	2	0	18	3	45	**62**	0	0	0	0
Total	**23848**	**7783**	**5067**	**292**	**812**	**951**	**698**	**3554**	**2614**	**1129**

Table 8: Distribution of rich document features over LS classes. Column headers list the rich document feature values extracted from OmniPage: alignment has five feature values; whereas bold, italic, bullet, picture, table columns represent for "yes" feature values as these are binary features. Each count in the

table is the number of lines in the LS training data having a particular feature value. Highlights in the last five columns show non-trivial values (as there are many cells having 0 counts).

Statistics in Table 8 show that the performance of OmniPage is reasonable. It detects, for example, that address, affiliation, author, email, and title have the tendency to take on a "center" alignment; similarly, figureCaption, sectionHeader, subsectionHeader, tableCaption and title are the major categories that are bolded. However, there is still confusion in these features as evidenced by the distribution of the same feature value in multiple categories. This, we believe, is where there is a need for a machine learner to come, to select good representative values for each subset of categories.

We note that the performance of OmniPage for object features is satisfactory as labels for listItem, figure, and table with respective accuracies 39.46, 60.75, and 80.78[vii]. These accuracies reflect baseline accuracy if one was to just use OmniPage output to label these classes. According to the confusion matrix in Table 9, SectLabel's corresponding accuracies are 77.98, 81.11 and 77.99[viii]. While comparable for the table category, the performance of listItem and figure are improved significantly.

Error Analysis

We feel the performance of the full composite $LS_{PC+RT+RD}$ and the GS_{crf} models are substantial, and significantly better than the state-of-the-art for both tasks in logical structure discovery. However, there is still room for improvement, and a careful review of the models' errors allows us to characterize problematic areas for improvement.

Logical structure classification

From the confusion matrix in Table 9, we have aggregated the number of false-positive (FP) labels for each category. These figures indicate that the groups of labels accounting for most errors include bodyText, equation, figure, listItem and table. We analyze the causes as:

1. A skew towards the majority class of bodyText, which influences learning. This is a common problem in text classification, and future models may want to specifically account for skewed categories. The 1098 misclassified bodyText lines are nearly half of all errors. The confusion matrix shows that a large amount construct, equation, figure, figureCaption, footnote, listItem, table and tableCaption are incorrectly categorized as bodyText.
2. The conflict between textual and spatial features. Tables, in several cases, are considered figures and vice versa. These mixtures challenge the classifier, as evident by the fact that 213/2175 = 9.79% figures are misclassified as tables, and 126/1098 = 11.48% vice versa. Figures and tables often contain equations and list items, making an independent, per-line judgment different than what we might suspect in the context of a page. We observe a case in which a table compared different formulas and was labeled as a figure.
3. Errors contributed by OCR recognition and our category annotation standard. In several cases, OCR recognizes texts within figures, many instances of which contain numbers and notations that resemble equations. Such figures include charts with numbers in axes or titles, or equations embedded as figures. Due to our convention of labeling these lines as figure, there is a significant confusion between figure and equation, evidenced by the

fact that 123/835 = 14.73% equations are mislabeled as figures, and 146/2175 = 6.71% vice versa. Finally, the construct category still poses a great challenge when our current best performance only achieves a poor F_1 of 33.11%. The confusion matrix reveals that a large portion of these lines were labeled as bodyText. This is partly due to the difficulty in labeling whole block of lines as construct. Definitions, for example, are hard to distinguish, with the exception of the initial line which may contain lexical cues such as "Definition 1".

	ad	af	body	cat	con	cop	eqn	fig	fCap	fn	kw	list	ref	sH	ssH	sssH	tab	tCap
address	53	4	0	0	0	0	0	3	0	0	0	0	0	1	0	0	2	0
affiliation	7	97	0	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0
bodyText	0	0	24636	0	13	1	52	84	50	12	0	197	11	2	5	0	3	5
category	0	0	0	60	0	0	0	0	7	0	4	0	0	2	0	0	0	0
construct	0	0	**143**	0	50	0	11	17	0	0	0	3	0	0	0	1	0	2
copyright	0	0	3	3	0	175	0	0	0	4	0	0	0	0	0	0	0	0
equation	0	0	53	0	3	0	611	**123**	0	0	0	13	0	0	1	1	30	0
figure	0	0	57	0	3	0	**146**	1718	8	3	0	21	0	0	1	0	**213**	0
figureCap	0	0	126	0	0	0	0	8	338	0	0	0	0	0	0	0	0	0
footnote	0	0	145	0	0	0	0	3	0	207	0	6	3	0	0	0	0	0
keyword	0	0	15	4	0	0	0	0	0	0	47	0	0	2	0	0	0	0
listItem	0	0	409	0	0	0	11	29	1	0	0	857	0	0	0	0	0	0
ref.	0	0	12	0	0	0	0	3	0	4	0	0	3945	0	0	0	0	0
sH	0	0	17	0	1	0	0	1	0	0	8	0	1	425	9	1	0	0
ssH	0	0	17	0	0	0	0	0	0	0	0	2	0	9	292	3	0	0
sssH	0	0	9	0	2	0	0	1	0	1	0	0	0	0	7	58	0	0
table	0	0	32	0	3	0	30	**126**	3	0	0	0	4	0	1	0	893	6
tableCap	0	0	60	0	0	0	0	1	0	0	0	0	0	0	0	0	4	163
FP	7	4	**1098**	7	25	1	**250**	**400**	69	24	12	**242**	19	19	24	6	**252**	13

Table 9: Full composite logical structure classifier $LS_{PC+RT+RD}$ *model's confusion matrix. Bold figures discussed in the text. For compactness and as there is little confusion among* author, email, note, page, *and* title, *these classes are omitted from the table. FP is the per-category false-positive, computed by summing up all values in a column minus the diagonal cell in that column.*

Generic section classification

Table 3 indicates that most errors fall into background and discussions categories. Instances of these categories were often mislabeled as methodology or conclusions. 61% of background instances were mistaken as methodology, 25% of discussions headers were labeled as methodology and 20% of them were labeled as conclusions.

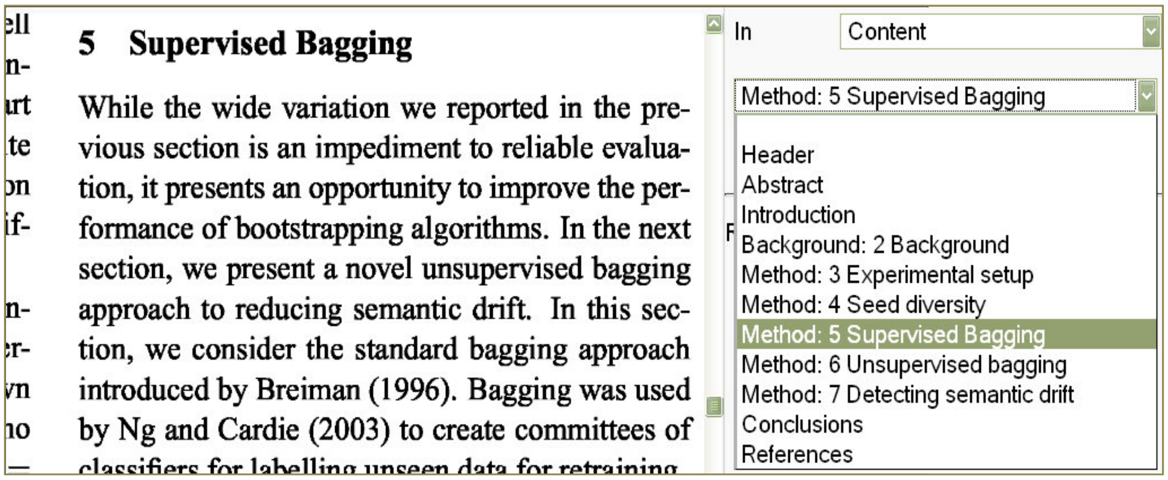
The feature set for GS_{crf} is currently based only on location and content of the headers, and underperforms at differentiating consecutive headers with different labels. As such, the errors are likely caused by three factors: First, the actual headers in these categories may not have any tokens in common with the memoized training data instances; second, the relative positions of consecutive headers (background vs. methodology, methodology vs. discussions, and discussions vs. conclusions) are quite similar to each other; we believe dataset skew also contributes to the difficulty here. There are a prominent number of methodology instances in our dataset, especially when we compare with the background and discussions categories. This also explains why many headers are mislabeled as methodology. We believe changing the CRF to encode constraints beyond the simple linear chain dependency may be helpful.

DEPLOYING SECTLABEL

Even though SectLabel is intended as groundwork for downstream applications, we believe that logical structure is useful to a reader in its own right. Logical structure, when exposed to the reader, can help a reader better understand the structure and argumentation of the document, and serves as a navigation aid. To demonstrate these capabilities, we have integrated our work in SectLabel with a document reading environment, ForeCiteReader.

Our integration offers the reader two methods of interacting with the automatically-recovered logical structure, corresponding to our two tasks. We note that these interfaces are preliminary and need to undergo more extensive design trials before we believe they will be effective for readers.

1. **Section Navigation** (from generic section detection): Figure 4 shows a list of document's headers, together with their corresponding generic headers. Through this list, users can navigate to different sections in a document. The generic headers allow readers to jump to certain key sections of a paper such as methodologies or empirical results. This interface works like a table of contents sidebar present in other digital reading environments, such as Adobe Acrobat.



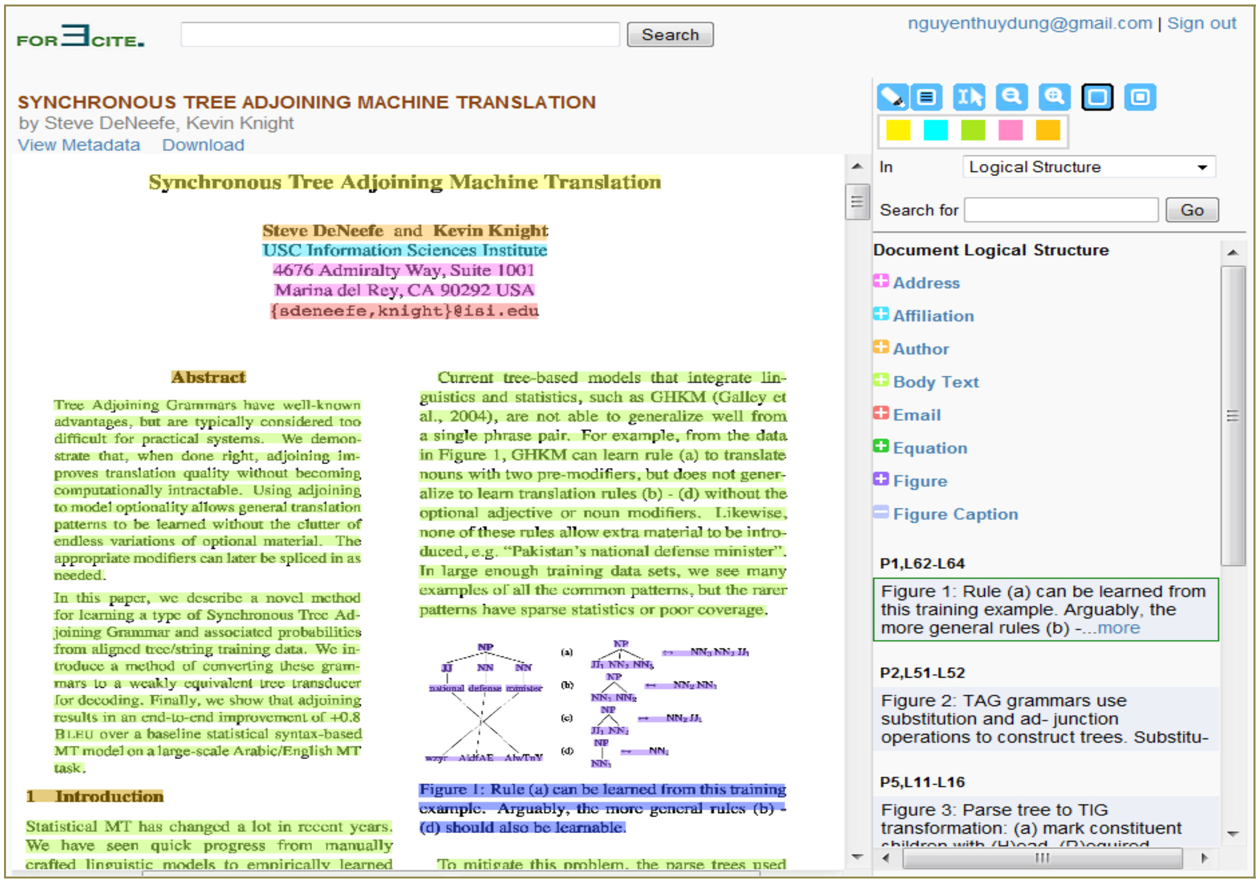
5 Supervised Bagging

While the wide variation we reported in the previous section is an impediment to reliable evaluation, it presents an opportunity to improve the performance of bootstrapping algorithms. In the next section, we present a novel unsupervised bagging approach to reducing semantic drift. In this section, we consider the standard bagging approach introduced by Breiman (1996). Bagging was used by Ng and Cardie (2003) to create committees of classifiers for labelling unseen data for retraining

In
Content
Method: 5 Supervised Bagging
Header
Abstract
Introduction
Background: 2 Background
Method: 3 Experimental setup
Method: 4 Seed diversity
Method: 5 Supervised Bagging
Method: 6 Unsupervised bagging
Method: 7 Detecting semantic drift
Conclusions
References

Figure 4: Section navigation in the ForeCiteReader reading environment.

2. **Object Navigation** (from logical structure detection): Figure 5 shows a screenshot of the reading interface in production. The right panel presents a collapsible interface with all objects in the document listed and grouped by types. Readers can thus jump to view specific objects such as tables, figures or equations.



SYNCHRONOUS TREE ADJOINING MACHINE TRANSLATION
by Steve DeNeefe, Kevin Knight

Synchronous Tree Adjoining Machine Translation

Steve DeNeefe and Kevin Knight
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292 USA
{sdeneefe,knight}@isi.edu

Abstract

Tree Adjoining Grammars have well-known advantages, but are typically considered too difficult for practical systems. We demonstrate that, when done right, adjoining improves translation quality without becoming computationally intractable. Using adjoining to model optionality allows general translation patterns to be learned without the clutter of endless variations of optional material. The appropriate modifiers can later be spliced in as needed.

In this paper, we describe a novel method for learning a type of Synchronous Tree Adjoining Grammar and associated probabilities from aligned tree/string training data. We introduce a method of converting these grammars to a weakly equivalent tree transducer for decoding. Finally, we show that adjoining results in an end-to-end improvement of +0.8 BLEU over a baseline statistical syntax-based MT model on a large-scale Arabic/English MT task.

1 Introduction

Statistical MT has changed a lot in recent years. We have seen quick progress from manually crafted linguistic models to empirically learned

Current tree-based models that integrate linguistics and statistics, such as GHKM (Galley et al., 2004), are not able to generalize well from a single phrase pair. For example, from the data in Figure 1, GHKM can learn rule (a) to translate nouns with two pre-modifiers, but does not generalize to learn translation rules (b) - (d) without the optional adjective or noun modifiers. Likewise, none of these rules allow extra material to be introduced, e.g. "Pakistan's national defense minister". In large enough training data sets, we see many examples of all the common patterns, but the rarer patterns have sparse statistics or poor coverage.

Figure 1: Rule (a) can be learned from this training example. Arguably, the more general rules (b) - (d) should also be learnable.

To mitigate this problem, the parse trees used

Document Logical Structure
Address
Affiliation
Author
Body Text
Email
Equation
Figure
Figure Caption

P1,L62-L64
Figure 1: Rule (a) can be learned from this training example. Arguably, the more general rules (b) -...more

P2,L51-L52
Figure 2: TAG grammars use substitution and ad- junction operations to construct trees. Substitu-

P5,L11-L16
Figure 3: Parse tree to TIG transformation: (a) mark constituent

Figure 5: Logical structure annotation in ForeCiteReader. The view shows object navigation interface, currently focusing on the list of figure captions.

CONCLUSION

We have described and evaluated SectLabel, an open-source freely available module for logical document structure classification. Logical structure, consisting of 23 categories, is determined on a per-line basis and section headers are further classified into one of 13 generic section types. SectLabel uses the conditional random field (CRF) framework to view both tasks as sequence labeling problems using binary feature functions.

We have explored and comprehensively evaluated the utility of different classes of features. We found that acceptable performance (~76 macro F_1) results from careful feature engineering on raw text. A key finding is that modeling additional per-page spatial information, yields a significant improvement of over 9 macro F_1 points, and significantly boosts detection of important categories such as paper metadata, captions, and hierarchical headers.

Our error analysis suggests areas for future work. For logical structure classification, further modeling text blocks will allow us to improve detection on construct and explicitly handling skewed categories like bodyText reduces confusion in the learning model. For generic section classification, selective analysis of the content of the sections may lead to further classification performance.

REFERENCES

Belaïd, A., & Rangoni, Y. (2008). Structure Extraction in Printed Documents Using Neural Approaches. *Machine Learning in Document Analysis and Recognition*, (pp. 21-43).

Councill, I. G., Giles, C. L., Iorio, E. D., Gori, M., Maggini, M., & Pucci, A. (2006). Towards Next Generation CiteSeer: A Flexible Architecture for Digital Library Deployment. *ECDL*, (pp. 111-122).

Councill, I., Giles, C. L., & Kan, M.-Y. (2008). ParsCit: an Open-source CRF Reference String Parsing Package. *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08).* Marrakech, Morocco: European Language Resources Association (ELRA).

Durbin, R., Eddy, S. R., Krogh, A., & Mitchison, G. J. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press.

Fujiyoshi, A., Suzuki, M., & Uchida, S. (2009). Syntactic Detection and Correction of Misrecognitions in Mathematical OCR. *ICDAR*, (pp. 1360-1364).

Han, H., Giles, C. L., Manavoglu, E., Zha, H., Zhang, Z., & Fox, E. A. (2003). Automatic document metadata extraction using support vector machines. *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries* (pp. 37 - 48). Washington, DC, USA: IEEE Computer Society.

Kim, J., Le, D. X., & Thomas, G. R. (2001). Automated labeling in document images. *Proceedings of SPIE Conference on Document Recognition and Retrieval VIII*, (pp. 111 - 122). San Jose, CA.

Klink, S., Dengel, A., & Kieninger, T. (2000). Document Structure Analysis Based on Layout and Textual Features. *Proceedings of the 4th IAPR International Workshop on Document Analysis Systems (DAS 2000)*, (pp. 99-111). Rio de Janeiro, Brazil.

Lafferty, J. D., McCallum, A., & Pereira, F. C. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 282 - 289). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing.* Cambridge, MA, USA: MIT Press.

Mao, S., Rosenfeld, A., & Kanungo, T. (2003). Document structure analysis algorithms: a literature survey. *Proc. SPIE Electronic Imaging*, (pp. 197-207).

Nakagawa, K., Nomura, A., & Suzuki, M. (2004). Extraction of Logical Structure from Articles in Mathematics. *International Conference on Mathematical Knowledge Management 2004*, (pp. 276-289).

Nguyen, T. D., & Kan, M.-Y. (2007). Keyphrase Extraction in Scientific Publications. *International Conference on Asian Digital Libraries 2007*, (pp. 317-326).

Peng, F., & McCallum, A. (2004). Accurate information extraction from research papers using conditional random fields. Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting 2004, (pp. 329 - 336).

Rabiner, L., & Juang, B.-H. (1993). *Fundamentals of speech recognition.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Rangoni, Y., & Belaïd, A. (2006). Document Logical Structure Analysis Based on Perceptive Cycles. *Document Analysis Systems*, (pp. 117-128).

Seymore, K., McCallum, A., & Rosenfeld, R. (1999). Learning Hidden Markov Model Structure for Information Extraction. *In AAAI 99 Workshop on Machine Learning for Information Extraction*, (pp. 37 - 42).

Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., & Kanahori, T. (2003). INFTY: an integrated OCR system for mathematical documents. *ACM Symposium on Document Engineering*, (pp. 95-104).

Takasu, A. (2003). Bibliographic attribute extraction from erroneous references based on a statistical model. *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries* (pp. 49 - 60). Washington, DC, USA: IEEE Computer Society.

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008). ArnetMiner: extraction and mining of academic social networks. *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 990 - 998). New York, NY, USA: ACM.

---

[i] http://wing.comp.nus.edu.sg/parsCit/

[ii] https://wiki.birncommunity.org:8443/display/NEWBIRNCC/LATISI+-+Literature+Annotation+Tool+from+the+Information+Sciences+Institute

[iii] http://crfpp.sourceforge.net/

[iv] We experimented with an orthography feature, which captures the overall capitalization pattern of a line, but it negatively affected the performance when combined with other features.

[v] (TP + TN)/(TP + FN + FP + TN)

[vi] For conciseness, throughout the remaining text, we will prefer to “macro F_1 average” as “macro F_1”, and “micro F_1 accuracy” as simply “micro F_1”.

[vii] (929/2354, 1588/2614, 912/1129) = (39.46%, 60.75%, and 80.78%)

[viii] (857/ (857 + 242), 1718/(1718 + 400), 893/(893 + 252)) = (77.98%, 81.11%, and 77.99%)