INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Metadata Extraction from Scholarly Articles

Ricardo Pereira Candeias

Dissertation for the achievement of the degree:

Master in Information Systems and Computer Engineering

Committee

Chairman:	Prof. Alberto Silva
Main supervisor:	Prof. Pavel Calado
Co supervisor:	Prof. Bruno Martins
Observers:	Prof. José Borbinha

November 2011

Abstract

Modern digital libraries of scholarly materials depend on the availability of according metadata. Thus, developing automated methods for metadata extraction from scholarly articles is an important problem that has been getting increasing attention. My MSc thesis addresses this problem through the usage of machine learning methods based on probabilistic models for sequence tagging. This dissertation presents the most important related concepts, introduces relevant previous research, and describes real world systems that have been built for the task. The dissertation also presents a prototype system for metadata extraction that uses multiple Conditional Random Fields models in a stacking-based architecture. Experiments with the prototype validate the research hypothesis, which stated that stacking Conditional Random Fields models is indeed an effective approach for addressing the problem of metadata extraction.

Keywords: Digital libraries , Conditional Random Fields , Metadata Extraction , PDF , Scholary Articles , Feature Engineering , Model Stacking

Resumo

Actualmente, as bibliotecas digitais que contêm material académico dependem bastante da disponibilidade dos seus metadados. Como tal, o desenvolvimento de mecanismos automáticos para a extracção de metadados de artigos com um cariz académico tem sido área de elevada importância. Neste âmbito, o trabalho apresentado neste documento pretende abordar este problema utilizando métodos de aprendizagem automática. Nesta dissertação são apresentados os conceitos mais importantes, trabalhos prévios relevantes e são descritos os sistemas que foram criados para a resolução desta tarefa. Neste documento são descritas experiências com um sistema para extracção de metadados que recorre aos modelos Conditional Random Fields com uma arquitectura baseada em composição. As experiências realizadas comprovam que a composição deste tipo de modelos pode ser uma abordagem valiosa para o problema de extracção de metadados a partir de artigos com um cariz acádémico.

Palavras- Chave: Bibliotecas Digitais, Conditional Random Fields, Extracção de Metadados, PDF, Artigos Científicos, Engenharia de Features, Composição de Modelos

Acknowledgements

(in Portuguese)

Tenho muito que agradecer a várias pessoas pela ajuda e disponibilidade ao longo do tempo em que elaborei esta dissertação. Assim, começo por agradecer ao meu orientador, Professor Pavel Calado e ao meu co-orientador Bruno Martins que ao longo deste ano sempre me apoiaram e tiveram disponibilidade para para me ajudar nesta fase final do curso.

Um agradecimento especial é devido aos meus colegas e amigos com os quais trabalhei ao longo dos anos.

Gostaria ainda de agradecer o suporte financeiro dado pelo projecto EuDML do programa CIP da comissão europeia.

Por último, gostaria de estender os meus agradecimentos a todos aqueles de uma forma ou de outra (por exemplo, fornecendo ideias e/ou criticas construtivas), foram ajudando anonimamente nas inúmeras discussões ao longo deste trabalho.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

After several years of massive digitization activities, current digital libraries hold large collections of scholarly materials. Moreover, most of the scholarly materials nowadays distributed by publishers are digital-born PDF documents. Still, most of these PDFs were designed for human consumption, lacking the descriptive metadata that is essential in supporting the advanced search and retrieval functionalities that are expected of modern digital libraries. Thus, an ongoing challenge within the Digital Libraries research community relates to the automatic extraction of metadata information from source documents, such as PDF scholarly articles (Giles *et al.* (1998)). This task raises important research challenges, since programatically recovering metadata fields from scholarly articles requires a machine to understand the structure of the strings in the different parts of the documents.

The contents of each PDF article contain the values for a set of metadata fields (e.g., author, title, year, journal, etc.) represented as string chunks, with implicit clues such as text position, punctuation and font size that can be used to assist in recovering the encoded information. While infering the metadata fields from the documents is often straightforward for human readers, the sheer diversity of document templates makes this process difficult to automate. Figure 1 provides an illustration of the metadata fields that are tipically found in a PDF article.

The task of metadata extraction consists of having an algorithm extracting the fields of interest automatically. Several solutions have been proposed, including rule-based parsers, based on regular expressions and machine learning methods. These are nowdays the state of the art algorithms for this problem ((Day *et al.*, 2007; Ding *et al.*, 1999; Han *et al.*, 2003; Peng & McCallum, 2004,)). In general, machine learning algorithms are robust and adaptable, as they can theoretically be used in any document set. The only setback is that generating the labeled training data is time consuming and costly. Nevertheless, even though ruled-based systems do not require any training, and are straightforward for implement, their dependence on the application domain and the need for an expert to set the rules, causes these

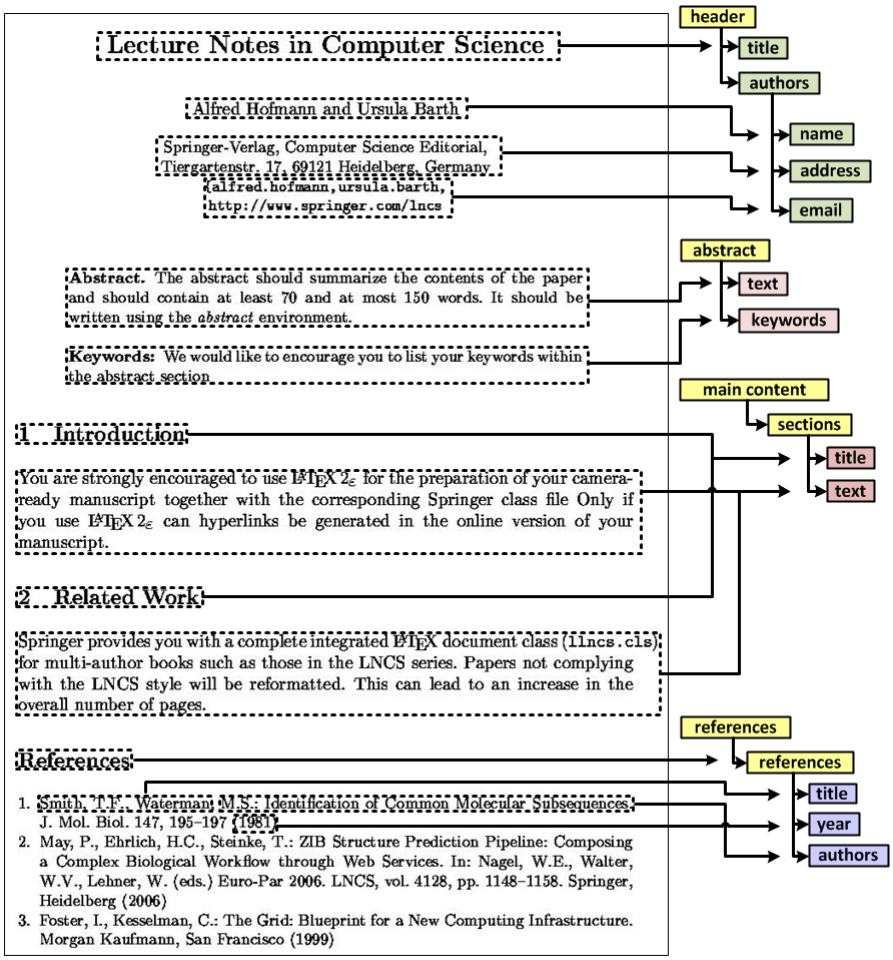

Figure 1.1: Metadata fields given a part of an article.

methods to have limited use. Therefore machine learning algorithms are usually prefered.

As previous researchs tried to address this problem of metadata extraction, these approaches still lack in a way to extract metadata from the full text of an article, and to consider certain properties found on the body of an article. Namelly most of the researchs done, focuses on the bibliographic extraction of metadata fields (e.g., author name, book title, etc.) and ignores the remanining text found in the articles that can be useful, like keywords, article title, and the authors of that article. Another aspect that most of the previous research does not consider is font information (e.g., font size, typeface), which can be useful in order to distinguish the different metadata fields found in the full text of an article. The work done in this dissertation aims to present a method for metadata extraction based on Conditional Random Fields, that focuses on the entire text of an article, and tries to devise a rich set of features based on its properties.

1.1 Hipothesis and Methodology

In the context of my MSc Dissertation, I propose to address the problem of recognizing individual metadata fields in scholarly articles by transcoding what is essentially a chunking problem (i.e., the task of discovering the chunks of text that correspond to specific metadata fields) into a tagging problem (i.e., a task of assigning tags to each individual token in the text of the articles, according their belonging to a given metadata field or not), using the standard begin/in/out (BIO) encoding of chunkings as tags. For this, I propose the use of a generative model commonly referred to as Conditional Random Fields (CRF) that offers an efficient and principled approach for addressing this sequence tagging problem ((Lafferty *et al.*, 2001; Peng & McCallum, 2004; Sarawagi & Cohen, 2004,)). A particular novelty introduced in my proposal is the usage of different CRF models for specific logical segments of the scholarly papers, instead of a single model for the entire contents of a document, similiar to what is done in Lu *et al.* (2007).

Specifically, I evaluated the use of stacked CRF models, for the task of metadata extraction in the various sections of the scholary articles (e.g., header, abstract, references, etc.), instead of just addressing bibliographic references extraction. These models use rich sets of features to label tokens as belonging to the different metadata fields.

1.2 Contributions

The proposed framework was evaluated through the usage of papers retrieved from arXiv[1] from the domain of Mathematics, using the method of cross validation, due to the reduced number of articles available. The results were obtained by the means of tag evaluation, and chunk evaluation. From the evaluation, the results attested for the following contributions:

- The development of a framework based on stacking of Conditional Random Fields, to use in the task of metadata extraction from scholarly articles, allowed this architecture to instead of using a single CRF model considering a larger set of classes corresponding to the entire set of metadata elements, to use simpler models which reduced the system difficulty associated with model learning, and data sparseness, thus requiring less training data. Also this allowed the training of the separate CRF models to be made through datasets of different sizes, and from different origins, which was the case seen in the model that identifies the metadata fields present in each bibliographic reference that uses the Cora dataset, and the remain models use the dataset build from arXiv.

[1] http://arxiv.org/

- The creation of an rich set of features, useful to the task, that includes in the set of features the font information resultant from the extraction of the text from a PDF document, that allows to capture properties useful in the identification of certain metadata fields, like the article title or section title.
- Construction of a collection of articless fully annotated for the considered metadata fields, from the domain of mathematics, based on arXiv, an e-print service, allowing future experiments in this domain of metadata extraction from scholarly articles.
- The results presented attest for the validation of the system with an overall F_1 score of 91%, that proves the usefulness of the method in obtaining the descriptive metadata required by digital libraries.

1.3 Document Organization

The rest of this document is organized as follows: Chapter 2 presents the general concepts involved in understanding the models themselves, and the previous work concerning metadata extraction. Chapter 3 presents the proposed extraction method. Chapter 4 presents the evaluation methodology proposed, and the results obtained during the validation of the method. Finally, Chapter 5 summarize the most important aspects of this document, and directions for future work.

Chapter 2

Concepts and Related Work

This chapter introduces the general concepts involved in the task of metadata extraction, such as machine learning models, that are commonly used to address this problem, and the concepts in which these models base themselves on. Also this chapter present the most relevant previous research done in the subject of metadata extraction.

2.1 Important Concepts

This section presents the most important concepts for the comprehension of this work. It explains how an unstructured text is treated in a Information Extraction (IE) problem (e.g., through the usage of token-level models), and shows how to evaluate if the words contain a certain property. It also introduces two frameworks for solving the information extraction problem through the labeling sequences of words, namelly Hidden Markov Models and Conditional Random Fields (Scheffer *et al.* (2002); Wallach (2004)).

Token-level models can be viewed as a statistical methodology for addressing information extraction in plain text, in which the unstructured text is treated as a sequence of tokens. In this context, a token is a word in the text of a document.

The information extraction problem consists in labeling each token according to it belonging to a given metadata field. Since metadata fields are normally composed of multiple tokens, it is typical to decompose each metadata field in “Begin”, “Inside” and “Out” tokens. This is commonly known as the BIO encoding (Sang & Veenstra (1999)). Another popular encoding for this problem decomposes the metadata fields in “Begin”, “Continue”, “End” and “Other” tokens, being refered to as the BCEO encoding (Sarawagi (2008)).

Two examples of token-level models that are nowadays commonly used in IE are Hidden Markov Models

(HMM) and Conditional Random Fields (CRF).

In the context of token-level models, token features consist of a set of clues for capturing various properties of each token and the context in which it lies. They can be thought of as a functions $f(x, i)$, which take as argument the sequence of tokens x together with the position i, returning a real-value that captures the properties of the ith token and of tokens that are in its neighborhood (Sarawagi (2008)). Common families of token properties used in IE are as follows:

- Word Features - The word itself can be used as a clue for assigning labels. An example of this feature is the comparison of the token we are evaluating to a certain word:

$$f_1(x, i) = [x_i \; equals \; "Lafferty"]$$

the function above would return true or false in case of the condition being verified, e.g., it would return true for x_3 in $\overset{Write}{x_1} \; \overset{by}{x_2} \; \overset{Lafferty}{x_3}$.

- Orthographic Features – The orthographic properties of the word (e.g., capitalization patterns), and the presence of certain symbols (e.g., punctuation symbols) can be used as clues for assigning labels. An example of this type of features can be a feature function that fires when a certain token starts with a capitalized letter followed by other non-capitalized letters (e.g., John).

- Dictionary Lookup Features – These are features that use a database of names (i.e., a dictionary) in order to match the given word.

2.1.1 Hidden Markov Models

Hidden Markov Models (HMM) provide a probabilistic framework for the problem of labeling and segmenting sequential data (Rakesh Dugab (1996)). HMMs consist of finite state machines with stochastic states and observation sequences. An HMM can also be viewed as a finite set of states $y_1, \ldots, y_c$ in which the state that the HMM is at a time t is given by y_t, and at each sequential position t the HMM generates an observation x_t. In this context, observations x_t are the individual words of a document, and the HMM state to which a word has been assigned represents the label of the token (Scheffer *et al.* (2002)). Figure 2.1.1 provides an illustration of the evolution of an HMM model.

In order to compute the probability of being at state y_t when observing token x_t at time t, a series of parameters are involved. An HMM λ is thus a set of parameters $\lambda = (A, B, \pi)$ where the array π_i (the probability of being in state i at the beginning of the experiment) denotes the probability $P(y_1 = y_i)$ of being at state i at the beginning of the experiment, A represents the probability of a transition from a state y_i to y_j and B quantifies the probability of an observation occuring in the state y_i we are at.

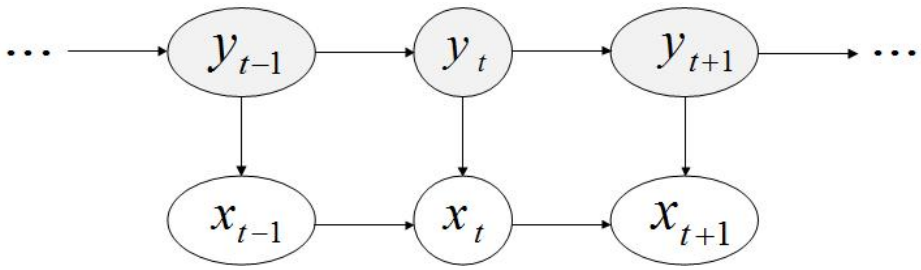

Figure 2.2: Sequential evolution of a Hidden Markov Model model.

Given the specification above, when applying the HMM, we can assume that a text is represented as a sequence $x_1, ..., x_c$ that corresponds to each word of the text. For each word x_c, we need to identify the state y_c, that represents the label which we want to assign. In HMM models, this is done through the following equation:

$$P(y|x, \lambda) = b_{i_1}(x_1)b_{i_2}(x_2) ... b_{i_T}(x_T) \tag{2.1}$$

After the states y_t are identified, it is necessary to extract those tokens which correspond to one of the target states $y_1, .., y_c$ and label them with the corresponding labels (Scheffer *et al.* (2002)).

The problem faced when applying the HMM is the choice of a state sequence I so that $P(x|I, \lambda)$ i.e., the joint probability of the observation sequence x and the state sequence I are maximized. To find the I that maximizes $P(x|I, \lambda)$, one can use the Viterbi Algorithm, which is a form of dynamic programming applicable to sequence tagging (Seymore *et al.* (1999)).

As for training of the HMM i.e., finding the parameters A, B and π one can use the Baulm-Welch algorithm or the Segmental K-Means algorithm (Rakesh Dugab (1996)). For the case of Segmental K-means the goal of the method is to adjust the parameters $\lambda = (A, B, \pi)$ in order to maximize the probability $P(I|x, \lambda)$, where I represents the optimal sequence of states for observation sequence x. As for the Baulm-Welch, the goal is to also adjust the parameters $\lambda = (A, B, \pi)$ to increase $P(x|\lambda)$ until a maximum value is reached. This method differs from Segmental K-Means as the focus in not given to a particular state sequence.

2.1.2 Conditional Random Fields

Conditional Random Fields (CRF) are another probabilistic framework for labeling and segmenting sequential data, drawing together the advantages of finite state HMMs with the use of arbitrary dependent features and a joint inference over entire sequences (Wallach (2004)). A CRF can be viewed as an undirected graphical model in which each vertex represents a random variable and each edge represents a

dependency between two random variables. For an input sequence of size c a CRF defines a conditional probability $P(y|x)$ for a sequence of labels $y = y_1 \dots y_c$ and the given input sequence $x = x_1 \dots x_c$ that as the following form:

$$p_\Lambda(y|x) = \frac{1}{\sum_y \prod_{c=1}^{C} \phi_c(y_c, y_{c-1}, x, c; \Lambda)} \prod_{c=1}^{C} \phi_c(y_c, y_{c-1}, x, c; \Lambda) \tag{2.2}$$

In the equation, ϕ are potential functions parametrized by Λ. Assuming ϕ_c factorizes a log-linear combination of features computed over the subsequence c, then:

$$\phi_c(y_c, y_{c-1}, x, c; \Lambda) = exp(\sum_k \lambda_k f_k(y_c, y_{c-1}, x, c))$$

In the formula, f is a set of arbitrary feature functions over the input, each having an associate model parameter λ_k. The feature functions can informally be thought of as measurements on the input sequence that partially determine the likelihood of each possible value for y_c. The parameter k represents the number of considered features. The parameters $\Lambda = \{\lambda_k\}$ are a set of real-valued weights, estimated from labeled training data by maximizing the data likelihood function through stochastic gradient descent.

Given a CRF model, finding the most probable label sequence, for an input x, can be made efficiently through the Viterbi algorithm in the same way as refered for the case of HMM models (Sarawagi (2008)). As for the training of a CRF model, the goal consists in given a training dataset $\left\{\left(x^{(k)}, y^{(k)}\right)\right\}$, estimating the parameters λ_k so that they maximize the probability of a set of label sequences. This can be done using likelihood-based training. Assuming training data that are independent and identical distributed, the product of equation (2.2) over all training sequences, as a function of the parameters λ_k, is known as the likelihood. Maximum likelihood training chooses parameter values such that the log-likelihood is maximized. So, for a CRF, the log-likelihood is given by the equation:

$$L(\lambda) = \sum_k \left[\log \frac{1}{Z(x^{(k)})} + \sum_i \lambda_i F_i(y^{(k)}, x^{(k)})\right] \tag{2.3}$$

Typical applications are based on first-order chain CRFs, which make the first-order Markov assumption on the dependencies among y (i.e., the transitions between tags y depend only on the origin and destination). For more details about the formalism of CRF models, the reader should refer to the original paper by (Lafferty *et al.* (2001)).

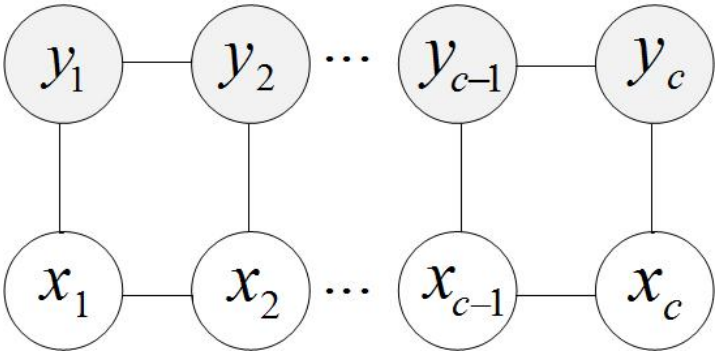

Figure 2.3: Conditional Random Fields Graph.

2.2 Related Work

Metadata extraction from scholarly articles has been described as *the most difficult task performed by an automated digital library system for research papers* (Councill *et al.* (2008a)). The task remains an important challenge for the Digital Libraries community, although several authors have proposed either machine learning approaches to address sequence labeling (Han *et al.* (2003)), or knowledge based approaches problem with basis on hand-tuned rules (Ivanyukovich & Marchese (2006); Lawrence *et al.* (1999)). The most common solutions are based on sequence labeling. There have been many models designed to tackle the sequence labeling task. Examples include Hidden Markov Models and Conditional Random Fields (CRFs), which are currently two of the most effective formalisms (Lafferty *et al.* (2001)).

This section presents previously proposed sequence labeling models and systems that have used them for metadata extraction. This includes systems like CiteSeerX (Councill *et al.* (2008a)), a search engine for scientific litterature, ParsCit (Councill *et al.* (2008b)), essentialy a reference string parsing software package which is also incorporated in CiteSeerX, and ArnetMiner (Tang *et al.* (2008)), an academic social network and search system.

2.2.1 Information Extraction from Scholary articles

This section presents the most well known systems for metadata extraction from scholary articles, which make use of machine learning models like Conditional Random Fields.

2.2.1.1 The CiteSeerX system

CiteSeerX is one of the first seach engines for scientific literature. It incorporates Autonomous Citation Indexing (ACI), a system that indexes academic literature in electronic format, such as Postscript and PDF files. CiteSeerX relies on information extraction approaches in order to build citation indexes (Giles

et al. (1998)). More precisely CiteSeer, has been built in order to provide a large flexibility, permiting the use of different approaches for metadata extraction.

In order to provide this flexible solution, CiteSeer implements a blackboard architecture (Councill *et al.* (2008a)), which consists of three main components, namelly (i) Knowledge Sources, (ii) a BlackBoard, and (iii) a Control component. The Knowledge Sources component is composed of *experts,* which correspond to modules that specialize in some part of the problem. These experts are the different approaches for solving the problem of information extraction. As for the BlackBoard, it corresponds to a global database which contains the input data, the partial solutions, and other information that the experts produce to support problem solving. Finally, the Control Component corresponds to the workflow controller that makes runtime decisions about the course of the problem solving, which more precisely consists of scheduling experts responsible for scheduling the knowledge sources that are chosen by the Control Component, depending on the extraction problem. For example, different scheduling experts can be used in order to dynamically change the model (e.g., CRF to HMM or vice-versa) and the accuracy value of the information produced by the models can be computed as the joint confidence among the experts.

One example of the use of this configuration in CiteSeer is grouping experts in three levels. The first level would be the recognition level, in which the experts give words a specific semantic augmentation (e.g., by tagging words) like recognition of named entities as first name, city, organization, etc. A second level would be row labeling in which the experts would label the tokens with labels like author, reference, section title, etc. Finally a third level would be the metadata construction level, where using all the extracted information from the previous labels the metadata record can be built.

Using this approach, CiteSeer can be integrated with other frameworks that specialize in the problem of metadata extraction. One case that takes high relevance is the integration of the open-source implementation of a reference string parsing package named ParsCit (Councill *et al.* (2008b)), that uses state of the art machine learning models and achieves a high accuracy. This work is addressed in more detail in the following section.

2.2.1.2 The ParsCit System

The work done in ParsCit focuses on two different tasks, namelly (i) extraction and parsing of reference strings, i.e the text strings in the bibliography or reference section of a published work that refer to a unique previously published document, and (ii) logical structure parsing of scientific documents (i.e., an extension to ParsCit known as SectLabel) (Councill *et al.* (2008b); Luong *et al.* (forthcoming)), i.e. identifying the hierarchy of logical components, such as titles, authors, affiliations, abstracts and

sections (Luong *et al.* (forthcoming)). Both tasks are architectured as supervised machine learning procedures that use Conditional Random Fields as their learning mechanism. ParsCit uses Conditional Random Fields to label the token sequences in the reference string, and it is coupled with heuristic processing to identify reference strings from a unstructured text file, and to retrieve the citation contexts.

For the task of extraction of reference strings, ParsCit given a reference string R, which is broken into a sequence of tokens $x_1 ... x_n$. The goal is to assign each token with the correct label from a set of classes $C = c_1, ..., c_{13}$. The classes considered by ParsCit for this task are, author, booktitle, date, editor, institution, journal, location, note, pages, publisher, tech, title, and volume. The classes correspond to common fields used in bibliographic references. Specifically the features used by this solution are as follows:

- **Token identity**: Different features are created for each token in three different forms, lowercased and lowercased stripped of punctuation.
- **N-gram prefix/suffix**: Four features are used for the first 1-4 characters of the token, and the same for the last 1-4 character and a feature that examines the last character of the token in order to encode whether it is uppercase, lowercase or numeric.
- **Orthographic case**: The token is analysed and then the value the feature function takes is either *Initialcaps*, *MixedCaps, ALLCAPS* or others.
- **Punctuation**: After analysis of the token, this feature takes the value multipleHyphens, continuingPunctuation, stopPunctuation, etc.
- **Number**: The token is analysed for numeric properties like for example, the feature takes the value *year* (if numeric value between 19xx and 20xx), *possiblePageRange* (for the case of [0-9]-[0-9]), etc.
- **Dictionary**: The token is verified in a dictionary of publisher names, places names, etc.
- **Location**: Indicates the relative position of the token within the reference string, like the position of important information such as author, title which is located in the beginning of the reference normally, in other words this feature tries to capture regularities in position.
- **Possible editor**: Indicates whether a token like “eds.” is present in the reference string. Also in order to avoid misclassifications of editors for authors the editor feature is modeled so that long-range dependencies are excluded. Also for important features the feature function is applied to a window of -2 to +2, which means that two tokens before and after the token that is being evaluated are considered.

ParsCit starts by finding the references using a set of heuristics that consist of searching by a labeled reference (e.g., “References”, “Bibliography”, etc.) section in the text. To do this the text is iteratively split in strings that are likely to be reference section labels. The final match is then considered the starting point of the reference section and the processing to find the ending starts by searching for subsequent section labels like appendices, figures, etc., or the end of the document. After the reference section is extracted, ParsCit segments it in individual strings. For the segmentation it uses regular expressions to identify the different types of reference start markers, and then by counting the number of hits to each regular expression. For those that matched more than 1/6 of the total lines in the reference section, the one with the greater number is choosen. Three cases for the start of a reference were considered, namelly (i) start with square braket or parenthetical markers (e.g., [1], (1)), (ii) start with naked number (e.g., 1), (iii) reference having no mark (e.g., such as APA style[1]). In case no reference markers are found, it applies a set heuristics in order to obtain the individual references, such as where individual strings start and end based on the lenght of previous lines, where strings that seem to be author name lists appear, and ending punctuation because the final line of a citation normally ends with a period. After obtaining the list of individual references the CRF model is applied.

After the previous phase, and using the CRF output, each tagged field is normalized, as author names may occur in different formats. The name string needs to be segmented into individual names, based on an analysis of separator locations (e.g, semicolon placement). Then, after the segmentation, each field is normalized for a predefined format.

For the task of logical structure parsing, ParsCit models the problem through a sequence of multiple lines $L = l_1, ..., l_n$, where each element l_i needs to be assigned with a label from a set of classes. It also makes the assumption that each line contains text belonging to only one class.

This module of the system is divided in two components, each with a CRF model. These are (i) Logical Structure (LS), which takes the full text of the paper as input, and (ii) Generic Section (GS), which focus on the header lines. These two differ in the aspect that while LS uses the raw text and layout information, and GS only uses the text headers from the raw text. Figure 2.4 shows the architecture of the system. The system starts by receiving a set of lines originated from the input paper, that is passed to the LS classifier (CRF). Then, the output headers are passed to the GS classifier (CRF), that relabels the headers. Afterwards the headers from the output of the GS classifier are incorporated in the output generated by the LS classifier, originating the final output.

For the LS subtask, the classes considered are, address, affiliation, author, bodyText, categories (i.e., like keywords denote papers metadata), construct (i.e., block of texts isolated from the main text, like mathematical expressions), copyright, email, equation, figure, figureCaption, footnote, keywords, listItem, note (i.e., text at the top or bottom of a page that is not a footnote or endnote, e.g., conference

[1] http://www.apastyle.org/

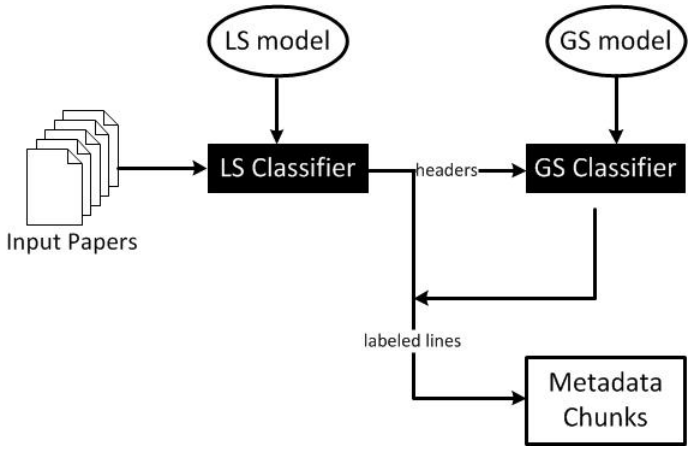

Figure 2.4: ParsCit system architecture. In the Figure, LS model and GS model stand for the manual labeled data and features extracted during training (Luong *et al.* (forthcoming)).

details), page, reference, sectionHeader, subsectionHeader, subsubsectionHeader, table, tableCaption, and title. On the other hand, for the GS subtask the following classes are considered: abstract, categories, general terms, keywords, introduction, background, related work, methodology, evaluation, discussion, conclusions, acknowledgments, and references.

The features devised for the LS component consist of *token-level* and *line-level* features. For the *token-level* features, the authors used similiar ones to those in the task reference extraction. Specifically, for the *line-level* features, the following features were used:

- **Location**: Encodes the relative position in the paper.
- **Number**: Checks the occurrrence of patterns, like (“1.1”) for subsectionHeaders, and (“1.1.1”) for subsubsectionHeaders.
- **Punctuation**: Whether the line consists of email addresses or web links (e.g., “[1]”, like in references or footnotes).
- **Length**: Measures the length of each line (takes values: 1token, 2token, 3token, 4token, 5+token).

For the CS component the authors demed a set of features for the header itself. Specifically those features are:

- **Position**: Encodes the absolute and relative position of a header in the papers.
- **First and Second Words**: Models the individual tokens of the header, as a way of differentiate the headers in the standard form, like *Abstract*.
- **Whole Header**: Models the whole header, acts as a memorization of all headers in the training data.

As raw text features lose performance when section headers are not marked in the text (e.g., numbered), ParsCit introduces a way for the learner (CRF) to use the information that is present in the positioning, font and size of the text. In order to accomplish this, ParsCit considers papers to be consecutive images, in a way to handle any paper that can be scanned. With this purpose, it uses an optical character recognition (OCR) engine that can extract fonts and the spacial layout of elements in the page. With the linearization of the information that the OCR outputs, a set of stationary features was devised. These features are meant to reflect the state of the current line of interest, e.g., whether it is bolded or italicized. These features consist of:

- **Location**: Encodes the relative position of a text line within a page.
- **Format**: Encodes the font information, in the form of FontSize, Bold, Italic. As the *relative font size* can be important to identify the different levels of headers.
- **Object**: Used to capture special line attributes, such as *Bullet*, *Picture* and *Table*.

The stationary features were proven useful in contrasting lines of different labels. In order to capture text blocks with multiple lines that use the same format, and assure that these lines are labeled consistently, additional features were created, namelly differential features. These features capture state changes between consecutive lines. However, these features are not based on the output of the OCR. Instead, they are synthesized from the output of stationary features in consecutive lines. These features are specifically:

- **Format**: This feature incorporates five properties, namelly *FonSize*, *Bold*, *Italic*, *FontFace*, *Alignment*, and takes the value *format_same*, if the five properties match or instead *format_new*.
- **Paragraph**: This feature is based on the tag for the paragraph *para*, from the output XML from the OCR, takes the values *para_new* for the first line of each text block, and the following lines take the value *para_same*.

2.2.1.3 ArnetMiner

ArnetMiner is a system that has the goal of extracting and mining academic social networks. From all the tasks this system addresses, the one that has more relevance to this work is the automatic extraction of researcher profiles from the Web, which resorts to Conditional Random Fields models for solving the labeling problem.

This problem consists in extracting the value of each property (e.g., name, affiliation, homepage, phone, research interest, etc.) in a person profile (researcher), which is a non-trivial problem to solve as the format of the researchers has much variations between different websites.

To solve this problem, a three step approach is proposed, relevant page identification, pre-processing and extraction. For relevant page identification and given a researcher name, the system acquires a list of web pages using a search engine (e.g, Google API) and then identifies the homepage using a binary classifier like an SVM. After obtaining this list in a pre-processing phase, the text of the pages is broken into tokens to which are assigned labels. In order to identify the tokens in a Web page, several heuristics are applied. Five types of tokens were defined, such as standard-word, special-word, term, etc., for example the special words were identified using regular expressions. After token identification the phase of token labeling starts, using a CRF model. For example, in the case of special words tokens, the labeling task consists, in assigning each token with the classes affiliation, email, address, phone, etc.

In order to accomplish the labeling of tokens, the CRF model uses three types of features, namelly (i) content features, like the morphology of the word, (ii) pattern feature, like whether the token contains a pre-defined positive/negative, word and (iii) term features, like whether a token contains a word in a dictionary. All considered features are boolean-valued functions.

2.2.1.4 Extraction of Bibliographic References from Patents

Patrice Lopez presented a system for the automatic extraction of bibliographic references for patent documents in multilingual texts (Lopez (2010)). The recognition process is done through the use of Conditional Random Fields. For the CRF used there are some variations, namelly some adaptations, depending if the task is related to patent documents or scholary articles, as in patent documents references may occur in the text, against to what happens in scholary articles that have the references in a separated section. These adaptations refer to the set of features used. Figure 2.5 gives an overview of the system processing realized. In a first step, the text body is extracted from the target document (e.g., PDF). The patent and non patent reference blocks are then identified in the text recurring to two different CRF models. Then non-patent references are annotated by a CRF for identifying a set of 12 classes (e.g., author, title, booktitle, etc.). Then both patent and non-patent, that result from the annotation via CRF are normalized by the use of regular expression. Afterwards in a last phase different online bibliographic services are accessed in order to validate the resulting reference.

For the generic CRF model, the features considered were:

- Current token;
- Prefix (i.e 1 to 4 characters of the current token);
- Suffix 1 to 4 characters of the current token;
- Capitalization information (takes the values *no*, *first letter*, *all*);

- Lower case form;
- Punctuation information for the token (e.g., hyphen);
- Number information for the token (takes values *no*, *one*, *all digit*);
- Length of the token;
- Year pattern and month pattern (both take boolean values);
- Common word (boolean value);
- Common US forname, common US family name and country name, (taking boolean values).

For features specific to patent references were used:

- Relative position of the token in the document;
- Common country codes (boolean value);
- Common kind code pattern (boolean value).

For non-patent literature were used:

- Relative position of the token in the document;
- Scientific and technical journal names (boolean value);
- Abbreviated names for sci-tech journal (boolean value).

Last, for isolated references a feature with the relative position of the token in the reference string was coded. The features referred are computed for each token and for the tokens in a 5-token window of the target token. Beyond the features referred, as this solution is for multilingual texts, sophisticated features of pattern matching and general lexicons for English, French and German were used.

2.2.2 Sequence Tagging models

This section presents works that proposed successful solutions for the problem of information extraction using Hidden Markov Models and Conditional Random Fields. The focus was given to works that experimented with information extraction in research papers (Mccallum & Li (2003); Zhou & Su (2002)).

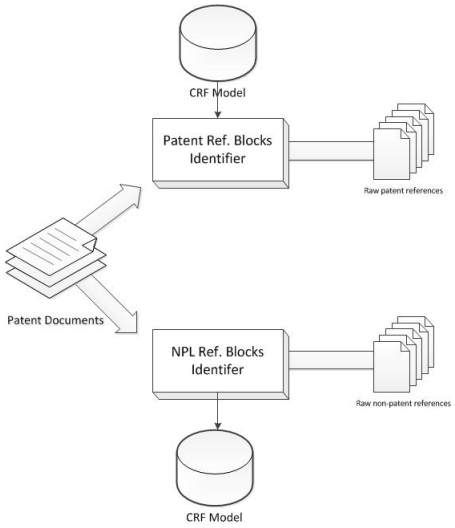

Figure 2.5: The bibliographic extraction processing system by Lopez (2010). The gray areas represent the components of the system that the section does not address. NPL stands for Non-Patent Literature (e.g., Scholary articles).

2.2.2.1 Hidden Markov Model

A work that explores Hidden Markov Models for information extraction in research papers is shown by in (Seymore *et al.* (1999)). This work refers that a typical approach using HMM with a fixed model structure of one state per class is not the best approach for solving the IE problem, and the concept of distantly-labeled data, i.e labeled data from another domain whose labels partially overlap those from the target domain (e.g., BibTex files that are bibliography databases that contain labeled citation information, in which labels such as the title, author or year occur in the research papers for example in the reference section and so can be used in the training for references extraction), that can improve classification accuracy.

The HMM proposed in this work uses an alternative to the typical solution of assigning one state per class by first using the training data to define a model structure, i.e each word in the training data is assigned with its own state, that transitions to the state of the word that follows it. Then merging techniques are afterwards applied, namelly neighbor-merging, i.e the combination of all states that share a transition and have the same class label (e.g., a sequence of adjacent abstract states from a single header are merged into a single state); V-merging, i.e two states that have the same label, and share transitions from or to a common state are merged (e.g., instead of having one state title that has many transitions to abstract states. This merging technique would merge all abstract states into one). This last merging technique presents the advantage of reducing the branching factor of the model.

2.2.2.2 Conditional Random Fields for IE

McCallum and Peng addressed the task of metadata extraction from research papers exploring several pratical issues in applying CRFs to information extraction, like (i) regularization with focus on first the Gaussian, exponential and hyperbolic-L_1 priors, and (ii) the families of features like text, lexicons and layout (Peng & McCallum (2004)). From the points addressed in this work, we will focus mostly on the feature families.

Given that feature induction has been shown to provide significant improvements in CRFs performance, McCallum and Peng addressed state transition features that correspond to different Markov order structures. Defining a general feature function over labels and a token sequence as $f_k(y_{c-1}, y_c, x)$, different state transitions features may be defined to form different Markov order structures. These features consist in, (i) first-order $f_k(y_c, x)$, where the input is evaluated in the context of the current state, (ii) first-order with the addition of state transition $f_k(y_c, x)$, $f(y_{c-1}, y_c)$, (iii) second-order $f_k(y_{c-1}, y_c, x)$, where the input is evaluated by the current and previous states, (iv) third-order in which the input is evaluated in the current the previous two states $f_k(y_{c-2}, y_{c-1}, y_c, x)$. Beyond these features, the authors explored local features, such as whether a token starts with a capitalized letter. Layout features, for example if the token is at the beginning of a line and external lexicon features, like a match of a token in author lexicon. Moreover a method for beneficially use of unsupported features was used, as it showed that those features with zero-count in the training data could be useful for to push the viterbi inference away from certain paths by assigning those features negative weight.

McCallum and Peng show that feature engeneering is essential part of conditionally trained models that have freedom to choose arbitrary features. Thus obtaining an improved performance in extracting metadata fields from research papers.

2.2.3 Improvements over HMM and CRF models

While research has been done in order to solve the problem of information extraction, this is still a recent area of research that is open to improvement. This section reviews previous work with the goal of extending and improving the Conditional Random Fields models that are most related to the subject of this dissertation. Specifically, work on semi-Markov conditional random fields, as it relates to the subject of this thesis and may be used improve the proposed solution in Chapter 3.

2.2.3.1 Semi-Markov Conditional Random Fields

The work by Sarawagi & Cohen (2004), uses semi-CRF models to solve the problem of sequence tagging by, given an input sequence x, outputting a segmentation of x in which labels are assigned to segments (set of words) of x, rather than to individual elements x_n of x. One important aspect when mentioning semi-CRF models is that their features can measure properties of segments and transitions within a segment can be non-Markovian.

The work by these authors has shown that a disadvantage of using semi-CRF would be in the necessity of maximizing over possible segment lengths. However, it is shown that this additional cost is only linear over L (upper bound on segment length), and semi-CRF also showed to have the advantage over conventional CRF of only considering sequences in which the same label is assigned to all L positions.

The semi-Markov CRF model, considering the previous presented Conditional Random Field model, defines a segmentation $s = s_1 ... s_c$ of x, where each segment $s_j = (t_j, u_j, y_j)$, which consists on a start position t_j, an end position u_j, and a label y_j. For the features the model assumes a vector $f = (f^1 ... f^k)$, each of which maps a (j, x, s) to a measurement $f^k(j, x, s) = f^k(y_j, y_{j-1}, x, t_j, u_j)$. So this defines a semi-CRF as being an estimator with the following form:

$$P_\Lambda(s|x) = \frac{1}{\sum_s \prod_{c=1}^{C} \phi_c(s_c, s_{c-1}, x, c; \Lambda)} \prod_{c=1}^{C} \phi_c(s_c, s_{c-1}, x, c; \Lambda) \tag{2.4}$$

As for the experiments the semi-CRF was trained to label segments with an encoding similar to the one referred in 2.1 having the difference that the Begin tag was not used. The semi-CRF was used to solve five named entity recognition problems, particularly the extraction of addresses, city names/states, company names, job titles and person names.

Some of the most important features that were used in the semi-CRF were word-level features and their logical extension to segments, namely indicators for the phrase inside a segment, indicators for words, and capitalization patterns in a 3-words window before and after the segment. External dictionary based features, similar to entity names in documents were also deed. Due to the variations in the way entity names are written, the authors used three similarity metrics, namelly Jaccard, TFIDF and Jaro Winkler (Cohen *et al.* (2003)). Another used feature were the internal segment dictionary of segments labeled as entities in the training data.

2.2.3.2 Hierarchical Conditional Random Fields

Lu *et al.* (2007) proposed an hierarchical approach that makes use of CRF models, using them for the problem of Chinese named entity tagging (NET). As Chinese words have strong dependencies between

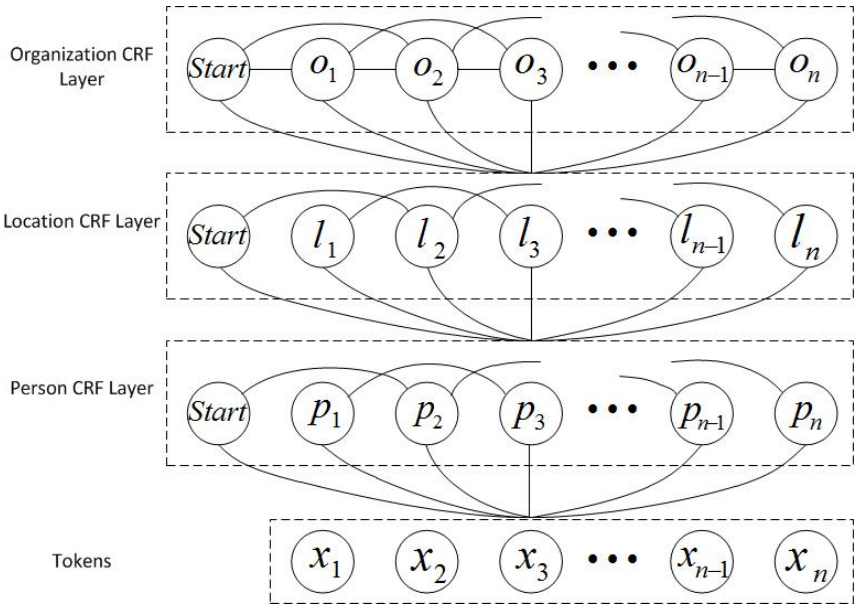

Figure 2.6: The Hierarchical Conditional Random Fields based on the figure presented in the research article (Lu *et al.* (2007)).

them, i.e Chinese words are compositional words, the segmentation of conventional CRF causes these approaches to show poor results. So, in order to solve the problem of Chinese NER this work uses an Hierarchical Conditional Random Field model of three layers, as it demonstrated a decrease in the propagation of errors in segmentation and labeling. Figure 2.6 provides a representation of the HCRF used in this work.

This HCRF uses three layers to solve the labeling problem, namelly (i) persons, as it is the simplest, (ii) location, and (iii) organizations. For the encoding of the CRF model, the authors used the BIO encoding. In the first layer, persons are labeled using a CRF model given a segmented text and part-of-speech (POS). Afterwards the token with the tag Begin_person and Inside_person are merged into person entity to modify POS of segmentation, and the result of this will be the token sequence of the next layer CRF. On the next layer the locations are labeled and then the tags Begin_location and Inside_location are merged like before. Following for the third layer the same process is repeated.

2.2.3.3 Semi-Supervised CRFs

Sequence labeling tasks that resorts to supervised classifiers, are trained with annotated text sequences, such as CRF and HMM, however using these approached introduces a dependency on training data. So often when using these approaches one is faced with the problem of not having available enough labeled data to use for training. As this data is difficult to obtain derived from its dependency on hand-labeling by human experts, with the scope of trying to solve this problem Qi *et al.* (2009) presents a semi-supervised method for sequence labeling with self-learned features (SLF), that can be used for CRF models.

This work focus on four classical IE tasks, such as named entity recognition, part-of-speech tagging, and one gene name recognition corpus. Namelly these SLFs present a way for training by averaging predicted labels over all cases in the unlabeled corpus, this approach builds class label distribution patterns for each word, and re-trains the model iteratively adding these distributions as extra word features, which as a result give improvement over supervised methods.

This semi-supervised strategy aims to provide semi supervision at the level of feature attributes rather than using examples, with the goal of avoiding the bias problem that semi-supervised methods suffers, i.e. examples that are labeled incorrectly and added may make the model even worse on the next iteration. Relying on the assumption that words have important label information, it is possible to derive self-learned features that relate to the distribution of target classes through large unlabeled corpus, and afterwards use these features (SLF) as extra features to retrain the model (CRF) in a iterative fashion as said before.

The basic SLF learns to model words probabilities to every possible target class introduced. In order to complement the basic SLF other three extensions for the SLF features were proposed, namelly (i) boundary SLF, i.e SLF that learns to represent words class boundary patterns, (ii) attribute SLF, i.e. models target class distribution patterns for each value of discrete word attribute, (iii) clustered SLF that clusters word according to SLF values, and the derived cluster ID would be used as extra features. Basic SLF specifically, is defined by a feature vector, that models the probability to each target class, of a certain word be assigned to it. As this distribution is unknown at the begin, it can be estimated from a training set or re-estimated using the unlabeled corpus, by applying a trained model that measures the proportion of text sequences assigned as a certain class given a specific word is present. The work also proposed a modification to the SLF in order to permit a window-based approach, as this is a common strategy for sequence segmentation. The extensions for SLF are namelly, (i) attribute SLF, i.e. words attributes, like capitalization for the class-distribution, (ii) boundary SLF, that incorporates a class boundary distribution, i.e models rare words which happen frequently before/after a certain class label, so that when the necessity of labeling those words arises its neighborhood words may give strong indications of its target class types, in cases that these words are always in the boundary of a certain class. Such extension is useful for tasks, such as named entity recognition or gene name recognition. (iii) clustered SLF uses a vector quantization technique in order to convert SLF distributions vectors to prototype vectors. Vector Quantization clusters groups of values together instead of one at a time, that in this case essentially means that all the words are clustered in multiple clusters, and then the clusters ID are used as feature representation (Gersho & Gray (1991)).

2.3 Summary and Critical Discussion

This chapter surveyed the general concepts behind the task of metadata extraction, such as the encoding used in this tasks for labeling, and the features commoly used in CRFs, followed by the probabilistic framework used in this field, namelly the Hidden Markov Models, and the current state of the art probabilistic framework Conditional Random Fields.

As for the related work, the chapter presents several works, focusing initially on the current systems that tackle with success the problem of metadata extraction. Next, it presents some of the successful work done in the past focusing on the models of HMM and CRF followed by some possible improvements for the performance of the models in question. Finally it can be concluded that even with the existent systems, that tackle the problem of metadata extraction from scholary articles (e.g., ParsCit), it is still possible to improve. Therefore two ways of improving the metadata extraction task for scholary articles, based on the related work reviewed are, (i) semi-Markov CRF, as it permits the use of a model, that evaluates the dependencies between the labels in a certain *interval,* unlike usual CRF that only depends on the previous label, which in case of sequence tokens may help to map the dependencies between the tokens, (ii) hierarchical models, as they permit to decompose the problem in different layers, allowing for temporal abstraction from observations at particular times, and (iii) the use of self-learned features in order to minimize the dependency from training data. In case of hierarchical models it is worth mention that when a labeling error occurs on a top level this error will propagate to the layers below, which may be a setback for this approach.

Chapter 3

Stacking CRF models for Metadata Extraction

Given the problem of recognizing individual metadata fields in scholarly articles, in my dissertation I use a generative model commonly referred to as Conditional Random Field that offers an efficient approach for addressing this sequence tagging problem. A particular novelty introduced in my proposal is the usage of different CRF models for specific logical segments of the scholarly papers, instead of a single model for the entire contents of a document.

This chapter presents the method developed, namelly the base structure of the hierarchy of Conditional Random Fields models, and the metadata fields considered by the system, followed by the process workflow, and its required pre-processing operations, ending with the features involved with the CRFs.

3.1 General Overview on the Proposed System

As referred in Chapter 1 I resort to Conditional Random Fields models to address the problem of metadata extraction by using the stacking of different CRF models, each that addresses different parts of the problem. These models are trained using annotations, each responsible for annotating the tokens from a specific part of the document according to particular metadata fields. The general structure of a scholarly article and the hierarchy of the CRF models of the system, in terms of its main logical parts and important metadata elements, corresponds to the hierarchy given in Figure 3.7.

With basis on the hierarchical structure illustrated above, each token from a scholarly paper is classified using two distinct CRF models, one corresponding to the logical structure of the paper (i.e., the five

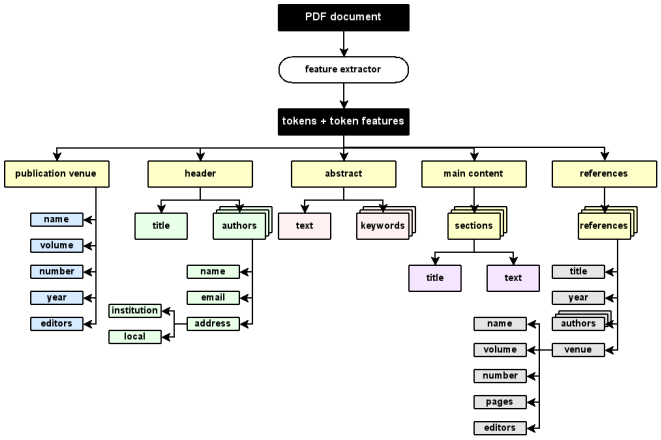

Figure 3.7: Hierarchical structure and important metadata elements of scholarly papers.

top different blocks shown in yellow over Figure 3.7) and another corresponding to the metadata fields that can be observed in the different logical blocks (i.e., the other colored blocks shown in Figure 3.7). Using this architecture based on stacking of multiple CRF models, instead of using a single CRF model considering a larger set of classes corresponding to the entire set of metadata elements, I hope to reduce the difficulty associated with model learning. Simpler models are less susceptible to data sparseness, thus requiring less training data. The training of the separate CRF models can also be made through datasets of different sizes, possibly from different origins, thus facilitating the gathering of training data.

Specifically, the first-level CRF model will distinguish between tokens that refer to (i) the publication venue of the paper, (ii) the header of the paper, (iii) the abstract and keywords of the paper, (iv) the individual sections that make up the content of the paper, and (v) the individual references that make up the bibliography of the paper. For each of the five classes considered in the first-level CRF model, there is a second-level CRF model that further segments the data. These models are as described bellow:

- A CRF model that analyzes the publication venue and distinguishes between the tokens that correspond to the name, volume, number, year and editors.
- A CRF model that analyzes the header of the article and distinguishes tokens that correspond to the title of the paper, the individual authors, the email addresses, the author institutions and the locations of these institutions.
- A CRF model that analyzes the abstract part of the paper and distinguishes tokens that belong to the text that composes the abstract from tokens that belong to individual keywords associated to the paper.
- A CRF model that analyzes each of the sections of the paper and distinguishes tokens that belong to section titles from tokens that belong to the contents of the individual sections.

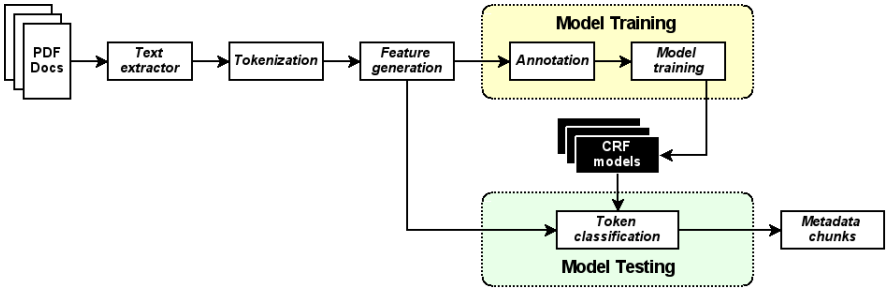

Figure 3.8: The PaperCut approach to metadata extraction from scholarly papers.

- A CRF model that analyzes each of the references in the paper and distinguishes tokens that belong to the title, publication year, individual authors, name of the journal, book or proceedings volume, the journal volume number, the issue number, the article's pages and the editors of the volume.

The different CRF models involve a set of common token features, which encode information relating to the textual contents, positions, and font styles.

3.2 System Worflow and Implementation

Besides the usage of CRF models, the proposed method for extracting metadata fields from PDF scholarly articles will also involve a series of pre-processing operations. Figure 3.8 provides an illustration for the envisioned approach.

The Portable Document Format (PDF) presents some problems to information extraction systems, mostly due to the fact that the order of textual objects in the PDF files does not always correspond to the reading order. Still, several tools for text extraction from PDF documents are nowadays available D'ejean & Meunier (2006); Hadjar *et al.* (2004). In this work, I use the PdfToHtml[1] for extracting the text from the PDF files to XML representations. This is a Java library for extracting the text from the PDF files that is particularly interesting, since it supports the extraction of font information associated with the text to the XML format.

After text extraction, the next steps in this approach will relate to tokenization and token feature generation. The contents of each publication P will be deterministically broken down into a sequence of tokens $\{x_1, x_2, \ldots, x_n\}$. This is made made through the tokenization functionality available in the LingPipe[2] package. Next, each token is going to be associated with a series of features, representing the evidence that will latter be used for classifying the tokens as belonging or not to a particular metadata

[1] http://sourceforge.net/projects/pdftohtml/
[2] http://alias-i.com/lingpipe/

field (i.e., according to the values of these features, each token is to be assigned the correct label from a set of classes $C = \{c_1, c_2, \ldots, c_m\}$, representing begin/in/out positions for the considered metadata fields). Section 3.3 details the set of suggested features, which encode information related to position, font style and textual contents.

This proposed approach for metadata extraction is based on supervised learning, thus requiring training data in the form of label class annotations for tokens described by features. For this are used ground-truth annotations (e.g., obtained with basis on the LaTeX sources associated to academic articles in arXiv[1], and latter manually revised) to learn a set of Conditional Random Fields (CRF) models, each responsible for annotating the tokens from a specific part of the document according to particular metadata fields. Further detail about the training data can be found in Chapter 4.

For the implementation, is used a package from the several available with linear-chain CRF, known as LingPipe[2] package, still there are other available like, Mallet [3] (that includes High-Order CRF, like semi-CRF) or CRF++[4] (Carpenter (2010)).

The trained CRF models can be used to assign the correct label to previously unseen tokens, with basis on the corresponding features. In the final step, after the tokens have been classified, the chunks of text corresponding to the individual metadata fields can be recovered by taking the sequences of textual tokens annotated as belonging to the same metadata field.

3.2.1 Framework Modifications

In order to allow the implementation of the architecture referred, I made some adaptations to the framework, namelly to the LingPipe framework. These adaptations aim to create a model (i.e., a chunker, an implementation of a CRF model) composed by a group of CRF models (hierarchical chunker), and to permit each of these models to consider the font size information associated with a certain token, instead of only considering the token itself.

In terms of the chunker that considers the font size, this modification is made possible by creating an extension of the representation of each token, to include the numeric value of each token, and adapting the existing chunker to read this values. In order to implement the hierarchical chunker, it was created an extension of the previously existing chunker, with a set of chunkers, instead of a single chunker. This implied an alteration to the way the chunker returns and evaluates the token list, in order to pass the output of a chunker to the remaining chunkers. To permit this behaviour, the start and end positions in each chunk are used in order to select the corresponding sequence of tokens in the original sequence,

[1] http://arxiv.org/
[2] http://alias-i.com/lingpipe/demos/tutorial/crf/read-me.html
[3] http://mallet.cs.umass.edu/
[4] http://crfpp.sourceforge.net/

in order to pass it to the remaining chunkers. To conclude the process of this evaluation, was coded a way to concatenate the results obtained by the chunkers.

The last modification made to the framework is due to the tag evaluation existent in LingPipe, in order to allow this framework to accept the previous created chunkers, this is done by changing the way in which the evaluator is created, allowing the evaluator to receive a chunker, from which the evaluator retrieves the tagger required for the evaluation process.

3.3 The Considered Features

The main advantage of CRF models over simpler approaches such as Hidden Markov Models (HMMs) comes from their modeling flexibility, which permits the feature functions to be complex, overlapping features of the input, without making additional assumptions on their inter-dependencies (Lafferty *et al.* (2001)). Taking inspiration from several previous works (Councill *et al.* (2008b); Peng & McCallum (2004)), the considered features in the CRF models in terms of the categories, token properties, lexicons and relative position is as follows:

3.3.1 Token Properties

- **Textual tokens :** The lowercased token identity for the target token.
- **Font size :** The font size for the target token, encoded as five binary features corresponding to the different font sizes (e.g., from tiny to Huge in LATEX).
- **Bold and italic font :** Whether the target token is given in bold, italics, bold and italics or name of the above.
- **Character case :** Whether the target token is given in a capitalized, lowercased, uppercase or mixed-case form.
- **Dashes and dots :** Whether the target token contains at least one dash, one dot, or neither of the above.
- **Single character alone :** Whether the target token contains a single uppercase character or an initial such as *A.*.
- **Punctuation :** Whether the target token is a punctuation character.
- **Prefix :** Whether the target token begins with a specific pattern.

- **Suffix :** Whether the target token ends with a specific pattern.
- **Numeric :** Wheter the target token corresponds to a number.
- **Year :** Whether the target token corresponds to a sequence of four numbers, thus being likely to correspond to a year.
- **URL :** Wheter the target token matches a regular expression for identifying Uniform Resource Locators (URLs).
- **Email :** Wheter the token matches a regular expression for email addresses.
- **Containment of a digit :** Wheter the target token contains digits.

3.3.2 Lexicons

- **Author name lexicon :** Wheter the target token appears in a list of author names which we compiled from sources such as the DBLP Computer Science Bibliography[1] or the Mathematics Genealogy Project[2].
- **Publication venue lexicon :** Wheter the target token appears in a list of names or abbreviations associated with popular conferences, journals and scientific publishers (e.g., tokens with a well-defined meaning such as ACM, IEEE, Elsevier, JCDL, SIGIR, TKDE, etc.) This lexicon was also compiled by us from sources such as the DBLP Computer Science Bibliography.
- **Month name lexicon:** Wheter the target token corresponds to the name of a month or a frequent abbreviation such as *Jan* or *Feb*.
- **Notes lexicon :** Wheter the target token appears in a lexicon containing words such as *appeared* or *submitted* that commonly appear in notes.
- **Abstract end lexicon :** Wheter the target token appears in a lexicon containing words such as *MSC, PACS* that are usually right after the end of the abstract.
- **Location lexicon :** Wheter the target token appears in a lexicon of location names, obtained from wikipedia.
- **Institution lexicon :** Wheter the target token appears in a lexicon containing words in different languages such as *institution* or *university*.

[1] http://dblp.uni-trier.de/
[2] http://genealogy.math.ndsu.nodak.edu/

3.3.3 Relative Position

- **Absolute position :** The target token's absolute position in the text.
- **Position relative to abstract :** Whether the target token occurs before or after the first occurrence of the token with lowercased identity *abstract*.
- **Position relative to references :** Whether the target token occurs before or after the first occurrence of the token with lowercased identity *references*.
- **Position relative to introduction :** Whether the target token occurs before or after the first occurrence of the token with lowercased identity *introduction*.
- **Position relative to keywords :** Whether the target token occurs before or after the first occurrence of the token with lowercased identity *keywords*.
- **Position relative to acknowledgments :** Whether the token occurs before or after the last occurrence of the token with lowercased identity *acknowledgments*.
- **Position relative to conclusion :** Whether the token occurs before or after the last occurrence of the token with lowercased identity *conclusion*.

All the above features are computed for each token in a document, as well as for the tokens within a 3-token window of the target token. Although first-order chain CRFs assume that the transitions between tags depend only on the origin and destination, the features can be made to represent longer sequences or even global information relating to the document.

I also use a small set of edge features, combining some of the previous items with the previous token tag, these features are as follows:

- **Font size :** Corresponds to the font size node feature combined with the previous tag.
- **Font size variation:** Wheter the target token font size has the same encoding as the previous token, combined with the previous tag.
- **Author name lexicon :** Corresponds to the author name lexicon node feature combined with the previous tag.
- **Location lexicon :** Corresponds to the location lexicon node feature combined with the previous tag.
- **Institution lexicon :** Corresponds to the institution name lexicon node feature combined with the previous tag.
- **Email :** Corresponds to the email name lexicon node feature combined with the previous tag.

3.4 Summary

This chapter presents the different parts involving the proposed method for metadata extraction in scholarly papers, namelly the metadata fields that are found in a scholarly paper, and were considered by the CRFs models involved in the presented method, such as, title, section title, author name, etc. Following, it presents the hierarchical structure of Conditional Random Fields models, trained for labeling tokens with a specific group of the considered metadata fields, with the goal of reducing the difficulty associated with model learning, making the models less susceptible to data sparseness, thus requiring less training data. Next, this chapter presents the worflow of the method, and the corresponding libraries, and tools, associated with the implementation of the workflow, such as, the PdfToHtml, used for the extraction of the PDF text, to a XML representation, and LingPipe, the library that implements the CRFs models to use. Finally the features involved with the CRF are presented.

Chapter 4

Validation

This chapter shows the evaluation methodology, used when evaluating the system, centering in the metrics used, and the training set, followed by the results obtained during the evaluation and the conclusions obtained when comparing the results against previous similar approaches.

4.1 Evaluation Metodology

The experimental validation of the system was based on a collection of 100 papers collected from arXiv from the domain of Mathematics, annotated according to most of the metadata fields listed in Section 3.1 (all fields except the ones referring to publication venues, which almost never occurred in the considered papers). The annotations were first collected automatically from the information available in the LaTeX sources associated to the articles, and later manually revised. For the metadata fields specific to bibliographic references, we relied on a larger dataset that had already been used in previous information extraction experiments, namely the Cora[1] dataset that contained research paper citations with labeled segments according to the considered fields. Each token had thus a BIO annotation corresponding to its relation to a logical part of the document (i.e., a part corresponding to a description the header of the paper, the abstract and keywords, individual sections, and individual references) and a BIO annotation corresponding to the specific metadata fields that can be observed in the different logical parts. This data set was used for training and validation, using 10-fold cross validation and comparing the generated annotations against those proposed by the method.

The results obtained from the evaluation of the system, are shown in terms of the precision, recall and F_1 metrics for the individual metadata fields considered by the system. More precisely the metric of

[1] http://www.cs.umass.edu/~mccallum/data.html

recall (i.e., a measure of completeness) consist in the fraction of correct results among the results that belong to the revelant subset. Precision (i.e., a measure of fidelity) consists in the fraction of correct results among those that the system believes to belong to the relevant subset. The F_1measure is the harmonic mean between the precision and recall. In the equations bellow tp stands for true positives, and fn stands for false negatives.

$$precision = \frac{tp}{tp + fp} \tag{4.5}$$

$$recall = \frac{tp}{tp + fn} \tag{4.6}$$

$$F_1 = 2 \,.\, \frac{precision.recall}{precision + recall} \tag{4.7}$$

The results are evaluated using two evaluation methods, one that verifies at each token if the correct the label is assigned, and another that verifies if each chunk corresponds to the correct metadata field.

Specifically in terms of the metrics above, the results are to be presented for each metadata field considered by the system.

The results obtained are also compared against previous approaches in order to obtain a better overview of the performance of the system (Councill *et al.* (2008b); Luong *et al.* (forthcoming); Peng & McCallum (2004)).

4.2 Results

Table 4.2 gives the precision, recall and F_1 metrics for the individual metadata fields considered by the PaperCut system. The obtained results show that the proposed approach can extract the composing structure of an article, with an F_1 score over 97%. As it can be observed in terms of the model for extracting the individual fields of bibliographic references, the obtained results are comparable against those obtained at previous works that also had used the Cora dataset, with an F_1 score over 82%, as shown in table 4.2. Overall, this model achieved some interesting results, except in some cases in which the low results may be attributed to the following reasons:

- Multiple different layouts that were associated to the articles used in the ground-truth collection;
- Failure by the tool responsible for extracting the PDF text, in terms of the recognition of the mathematical symbols, and accentuation, which led to a larger fragmentation of the text, thus losing the information of which tokens compose each line, and of which characters compose each token, that

	Token-based evaluation				Token-based evaluation			Chunk-based evaluation		
Large logical sections	Prec	Rec	F_1	Metadata fields	Prec	Rec	F_1	Prec	Rec	F_1
Header	1.00	1.00	1.00	Title	0.87	0.97	0.91	0.69	0.68	0.68
				Author Names	0.85	0.75	0.79	0.40	0.40	0.40
				Author Emails	0.92	0.80	0.85	0.02	0.01	0.01
				Institutions	0.80	0.73	0.76	0.01	0.02	0.02
				Locations	0.87	0.74	0.78	0.03	0.05	0.07
				Overall	0.86	0.80	0.82	0.21	0.22	0.22
Abstract	0.98	0.98	0.98	Keywords	0.97	0.84	0.89	0.02	0.04	0.02
				Text	0.99	0.99	0.99	-	-	-
				Overall	0.98	0.92	0.94	-	-	-
Sections	0.99	0.99	0.99	Title	0.52	0.81	0.61	0.22	0.17	0.11
				Text	0.99	0.99	0.99	-	-	-
				Overall	0.76	0.90	0.80	-	-	-
References	0.99	0.97	0.98	Title	0.95	0.95	0.95	0.90	0.86	0.88
				Year	0.97	0.96	0.97	0.92	0.76	0.84
				Authors	0.98	1.00	0.99	0.98	1.00	0.99
				Institution	0.90	0.90	0.90	0.83	0.71	0.77
				Location	0.88	0.92	0.90	0.77	0.74	0.76
				Book title	0.90	0.87	0.88	0.79	0.75	0.77
				Journal	0.81	0.82	0.82	0.71	0.73	0.72
				Editor	0.98	0.76	0.86	0.85	0.75	0.80
				Overall	0.80	0.90	0.91	0.84	0.78	0.82
Overall	0.97	0.98	0.97	Overall	0.85	0.88	0.87	0.66	0.30	0.29

Table 4.1: Token-based and chunk-based evaluation results.

would be useful in terms of the section title and author name identification;

- The small size of the training dataset, which was caused by the difficulty in finding (i.e, caused by the lack of structure in the articles at arXiv that was required to use features like font), and labeling articles.
- In terms of the header model, due to the identification of institution and locations, these have shown a low result due to the name of these entities being written in the native language of the author, making the use of certain lexicon features unsuccessful in these cases.

4.3 Summary

This chapter presents and discusses the results for each metada field resulting from the task of metadata extraction in scholarly articles by the proposed method. The main findings were are as follows:

- Overall the method previously proposed by ParsCit achieved best results for the task of metadata extraction in scholarly articles. However, the proposed method in most of the metadata fields

showed similar results, which demonstrates the validity of this method. Still in terms of the metadata fields that presented lower results, this may be due to the lack of annotations, difference in the layout of the various articles, and the presence of mathematical formulas, that the tools of extraction cannot extract successfully, which may be resolved by the use of OCR techniques.

- As for the part corresponding particularly to the labeling of the metadata fields composing the references, this model as an overall high result in most of the metadata fields, similiar to the ones present in ParsCit method, in terms of the tests done with the Cora dataset.

Chapter 5

Conclusions

This work addressed the problem of metadata extraction, specifically the problem of metadata extraction in scholarly articles, namelly from the domain of Mathematics. Given that after massive digitalization activities, current digital libraies hold large collections of scholarly materials, requiring the descriptive metadata that is essential in supporting the advanced search and retrieval functionalities that are expected of modern digital libraries. With the goal, of elaborating a method to obtain the descriptive metadata required, and based on the state of the art formalism Conditional Random Fields, this work presents a framework based on stacking of Conditional Random Fields together with a rich set of features, in order to identify the various metadata fields present in scholarly articles.

The method addresses the problem of metadata extraction by using the stacking of different CRF models, each addressing a different fragment of the problem. Namelly, this method is composed by a higher level model that identifies which part of the article belongs to either publication venue, header, abstract, sections and references, followed by five models each one responsible for identifying the metadata fields composing each of the previous parts of an article.

The system was is evaluated with a set of 50 papers collected from arXiv from the domain of Mathematics annotated accordingly. The results are obtained for each of the metadata fields considered by the system, in terms of precision, recall, and F_1score.

5.1 Summary of Contributions

Using of the proposed framework on papers retrieved from the arXiv, the results attest for the following contributions:

- The development of a framework based on stacking of Conditional Random Fields, to use in the task of metadata extraction from scholarly articles, allowed this architecture to use simpler models which reduced the difficulty associated with model learning and data sparseness, thus requiring less training data. Also this allowed the training of the separate CRF models to be made through datasets of different sizes, and from different origins, which was the case with the model that identifies the metadata fields present in each bibligraphic reference which use the Cora dataset, and the remaining models use the dataset build from arXiv.
- The evaluation of an rich set of features, that includes font information resultant from the extraction of the text from a PDF document, allowing the capture properties useful in the identification of certain metadata fields, like the article title or section title.
- Construction of a collection of papers fully annotated for the considered metadata fields, from the domain of mathematics, based on arXiv, an e-print service, allowing future experiments in this domain of metadata extraction from scholarly articles.
- The validation of the system, with an overall F_1 score of 91%, thus proving the usefulness of the method in obtaining the descriptive metadata required by digital libraries.

5.2 Future Work

Despite the interesting results, there are also many ideas for future improvements. The main challenge is related to improving the validation mechanism, through the usage of a larger set of annotated scholarly articles, and by evaluating the system against articles from other domains, such as computer science, in which the presence of mathematical formulas is reduced, thus improving the structure of the PDF text extracted, allowing a lower fragmentation of the text. Another means of improving the results would be the by adding lexicons for other languages, that are found in certain metadata elements of the papers. It would also be interesting to experiment with higher-order CRF models, capable of modeling transitions in longer token sequences (Ye *et al.* (2009)). The performance of our CRF models is also restricted by the availability of labeled training data, making it interesting to experiment with semi-supervised CRF models, capable of leveraging on small sets of hand-labeled data together with large amounts of unlabeled data (Qi *et al.* (2009)).

The metadata extracted through the method also presents some important limitations. The proposed method is, for instance, not capable of associating the extracted e-mail addresses and institutions to the corresponding authors. For future work, it would be interesting to extend the method in order to perform other related tasks besides the extraction of metadata from PDFs, such as citation matching and the normalization of author names, institutions and publication venues (Pasula *et al.* (2002)), the extraction

of keywords from the textual contents (Nguyen & Kan (2007)), or the classification of article citations according to different semantic dimensions (Teufel *et al.* (2006)) (e.g., self-citations, citations to articles with similar approaches, citations to articles with contrasting approaches, etc.). Specifically in case of articles from the domain of mathematics, it would be interesting to combine the approach proposed here with advanced OCR techniques for recognizing formulas in the publications (Suzuki *et al.* (2003)), since mathematical formulas can be used to support particularly interesting retrieval functionalities. Also an OCR would be useful to retrieve other properties from the text useful to the set features, like que size of lines, format of the text (i.e., centered, paragraph, etc.,).

Bibliography

CARPENTER, B. (2010). *Text Analysis with LingPipe 4.0*. LingPipe Publishing.

COHEN, W.W., RAVIKUMAR, P. & FIENBERG, S.E. (2003). A comparison of string distance metrics for name-matching tasks. In *IJCAI-2003 Workshop on Information Integration on the Web*, 73–78.

COUNCILL, I.G., GILES, C.L., IORIO, E.D., GORI, M., MAGGINI, M. & PUCCI, A. (2008a). Towards next generation citeseer: A flexible architecture for digital library deployment. In *Proceedings of the 10th European Conference on Research and Advanced Technology for Digital Libraries*.

COUNCILL, I.G., GILES, C.L. & KAN, M.Y. (2008b). ParsCit: An open-source CRF reference string parsing package. In *Proceedings of the 6th Language Resources and Evaluation Conference*, European Language Resources Association.

DAY, M.Y., TSAI, R.T.H., SUNG, C.L., HSIEH, C.C., LEE, C.W., WU, S.H., WU, K.P., ONG, C.S. & HSU, W.L. (2007). Reference metadata extraction using a hierarchical knowledge representation framework. *Decision Support Systems*, **43**, 152–167.

D'EJEAN, H. & MEUNIER, J.L. (2006). A system for converting pdf documents into structured xml format. In *Proceedings of the 7th IAPR International Workshop on Document Analysis Systems*.

DING, Y., CHOWDHURY, G. & FOO, S. (1999). Template mining for the extraction of citation from digital documents. In *Proceedings of the 2nd Asian Digital Library Conference*, 47–62.

GERSHO, A. & GRAY, R.M. (1991). *Vector quantization and signal compression*. Kluwer Academic Publishers.

GILES, C.L., BOLLACKER, K.D. & LAWRENCE, S. (1998). Citeseer: an automatic citation indexing system. In *3rd International Conference on Digital Libraries*, 89–98, Association for Computing Machinery Press.

HADJAR, K., RIGAMONTI, M., LALANNE, D. & INGOLD, R. (2004). Xed : a new tool for extracting hidden structures from electronic documents. In *Proceedings of the 1st International Conference on Document Image Analysis for Libraries*.

HAN, H., GILES, C.L., MANAVOGLU, E., ZHA, H., ZHANG, Z. & FOX, E.A. (2003). Automatic document metadata extraction using support vector machines. In *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*.

IVANYUKOVICH, A. & MARCHESE, M. (2006). Unsupervised metadata extraction in scientific digital libraries using a-priori domain-specific knowledge. *Semantic Web Applications and Perspectives*.

LAFFERTY, J.D., MCCALLUM, A. & PEREIRA, F.C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*.

LAWRENCE, S., GILES, C.L. & BOLLACKER, K. (1999). Digital libraries and autonomous citation indexing. *IEEE Computer*, **vol. 32**.

LOPEZ, P. (2010). Automatic extraction and resolution of bibliographical references in patent documents. In H. Cunningham, A. Hanbury & S. Rüger, eds., *Advances in Multidisciplinary Retrieval*, vol. 6107 of *Lecture Notes in Computer Science*, 120–135, Springer Berlin / Heidelberg.

LU, P., YANG, Y., GAO, Y. & REN, H. (2007). Hierarchical conditional random fields for chinese name entity tagging. In *Proceedings of the 3rd International Conference on Natural Computation*, vol. 5.

LUONG, M.T., NGUYEN, T.D. & KAN, M.Y. (forthcoming). Logical structure recovery in scholarly articles with rich document features. *International Journal of Digital Library Systems*.

MCCALLUM, A. & LI, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th conference on Natural language learning at North American Chapter of the Association for Computational Linguistics - Human Language Technologies*.

NGUYEN, T.D. & KAN, M.Y. (2007). Keyphrase extraction in scientific publications. In *International Conference on Asian Digital Libraries*.

PASULA, H., MARTHI, B., MILCH, B., RUSSELL, S. & SHPITSER, I. (2002). Identity uncertainty and citation matching. In *Proceedings of 15th Annual Conference on Neural Information Processing Systems*.

PENG, F. & MCCALLUM, A. (2004). Accurate information extraction from research papers using conditional random fields. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies*.

QI, Y., KUKSA, P., COLLOBERT, R., SADAMASA, K., KAVUKCUOGLU, K. & WESTON, J. (2009). Semi-supervised sequence labeling with self-learned features. In *Proceedings of the 9th IEEE International Conference on Data Mining*.

RAKESH DUGAB, U.D. (1996). A tutorial of hidden Markov models. Tech. Rep. SPANN-96.1, Signal Processing and Artificial Neural Networks Laboratory Department of Electrical Engineering Indian Institute of Technology.

SANG, E.F.T.K. & VEENSTRA, J. (1999). Representing text chunks. In *In Proc. of EACL 99: 9th Conference of the European Chapter of the ACL*, 173–179.

SARAWAGI, S. (2008). Information extraction. *Foundations and Trends in Databases*, **vol. 1**, 296–307.

SARAWAGI, S. & COHEN, W.W. (2004). Semi-markov conditional random fields for information extraction. In *In Advances in Neural Information Processing Systems 17*, 1185–1192.

SCHEFFER, T., WROBEL, S., POPOV, B., OGNIANOV, D., DECOMAIN, C. & HOCHE, S. (2002). Learning hidden Markov models for information extraction actively from partially labeled text. Kunstliche Intelligenz.

SEYMORE, K., MCCALLUM, A. & ROSENFELD, R. (1999). Learning hidden Markov model structure for information extraction. In *Proceedings of the AAAI Workshop on Machine Learning for Information Extraction*.

SUZUKI, M., TAMARI, F., FUKUDA, R., UCHIDA, S. & KANAHORI, T. (2003). INFTY : an integrated OCR system for mathematical documents. In *Proceedings of the ACM Symposium on Document Engineering*.

TANG, J., ZHANG, J., YAO, L., LI, J., ZHANG, L. & SU, Z. (2008). ArnetMiner: extraction and mining of academic social networks. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*.

TEUFEL, S., SIDDHARTHAN, A. & TIDHAR, D. (2006). Automatic classification of citation function. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

WALLACH, H.M. (2004). Conditional random fields: An introduction. Tech. Rep. MS-CIS-04-21, University of Pennsylvania.

YE, N., LEE, W.S., CHIEU, H.L. & WU, D. (2009). Conditional random fields with high-order features for sequence labeling. In *Proceedings of 23rd Annual Conference on Neural Information Processing Systems*.

ZHOU, G. & SU, J. (2002). Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.