HONOURS YEAR PROJECT REPORT

Citation Parsing Using Maximum Entropy and Repairs

By

Ng Yong Kiat

Department of Computer Science

School of Computing

National University of Singapore

2004/2005

HONOURS YEAR PROJECT REPORT

Citation Parsing Using Maximum Entropy and Repairs

By

Ng Yong Kiat

Department of Computer Science

School of Computing

National University of Singapore

2004/2005

Project No:	H79040
Advisor:	Assistant Professor Kan Min-Yen
Deliverables:	Report 1 Volume

Abstract

This thesis presents ParsCit, a system which parses citations from a publication or article, and label parts of citations with their corresponding field names. As citation styles differ between disciplines, publishers and authors, the task of citation parsing is difficult and inherently ambiguous. Whereas traditional citation parsers use a manually compiled set of rules to perform parsing, ParsCit adopts the framework of machine learning, learning rules for parsing from annotated data. A maximum entropy framework is employed to create a basic citation parse, and a series of repairs is performed to improve system accuracy. Given as few as 500 training examples, ParsCit is able to achieve a token accuracy of 94.20%, comparable to related machine learning approaches which lack the use of features applicable to the context of citation parsing. In this paper, I shall also emphasize the efficiency of the system's set of repairs, which is absent in citation parsers seen to date.

Subject Descriptors:

I.2.6 Learning

I.2.7 Natural Language Processing

Keywords:

Text processing, Maximum Entity, Citation Parser

Implementation Software and Hardware:

DELL PC Intel Pentium 4 CPU 1.60GHz with 256 MB RAM, Windows XP, Perl, HTML, Java, MAXENT Package, pdftotext from Xpdf

Acknowledgments

I would like to express my deepest gratitude to my Honours Year Project supervisor Assistant Professor Kan Min-Yen for his invaluable advice and encouragements throughout the course of my research. Assistant Prof. Kan is always there to patiently explain to me theories and concepts related to my work.

I would also like express my sincere thanks to all Computer Engineering course mates as well as friends from Honours Year Lab 7 for their enduring support.

List of Tables

List of Figures

Table of Contents

1 Introduction

In the scholarly communities, citations are a key part of linking discrete pieces of knowledge into a well-structured record of a field’s advancement. Although citations can come in many forms, a common standard places a reference to each paper, article or book used by the authors, in a separate section of the work. This list, often known as a bibliography, acts as an acknowledgement to these materials and provides exact publication information to identify each source and allow a reader to locate it for further study. Figure 1 below shows an example of a bibliography from a research paper.

References
[1] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and Autonomous Citation Indexing. *IEEE Computer*, 32(6):67–71, 1999.
[2] A. Odlyzko. The rapid evolution of scholarly communication. *Learned Publishing*, 2001. to appear.

Figure 1 Bibliography

Individual references have internal structure: the title, author, publication venue and date are all distinct fields. When writing, most authors manage references in a structured format, using tools such as BibTeX (Patashnik, 1988) and EndNote®[1] to sort, list and format the reference data. Figure 2 below shows an example entry in a BibTeX. Once the bibliography is converted to textual string in the publication using such tools, the explicit internal structure of each reference is lost (as in Figure 1). A reverse engineering procedure is required to infer the individual fields from each reference.

[1] EndNote® by Thomson ResearchSoft. http://www.endnote.com

```
@InProceedings{ bollacker98citeseer,
  author       = "Kurt Bollacker and Steve Lawrence and C. Lee Giles",
  title        = "{CiteSeer}: An Autonomous Web Agent for Automatic Retrieval
and Identification of Interesting Publications",
  booktitle    = "Proceedings of the Second International Conference on
Autonomous Agents",
  editor       = "Katia P. Sycara and Michael Wooldridge",
  publisher    = "ACM Press",
  address      = "New York",
  pages        = "116-123",
  year         = "1998"
}
```

Figure 2 Example of a BibTeX entry

ParsCit's objective is to recover this structure. It performs this process of citation parsing by examining each reference and identifying which field each token belongs to. Figure 3 shows a sample output of a single citation parsed by ParsCit. The extracted fields from each reference can then be used directly by another author. More importantly, the recovery of the internal structure of references is mandatory for various bibliometric analyses and citation indexing.

```
-----------------------------------------------------------------------------------------------
[1] J . F . Allen , Maintaining knowledge about temporal intervals , Comm . ACM
26 ( 11 ) ( 1983 ) 832 - 843 .
-----------------------------------------------------------------------------------------------
AUTHOR:
J . F . Allen

TITLE:
Maintaining knowledge about temporal intervals

DATE:
( 1983 )

JOURNAL:
Comm . ACM

VOLUME:
26 ( 11 )

PAGES:
832 - 843 .
```

Figure 3 Sample output from ParsCit

Automatic extraction of the structure in references has been achieved only in recent years, but manual effort is still largely required. ParaTools (Jewell, 2002) is a software module that performs this task using rigid predefined citation styles. It is thus limited in portability and achieves limited accuracy. CiteSeer (Lawrence, 1998) does citation parsing by using heuristics and probabilistic models that also requires manual data collection: the method requires the manual collection of a large database of author names and journal names to identify reference fields. To correct system errors, CiteSeer employs Error Distribution Correction (Lawrence, 1999), allowing users to provide correction to any fields of a publication. Again, this involves manual supervision.

There have also been centralized attempts at creating unambiguous references by means of having a standard format. In (Cameron, 1997), Cameron proposed a universal citation database that required citations in a standardized format. This would negate the need for citation parsing. However, this meant that all authors would have to conform strictly to the citation formats, and the concept did not gain much acceptance. I believe that any centralized approach is bound to fail as individuals often make mistakes or may want to present their references differently than others.

In the next chapter, I shall touch on works related to citation parsing. In Chapter 3, I detail the ParsCit algorithm, which is the core contribution of my research. A detailed, comparative evaluation of ParsCit follows in Chapter 4. I conclude with an examination of applications and areas of future work in Chapter 5.

2 Related Works

The problem of citation parsing has been the focus of past research initiatives, as documented in the literature. I shall examine existing citation parsers, which can be generally divided into two categories: template matching and machine learning based approaches.

2.1 Template Matching Approach: ParaTools

A template matching approach takes an input citation and matches its syntactic pattern against known templates. The template with the best fit to the input is then used to label the citation's tokens as fields. The canonical example of a template based approach is ParaTools (short for ParaCite Tools), released by the University of Southhampton as part of its larger citation indexing project. ParaTools is a set of Perl modules that parses references into component fields using template matching.

This technique works fairly well for citations which adhere to simple citation patterns, but is susceptible to errors when it tries to extract fields from citations with many punctuations, since there may be multiple templates that fit equally well. If the wrong template is chosen, entire fields will be tagged incorrectly. As references can be parsed accurately only if their citation style resides among the existing templates, any template based approach is limited by the coverage of its templates. This is the case with ParaTools. While ParaTools contains 400 templates, my experiments with the system show that even this large amount manifests coverage problems. While users may choose to add new templates to ParaTools manually, the process is cumbersome and there might be indefinitely many styles for citing different material. The fact that authors may not strictly adhere to citation styles also diminishes any efforts to improve template matching techniques by template mining.

A further weakness of ParaTools is that it tags certain fields as ambiguously as "Any" as seen in Figure 4. Tagging a token as "Any" is equivalent to not tagging the token in any field, and while it simplifies the problem for ParaTools, places more burden on human annotators and post-processing editing.

Authors	:	M. Kitsuregawa, H
Title	:	Tanaka, and T
Publication	:	Moto-oka
Any	:	Application of hash to data base machine and its architecture
Any	:	New Generation Computing, 1(1)
Year	:	1983

Figure 4 Sample output of a citation parsed by ParaTools

A recent thesis (Huang, Ho, Kao and Lin, 2004),reported ParaTool's precision as approximately 30%. In my view, this low level of performance and lack of portability make the approach unsuitable for doing post-parsing processing, such as citation indexing (Garfield, 1979).

2.2 Machine Learning Based Citation Parsing Techniques

The limitations of the template-based approach have encouraged researchers to try alternative models for citation parsing. The remaining approaches that I will survey are all examples of machine learning approaches. Given sufficient training data, a machine-learned parser can produce high performance in accuracy, regardless of citation styles.

Hidden Markov Models (HMMs) are a powerful probabilistic tool and has been applied extensively on various language related tasks. HMMs are a finite state automaton with stochastic state transitions and symbol emissions (Rabiner, 1989). Associated with each set of states, $S = \{s_1, s_2, ..., s_n\}$ is a probability distribution over the output symbols $V = \{w_1, w_2, ..., w_n\}$, in the emission vocabulary. The probability that state s_i will emit the symbol w is $P(w \mid s_i)$. Similarly, associated with each state is a distribution over its set of outgoing transitions. The probability of transiting from state s_{i-1} to s_i is then $P(s_i \mid s_{i-1})$. A dynamic programming solution called the Viterbi algorithm is used to find the most likely state sequence given a Hidden Markov Model M and a sequence of symbols, This algorithm has a time complexity of $O(TN^2)$, where T is the length of the sequence and N is the number of states in M.

HMMs may be used for citation parsing by formulating a model in the following way: each state is associated with a citation field name (hereby called "tag") such as title, author or date. A labeled training dataset is first used to train a HMM. This model is then used to recover the most-likely state sequence that produces the sequence of observation symbols (i.e. the word tokens of a citation). For example, a sequence of symbols:

```
Steve Lawrence . ( 1999 ) Digital libraries and Autonomous Citation Indexing .
```

may have the most likely state sequence:

```
author author author date date date title title title title title title title
```

While HMMs models the $w_t \leftrightarrow x_t$ and $s_t \leftrightarrow s_{t+1}$ relationships as depicted in Figure 5, the modeling of relationships $V \leftrightarrow s_t$ are not permitted. In another words, we cannot use several w_t's to predict s_t as the Markovian assumption states that the emission and the transition probabilities depend only on the current state.

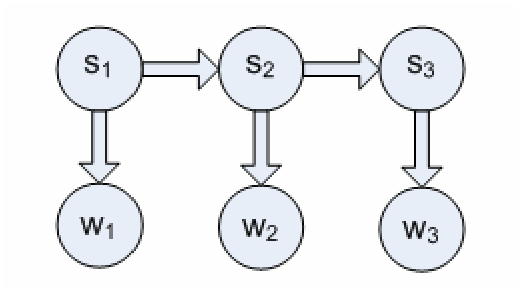


Figure 5 Relationships between states and output symbols in a HMM

In (McCallum and Peng, 2004), a HMM is used to train on labeled training citation data provided from the now-defunct Cora publication service. Citations are then parsed by using the trained model to label each word token of a citation. McCallum reports an overall accuracy of 93.1% for the HMM based parser. Despite of significant success in various sequence labeling tasks such as Part-of-Speech (POS) tagging (Ratnaparkhi, 1996) and predicting protein sequence labeling, the HMM method is less effective due to its assumptions of independent and non-overlapping features.

In the same paper, McCallum also examined the use of Conditional Random Fields (CRF) model to parse citations in another experiment. CRFs are undirected graphical models trained to maximize a conditional probability (Lafferty, McCallum and Pereira, 2001) and have been applied on tasks such as name entity extraction. A citation parser based on a CRF model outperforms one that is based on HMM, boosting overall accuracy to 95.37%. However training time is a concern, as CRFs converge slowly. It requires approximately 500 iterations for the model based on the same training set to stabilize.

In view of the above drawbacks and the absence of any well-documented CRF packages, I pursued another machine learning model. The Maximum Entropy Model provides flexibility given sufficient training datasets and serves as a balance between the two mentioned machine learning models.

Table 1 below summarizes the pros and cons of the various techniques discussed. (Characteristics on the last 2 columns shall be explained in full detail in the next chapter)

	Portability	**Package Avail.**	**Training Time (for 350 lines)**	**Word Accuracy**	**Repeated Fields**	**Takes Global Context into Account**
ParaTools	Cumber-some	Open Source (Perl)	Not Required	~ 30%	No	No
HMM	Yes	QTAG[2] (Java)	< 1 sec	~ 93%	Yes	No
CRF Model	Yes	Not Available	500 iterations	~ 95%	Yes	No
ME Model	Yes	MAXENT[3] (Java)	---	---	Yes	No

Table 1 Table of pros and cons of various approaches

[2] QTAG 3.1 by Mason, O. http://web.bham.ac.uk/O.Mason/

[3] opennlp.maxent Package version 2.3.0. http://maxent.sourceforge.net

2.3 Maximum Entropy Modeling

The model used for parsing in ParsCit is based on the maximum entropy model. Using this model, Ratnaparkhi obtained state-of-the-art accuracy of 96.6% in POS tagging (Ratnaparkhi, 1996). Since then, maximum entropy techniques have widely used for natural language tasks such as identifying sentence boundaries (Reynar and Ratnaparkhi, 1997) and text classification (Nigam, Lafferty and McCallum, 1999).

Maximum entropy (ME) is a probability distribution modeling technique. The principle of ME modeling is to model all that is known and assume nothing about which is unknown. In other words, given a training dataset, choose a model consistent with all the constraints, but otherwise as uniform as possible.

ME is used commonly to solve classification problems, in which the objective is to estimate a classification function $cl : X \rightarrow Y$, which maps an object $x \in X$ to its correct class $y \in Y$. X consists of words or sentences and Y is the predefined set of classes. Given a sentence of n words, we define $\boldsymbol{x} \in$ {the n-word sentence}. POS tagging is the process of predicting a sequence of n tags corresponding to the sentence, where $\mathbf{x} \in \mathbf{T^n}$ and $\mathbf{T}$ are the allocable POS tags for a word. This is a typical example of sequence tagging. Such complex problems are decomposed into a sequence of simpler classification problems by having a classifier predict the class for each object x. Since tags are predicted in sequence in POS, a classifier can exploit the previously predicted n-1 tags to predict the n^{th} tag of the n^{th} word.

In ME, the classification function is implemented with a conditional probability model p by choosing the class with the highest conditional probability:

$$cl(x) = \arg\max_{y} p(y \mid x)$$

Similarly, probability models can implement a complex classifier $cl : X^n \rightarrow Y^n$ for sequence tagging:

$$cl(x_1 ... x_n) = \arg\max_{y_1 ... y_n} \prod_{i=1}^{n} p(y_i \mid x_1 ... x_n, y_1 ... y_{i-1})$$

where $x_1 \ldots x_n$, $y_1 \ldots y_{i-1}$, informally called the "context" or "history", is the textual material available for predicting the class for the i^{th} object. Under the ME framework, the probability for a class *y* and object *x* depends solely on "features" that are active for the pair (*x*, *y*). A predefined feature is defined as a function $f:(X,Y)\rightarrow\{0,1\}$, i.e. this feature is either active or inactive. Features are the means through which problem-specific information is added to the ME model. They encode information deemed useful in classifying the objects. The importance of each feature is determined automatically by a parameter estimation algorithm over a pre-classified dataset (to be explained in the next section).

Functions known as contextual predicates are used in features. If $A=\{a_1...a_n\}$ represents the set of possible classes in the ME model and *B* represents the set of possible textual material that we can observe, then a contextual predicate is represented as the function

$$cp: B\rightarrow\{true,\ false\}$$

which returns either true or false, corresponding to the presence or absence of useful information in some context $b\in B$. Features are hence functions of the form

$$f: A \text{ x } B \rightarrow \{0, 1\}$$

As such, features will have the form

$$f_{cp,a'}(a,b) = \begin{cases} 1 \text{ if } a = a' \text{ and } cp(b) = true \\ 0 \text{ otherwise} \end{cases}$$

2.4 Machine Learning Approach under the ME Framework

In the machine learning approach, evidence required to make classifications of an object can be combined by weighting the features appropriately. With a training corpus $\mathcal{T}=\{(a_1,b_1)...(a_n,b_n)\}$, each tuple (a_i,b_i) is a pair of contextual feature b_i with its respective correct class a_i, the weight of each feature can be derived from the log-linear model (Ratnaparkhi, 1998):

$$p(a\,|\,b)=\frac{1}{Z(b)}\prod_{j=1}^{k}\alpha_j^{\,f_j(a,b)} \qquad (2.1)$$

where *k* is the number of features and *Z*(*b*) is a normalization factor (2.2) used to ensure that $\sum_a p(a\,|\,b)=1$.

$$Z(b) = \sum_{a} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)} \tag{2.2}$$

Parameter α_j in the model is a non-negative weight to its corresponding feature f_j. The probability $p(a \mid b)$ is then a product of the weight of features that are present (or "active"), i.e. features f_j such that $f_j(a,b) = 1$, normalized over $Z(b)$. Since the objective of ME is to maximize the uncertainty (hence the name maximum entropy) so as to get a model assumes nothing about which is unknown, ME maximizes

$$H(p) = -\sum_{a \in \{x,y\}, b \in \{0,1\}} p(a,b) \log p(a,b)$$

the entropy averaged over τ while keeping to constraints the model.

Given *k* features, each constraint has the form

$$E_p f_j = \sum_{a \in \{x,y\}, b \in \{0,1\}} p(a,b)\, f_j(a,b)$$

in which $E_p f_j$ is the expectation of feature f_j for model *p*.

In classification problems, we are interested in predicting the class of an object, given observations and evidence. Hence, the goal is to find an estimate for the conditional probability $p(a \mid b)$. Under ME, Ratnaparkhi (1998) in his thesis gives the optimal solution, p^*, which is the most uncertain distribution that satisfies the *k* constraints on the feature expectations such that:

$$p^* = \arg\max_{p \in P} H(p)$$

$$H(p) = -\sum_{a,b} \tilde{p}(b)\, p(a \mid b) \log p(a \mid b)$$

$$P = \{ p \mid E_p f_j = E_{\tilde{p}} f_j,\ j = \{1...k\} \}$$

$$E_{\tilde{p}} f_j = \sum_{a,b} \tilde{p}(a,b)\, f_j(a,b)$$

$$E_p f_j = \sum_{a,b} \tilde{p}(b)\, p(a \mid b)\, f_j(a,b)$$

where $E_{\tilde{p}} f_j$ denotes the observed expectation of a feature f_j, $\tilde{p}(a,b)$ denotes the observed probability of (*a*, *b*) in a fixed training set, and *P* denotes the set of probability models consistent with the observed evidence.

Generalized Iterative Scaling (GIS) algorithm (Darroch and Ratcliff, 1972) can then be used to compute the weights $\alpha_1 ... \alpha_2$ of the probability distribution p^*.

The use of ME in citation parsing is motivated by its claim of accuracy, knowledge-poor features and reusability. Ratnaparkhi's thesis (Ratnaparkhi, 1998) provided experimental support for state-of-the-art accuracy in various natural language processing tasks. It is also mentioned that feature sets used "rely less on linguistic knowledge, preprocessing, or semantic databases than competing approaches". Hence, it is a candidate for a machine learning based citation parser, given an appropriate set of predefined features which contains contextual information. Besides, the fact that the model assumes nothing apart from a given set of constraints, a citation parser built upon this framework should also perform reasonably well for citations from different disciplines, even if the training set is small.

Tasks such as sequence labeling problems benefit from a richer representation of observations, in particular a representation that describes observations in terms of overlapping features. While Hidden Markov Model is unable to represent overlapping features, the ME model is adequate in this aspect.

Maximum Entropy Model has its drawbacks as well. As with other machine learning models, we cannot easily represent that states (fields) cannot be repeated after appearing once in a parse sequence. In other words, the maximum entropy model occasionally repeats fields which are only allowed to appear once in a citation (as depicted in Table 1). For example, the citation:

"**Stuart, A.,** 1984. The Ideas of Sampling., **Charles Griffen,** London ."

when parsed by a machine learner may have both "Stuart, A." and "Charles Griffen" labeled as 'author' when the latter should be properly labeled as 'publisher'. This is a common problem of most machine learning approaches to citation parsing. Since fields generally do not repeat in the citations, such errors can be easily detected.

3 ParsCit System Architecture

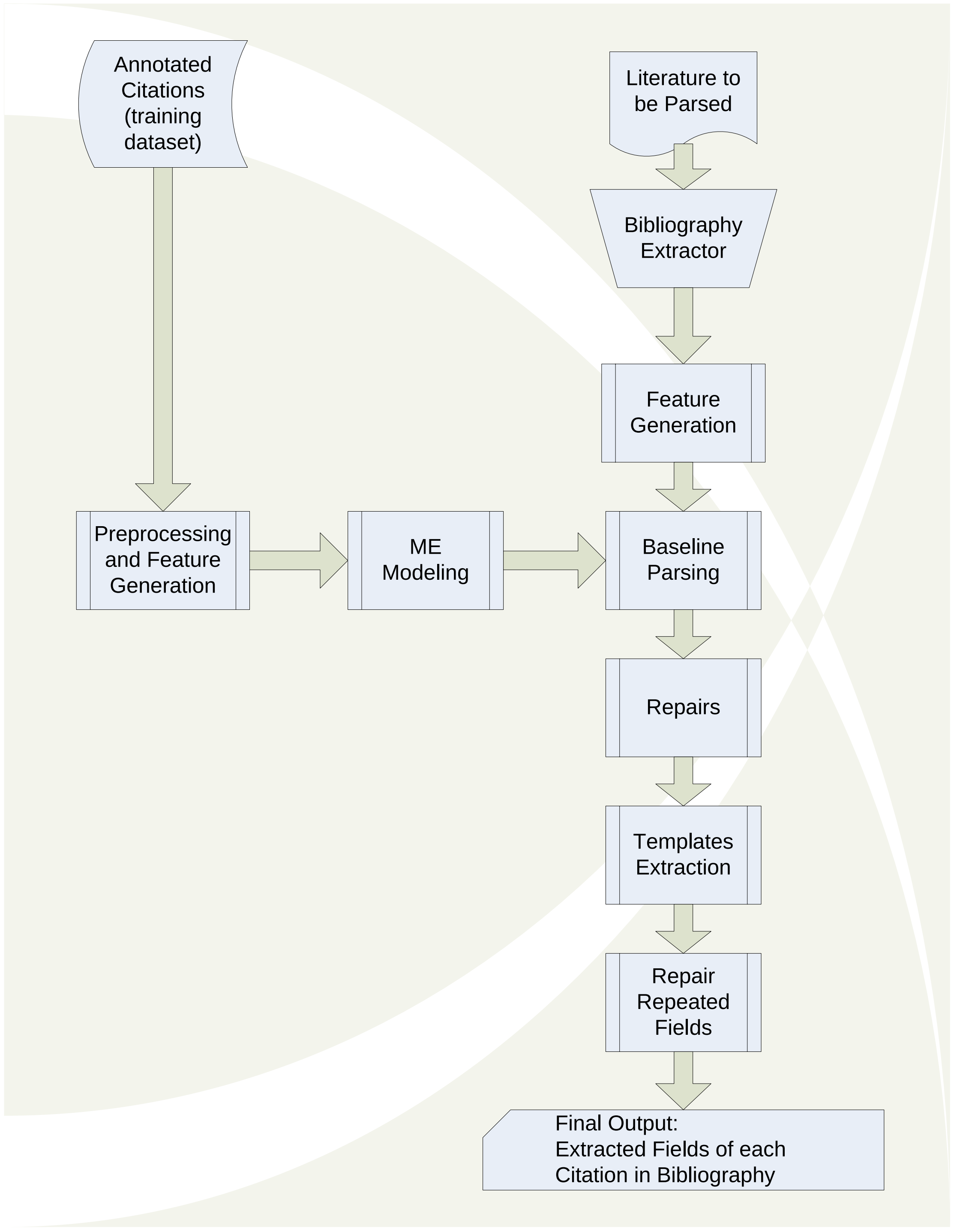


Figure 6 System Flow of ParsCit

Figure 6 shows the flow of the ParsCit system. This chapter will go into the explanation of each step. The main contribution of this work is twofold: the selection of features used in the context of citation parsing and the repairs done on a baseline parse in section 3.5. This set of repairs, which distinguishes ParsCit from other citation parsers, shows considerable improvement in accuracy as evaluated in Chapter 4.

3.1 ParsCit System Features

Features **F1 to F19** used in training and testing by ParsCit are specific to the context of citation parsing. The features can be classified into the following categories:

- **Lexical Features**

  These features observes the morphological features of the current token and that of its surrounding tokens

- **Layout Features**

  Features which observe
  - position of token relative to the entire line of the current citation
  - special characteristics of surrounding tokens
  - tags of surrounding tokens

- **Local Features**

  Features which are derived from the current token. These features observes certain characteristics of the current token.

- **Dictionary Features**

  Features used to indicate if the current token belongs to any of the dictionaries incorporated into ParsCit. The vocabulary includes keywords or tokens deemed as useful in predicting the tags. Table 2 below shows the dictionary used in ParsCit. A database of 97900 male names, female names and last names from various languages gathered from various sources[4] are used in the IsName feature.

[4] Names from various sources:
ftp://ftp.funet.fi/pub/doc/dictionaries/DanKlein/
http://www.census.gov/genealogy/names/
http://www.geocities.com/Tokyo/3919/atoz.html

Table 2 below lists all the System Features in ParsCit.

Set	Feature	Conditions and Examples
Lexical		

F1	Pre1, Pre2… Pre4
	Suf1, Suf2… Suf4
		Prefix and suffix of the current token.
		For a word token 'Artificial', features
		Pre1=A
		Pre2=Ar
		Pre3=Art
		Pre4=Arti
		Suf1=l
		Suf2=al
		Suf3=ial
		Suf4=cial
		are generated

Layout		
F2	Wi-2, Wi-1, Wi, Wi+1, Wi+2	Wi-1 as previous token, Wi as current token, Wi+1 as next token
F3	TV, TVV, TVVV, TVVVV	Tags of previous tokens

F4	FieldV, FieldVV, FieldVVV	Fields prior to the current token.
		For a token currently at the 4th field in a citation, with the first 3 fields 'author', 'date', 'title',
		FieldV=title
		FieldVV=datetitle
		FieldVVV=authordatetitle

F5	Posn	The position, ranging from 1 to 10, of a token relative to the line of citation is calculated. Given a 3rd token of a 5 token citation line, Posn=6 for this token
F6	BoundV	Previous token is a potential field boundary (ie the tokens ',', '.', ';')
F7	(Pre)Next	Previous token is '(' Next token is ')'
F8	NumPre NumNext	Previous token is a number. Next token is a number
F9	PunctPre PunctNext	Previous token is a punctuation Next token is a punctuation
F10	/Pre, /Next	Previous token is '/' Next token is '/'
Local		
F11	Year	Contains a 4-digit number in the form 18XX, 19XX, or 20XX. e.g. '1999a', '1999b'
F12	1DigNum, 2DigNum, 3DigNum, 4DigNum	Current token is a 1-digit number, 2-digit number, 3-digit number or 4-digit number
F13	ContainsDIG	Contains numerical digits

		e.g. ‘4th'
F14	OneCap	Is a single capital letter. e.g. ‘A’, ‘K’
F15	InitCap AllCaps	Begins with a capital letter. e.g. ‘Proceedings’ All capital letters e.g. ‘IEEE’
Dictionary		
F16	MonthName	Current token is a month name e.g. ‘January’, ‘Jan’
F17	IsName	Current token is a name or last name
F18	EdKeys	Current token is a keyword common in ‘editor’ fields. e.g. ‘Ed’, ‘edited’
F19	PgKeys	Current token is a keyword common in ‘page’ fields e.g. ‘pp’, ‘page’, ‘pg’, ‘-’

Table 2 Features used in ParsCit

3.2 Preprocessing of Training Set

ParsCit’s system training requires an existing corpus of correctly tagged citations. The dataset, hereby defined as **C**, used in ParsCit was created by the Cora project (McCallum, Nigam, Rennie and Seymore, 2000). **C** is a dataset of 500 correctly tagged citations, made up of a total of 12,153 word tokens. The 13 tags used in this dataset are: author, title, editor, booktitle, date, journal, volume, tech, institution, pages, location, publisher and note. ParsCit uses the same set of 13 tags in its implementation for citation parsing for ease of comparison with previous work. A sample of tagged citations from **C** is shown in Figure 7.

```
<author> A. Cau, R. Kuiper, and W.-P. de Roever. </author> <title> Formalising
Dijkstra's development strategy within Stark's formalism. </title> <editor> In C. B.
Jones, R. C. Shaw, and T. Denvir, editors, </editor> <booktitle> Proc. 5th. BCS-
FACS Refinement Workshop, </booktitle> <date> 1992. </date>

<author> M. Kitsuregawa, H. Tanaka, and T. Moto-oka. </author> <title>
Application of hash to data base machine and its architecture. </title> <journal>
New Generation Computing, </journal> <volume> 1(1), </volume> <date> 1983.
</date>

<author> Alexander Vrchoticky. </author> <title> Modula/R language definition.
</title> <tech> Technical Report TU Wien rr-02-92, version 2.0, </tech>
<institution> Dept. for Real-Time Systems, Technical University of Vienna,
</institution> <date> May 1993. </date>
```

Figure 7 Sample of tagged corpus from Cora dataset

Dataset **C** has to be preprocessed in order obtain features required to create a ME model. The procedures to do this are as follows:

- Tokenize **C**
- Append correct tag to each token accordingly (including punctuations) to form (token, tag) pairs
- Add <start> and <end> tokens to each citation

First, **C** is tokenized so that punctuations are separated as tokens by themselves; otherwise, '1993' and '1993.' would be treated as two different tokens. The citations are re-tagged such that each token is followed by its corresponding tag, forming (token, tag) pairs. I employ a chunk-based technique used for tagging each token similar to that was used for Named Entity Recognition in (Borthwick, Sterling, Agichtein and Grishman, 1998). For any particular tag among the 13, take for example 'author', a token could be in any of the four states:

- author%HEAD
- author%BODY
- author%TAIL
- author%SOLO

where I define the trailing 'HEAD', 'BODY', 'TAIL' and 'SOLO' as chunk-descriptors. Instead of using 13 tags to define the citation fields, an extended set of 52 tags is used as seen in Figure 8. For example, during system training, an entire ‘page’ field

```
<pages> pp. 168–175 </pages>
```

from **C** will be re-tagged as

```
pp_pages%HEAD ._pages%BODY 168_pages%BODY -_pages%BODY
175_pages%TAIL
```

Fields that contain only a single word token will be retagged with “%SOLO” appended to the field name. The training dataset after the preprocessing procedures is defined as **C’** (please refer to Appendix B – Sample from Training dataset **C’**).
Finally, a ‘<start> <start>’ pair is then added before the first token of each citation to indicate the beginning of the citation and a ‘<end> <end>’ pair appended to the end of it. **C’** now consists of a dataset of citations in the form of (token, tag) pairs.

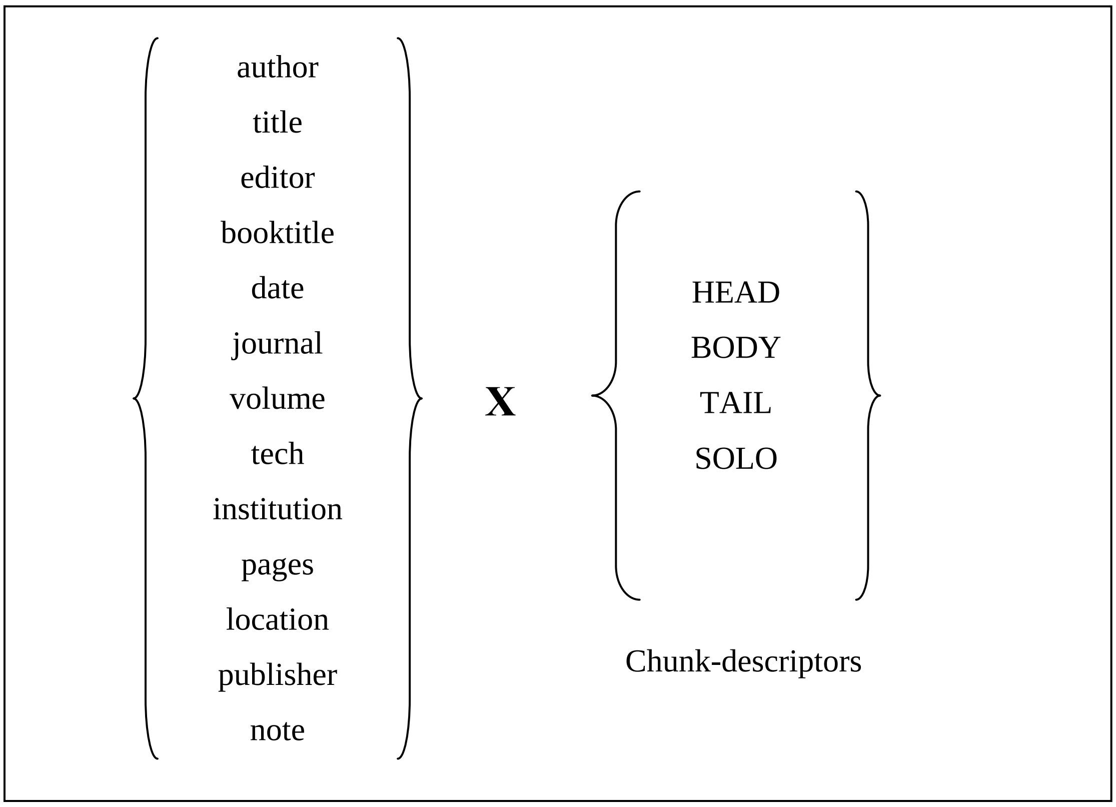


Figure 8 Extended set of 52 tags used for system training

3.3 Creating a Maximum Entropy Model

Features from annotated citations in the form of (token, tag) tuples are generated from **C'**. For each word token in **C'**, features listed as in Table 2 are generated.

For the token 'Software' in the following annotated citation line in **C'**,

> Ostroff , J . S . , **" Temporal Logic for Real - Time Systems " ,** *Advanced Software Development Series* , England , 1989 .
>
> *Note: annotated citations in this thesis will hereafter be "color-font-coded" for easier viewing. Please refer to Appendix B – Legend for Fields*

some of the features generated for the token 'Software' are listed in the following table:

Features Generated for 'Software'	**Explanation**
Pre1=S Pre2=So Pre3=Sof Pre4=Soft Suf1=e Suf2=re Suf3=are Suf4=ware	Its lexical features
Wi-1=Advanced Wi=Software Wi+1=Development	The surrounding words
TV=booktitle%HEAD TVV=title%TAILbooktitle%HEAD	The previous two tags
FieldV=author FieldVV=authortitle	'Software' is in the 3rd field. Its previous fields are 'author' and 'title'
Posn=7	19th token out of 26 total tokens gives a position of value 7

Table 3 Features generated for the annotated citation. See Table 2 for feature descriptions.

A training file consisting of pairs of features and the correct tag associated with each pair is then used for training. ParsCit uses an open source package, opennlp.maxent (Baldridge, Morton, Bierner and Friedman, 2001) to create a probability model p^* based on this training file.

GIS is used to find optimal weights. The time complexity of each iteration *is O(N P A)*, where *N* is the training set size, *P* is the number of predictions, and *A* is the average number of features active for each prediction. The number of iterations used for ParsCit's training is set at 100 since using more iteration produces only minute accuracy gains.

3.4 Baseline Tagging based on ME Model

A similar feature generation for each word token is done for the process of citation parsing. The only difference is that features F3 and F4 are dependent on the predictions made for previous tokens. In the example above, given that the previous token ‘Advanced’ is tagged as ‘booktitle%HEAD’, feature ‘TV=booktitle%HEAD’ would have been appended to features corresponding to the token ‘Software’ when ParsCit is tagging ‘Software’. Since ParsCit does tagging token by token from left to right, F3 and F4 for each test token are created on the fly.

After each token is tagged using the probability model trained previously, the chunk-descriptors are removed, resulting in tuples of (token, tags). This output is considered the baseline parse.

3.5 Repairs on the output of ParsCit’s Baseline Parse

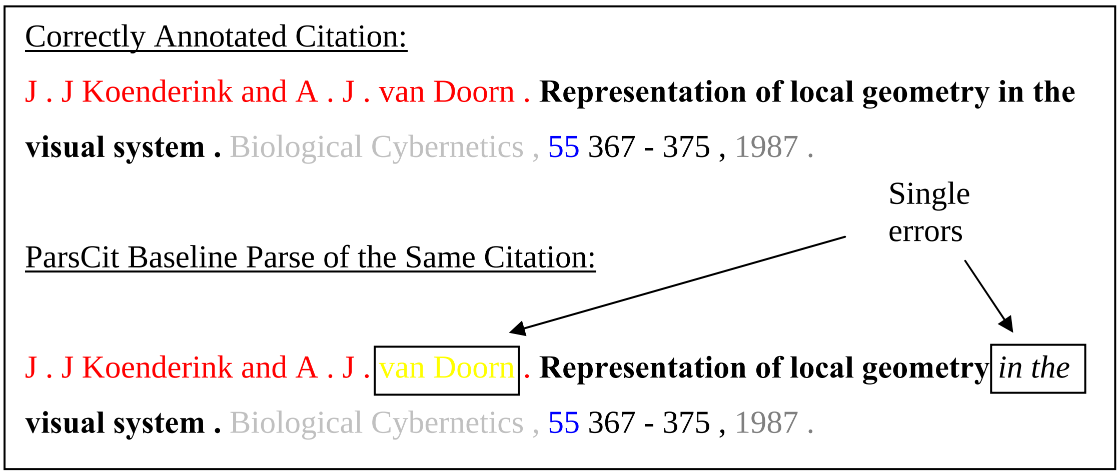
A detailed analysis on the results of the baseline parsing shows that there are a number of ways in which they can be repaired to gain a higher accuracy of extracted fields. A confusion matrix **M**, derived from the ten-fold cross validation on **C’**, is one that shows the number of word tokens from one field (TagX) that are incorrectly tagged as another field (TagY). The confusion matrix **M** after a 10-fold cross validation baseline parse is as follows:

TagX / TagY	author	title	editor	btitle	date	journal	volume	tech	instn	pages	location	publr	note
author	5308	138	174	8	18	20	4	4	9	1	9	3	2
title	12	4403	16	76	2	32	1	5	16	4	10	4	26
editor	17	0	385	36	3	6	5	0	0	0	4	3	3
btitle	0	15	3	2083	22	73	26	13	8	4	27	15	25
date	1	10	12	7	1493	3	4	6	3	10	14	3	9
journal	1	26	3	46	0	671	5	22	6	2	14	3	15
volume	0	2	1	11	11	4	608	2	3	9	8	2	1
tech	0	4	0	8	0	2	0	289	5	0	1	0	6
instn	1	0	0	1	3	3	1	5	350	2	27	5	1
pages	0	4	1	4	16	3	14	8	1	1311	11	0	9
location	0	2	0	8	7	1	1	2	7	1	446	6	12
publr	3	0	0	11	2	8	0	6	8	1	1	319	2
Note	0	1	1	1	2	11	0	0	0	0	0	0	80
TOTAL	5343	4605	596	2300	1579	837	669	362	416	1345	572	363	191
Acc	99.3%	95.6%	64.6%	90.6%	94.6%	80.2%	90.9%	79.8%	84.1%	97.5%	78.0%	87.9%	41.9%

Table 4 Confusion Matrix on output of baseline parsing

An examination of the output shows that errors are often due to sequences of tokens mistagged as another tag, i.e. all tokens in a sequence are tagged incorrectly as X when they should be tagged as Y. I define errors of this type as *single errors*. As to be explained in section 3.5.4, parses with one repeated field after repairs are also considered single errors. ParsCit attempts to isolate single errors from the baseline parse and perform appropriate repairs on them.

Figure 9 below shows an example baseline parse of a single citation with 2 single errors.



Correctly Annotated Citation:

J . J Koenderink and A . J . van Doorn . **Representation of local geometry in the visual system .** Biological Cybernetics , 55 367 - 375 , 1987 .

ParsCit Baseline Parse of the Same Citation:

J . J Koenderink and A . J . van Doorn . **Representation of local geometry** *in the* **visual system .** Biological Cybernetics , 55 367 - 375 , 1987 .

Figure 9 Correctly annotated citation and its ParsCit baseline parse with single errors

In this section, I shall go into the details of the series of repairs done by ParsCit based on the output of a baseline parse:

- Repairing of Editors
- Repairing of Bubbles
- Repairing of Repeated Fields (with respect to global context)

3.5.1 Repairing of Editor Field

As derived from **M**, the baseline parser performs poorly at tagging editor fields in citations. The ‘editor’ field has a token accuracy of only 64.6%, with 82.5% of its incorrectly tagged tokens tagged as ‘author’. This is largely due to the similar nature of these two fields. Also, even though the author field appears as the first field in citations most of the times, there are citations with the editor field (and no author field in the citation) appearing as the first field.

Editor names are almost certainly followed by certain cue words in a citation, e.g.

In Bouma , H . , & Elsendoorn , A . G . (**Eds** .) , Working Models of Human Perception , pp . 391 - 410 . Academic Press , London , England .

ParsCit attempts to locate these keywords in each citation and do the repairs according to the repairing algorithm:

```
Define cue word-tokens set, E as {“ed”, “eds”, “editor”, “editors”}
For each citation,
        Get index i of any word token that belongs to E
        For each token before i (going leftwards from i),
                If token is tagged as author, retag as editor
                Else break
        End for
End for
```

3.5.2 Repairing of Bubbles

Another problem of the output of a baseline parse is that a small number of tokens are tagged incorrectly amongst neighboring word tokens which are tagged correctly and are of the same tag. In this thesis, I define such bounded tokens as *bubbles*. Following the same example in Figure 9, tokens ‘in’ and ‘the’ are incorrectly tagged as ‘booktitle’, while their immediate neighboring tokens were all correctly tagged as ‘title’.

Bubbles are errors inherent for taggers for sequence labeling tasks as tokens are tagged iteratively from left to right. Such bubbles are removed by retagging it with the same tag as its left and right neighbor tags. ParsCit determines which bubbles to repair by heuristics based on statistics derived from **C**. A table of statistics Table 5 below shows the distribution of each field according to its length in tokens in **C,** and the proposed bubble size to be repaired.

	Length of Field (less then or equal to)					
Field name	**<=1**	**<=2**	**<=3**	**<=4**	**>4**	**Proposed repair size (<=)**
author	0.00%	0.41%	2.65%	18.57%	81.43%	3
title	0.20%	0.40%	1.41%	4.45%	95.55%	3
editor	0.00%	0.00%	0.00%	0.00%	100.00%	3
booktitle	**0.00%**	**0.43%**	5.21%	9.13%	90.87%	**2**
date	0.00%	31.25%	55.44%	97.17%	2.83%	1
journal	0.60%	8.43%	34.94%	48.19%	51.81%	1
volume	**10.99%**	39.56%	42.31%	64.29%	35.71%	**-**
tech	1.64%	8.20%	36.07%	42.62%	57.38%	1
institution	0.00%	1.72%	15.51%	31.03%	68.97%	2
pages	1.04%	2.08%	3.46%	51.90%	48.10%	3
location	0.73%	18.98%	29.20%	68.61%	31.39%	1
publisher	0.00%	9.90%	53.46%	90.10%	9.90%	1
note	0.00%	20.00%	46.67%	43.33%	46.67%	1

Table 5 Field lengths and proposed bubble size to be repaired

Using a threshold of 5%, ParsCit attempts to remove bubbles according to the statistics reflected above. For example, ‘volume’ appears as a single token 10.99% of the time, ‘volume’ bubbles are not repaired. ‘Booktitle’, on the other hand, rarely has a field length of less than 3, thus ‘booktitle’ bubbles of token length less than 3 are retagged by ParsCit.

The baseline parse followed by the repairs discussed thus far makes up the output of ParsCit’s first iteration.

3.5.3 Repairing of Repeated Fields

It is uncommon to see field repetitions in a line of citation, i.e. a field does not normally appear as two or more separate chunks in the same line. In **C**, only 4% of the citations contain repeated fields. This motivates a re-tagging to remove repeated fields.

How is this repair done? I consider a novel source of information, previous ignored by other citation parsers: the citation’s context within a bibliography. Citations appearing within a single bibliography tend to follow matching styles. As mentioned in Chapter 1, many authors use tools for managing references. As such, citations appearing from the same paper may follow a set of fixed template. I leverage this property by attempting to eliminate repeated fields by matching with probable templates. These probable templates are derived from the first iteration of parsing. A table of keys below is used for naming templates.

Field Name	**Key**	**Field Name**	**Key**
Author	A	Tech	K
Title	T	Institution	I
Editor	E	Pages	G
Booktitle	B	Location	L
Date	D	Publisher	P
Journal	J	Note	O
Volume	V		

Table 6 Table of keys of fields

A template for each citation is derived from the parsed output after the various repairs mentioned have been done. Only templates which do not contain repeated fields are added to a list of "seen" templates, **T**. For example, if the citation below

> L . G . Valiant . **A Bridging Model for Parallel Computation .** Communications of the ACM , 33 (8) , 103 - 111 , 1990 .

is parsed correctly by ParsCit, the template derived would be ATJVGD. With this template list **T**, citations which contain one repeated field after parsing are repaired by choosing a matching template (if available) from **T**.

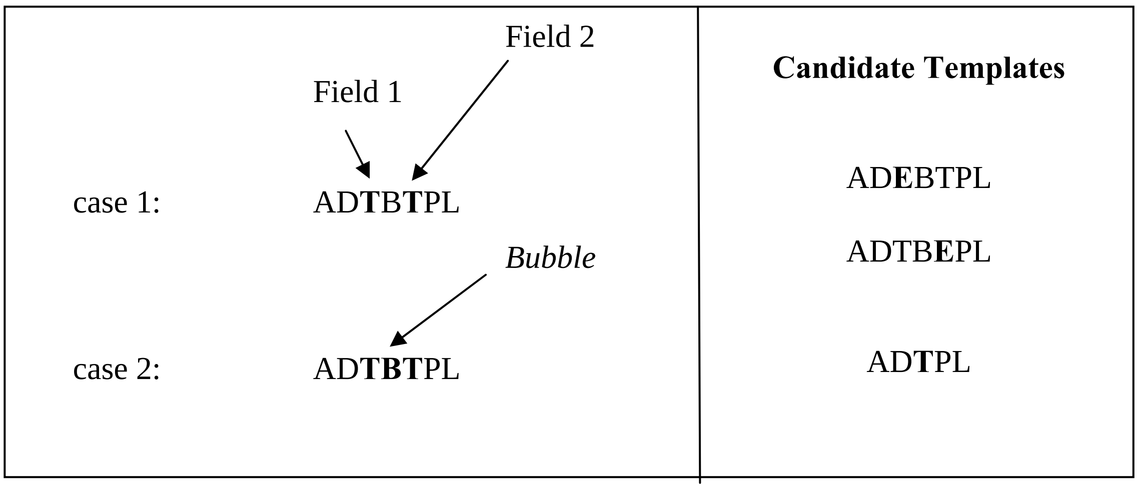


Figure 10 Templates with one repeated field

A template with one repeated field can hence be repaired in 3 ways:

Case 1:

- Re-tag tokens in Field 1 to tag X if candidate template exists in **T**
- Re-tag tokens in Field 2 to tag X if candidate template exists in **T**

Case 2:

- Remove bubble by retagging bubble with the same tag as its neighbor tags, if this candidate template exists in **T**

Where there are more than one candidate templates that can be applied, the template with highest observed frequency is used.

4 Overall Performance Comparison

Adopting a common performance measurement, this thesis defines the following:

A: number of true positive tokens (e.g. ‘author’ tokens tagged as ‘author’)
B: number of false negative tokens (e.g. non-author tokens tagged as ‘author’)
C: number of false positive tokens (e.g. ‘author’ tokens tagged as anything but ‘author’)
D: number of true negative tokens (e.g. non-author tokens tagged as anything but ‘author’)

$$\textit{Token Accuracy} = \frac{A+D}{A+B+C+D}$$

$$\textit{Precision} = \frac{A}{A+C}$$

$$\textit{Recall} = \frac{A}{A+B}$$

$$F_1 = \frac{2 \times \textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

Overall Acc.:	percentage of tokens whose predicted tag equals their true field name
Instance Acc.:	percentage of citation lines in which all fields are extracted correctly
Field Acc.:	percentage of total fields which are identified and extracted successfully in all citations tested

4.1 Evaluation of ParsCit Features

Morivated the success of POS tagging in (Ratnaparkhi, 1998), ParsCit started off with features F1, F2 and F3 similar to that was mentioned in the thesis. Finalized features F1 to F19 are then used for the baseline parse of the system. The results of using different features are shown in Table 7.

Features	Token Acc.	Average F_1	Inst Acc.	Field Acc.
Ratnaparkhi POS Features	90.17%	82.07%	42.60%	82.93%
Lexical + Layout + Local	92.19% (+2.2%)	85.37% (+4.0%)	45.60% (+7.0%)	84.62% (+2.0%)
Lexical + Layout + Local + Dictionary (baseline)	92.51% (+2.6%)	85.39% (+4.0%)	45.60% (+7.0%)	85.05% (+2.6%)

Table 7 Performance of ParsCit using different sets of features

In the context of citation parsing, we are concerned with the ability of the system to extract fields accurately; it is more useful to evaluate the performance by looking at the field accuracy, rather than token accuracy. As shown in Table 7, there is significant improvement in field accuracy with the increasing set of features used in ParsCit.

4.2 Comparison between Methods of Tagging

	HMM		CRF		ParsCit (baseline)	
Overall acc.	73.99%		95.37%		92.51%	
Instance acc	4.40%		77.33%		45.6%	
	Acc (%)	**F_1 (%)**	**Acc (%)**	**F_1 (%)**	**Acc (%)**	**F_1 (%)**
author	89.9	82.8	99.9	99.4	97.8	96.2
booktitle	92.5	71.2	97.7	93.7	97.7	90.3
date	98.1	88.8	99.8	98.9	99.1	94.7
editor	97.0	40.8	99.5	87.7	98.5	72.8
institution	98.3	53.3	99.7	94.0	99.4	85.9
journal	96.8	59.2	99.1	91.3	98.4	81.3
location	97.8	57.1	99.3	87.2	99.1	83.8
note	98.6	31.0	99.7	80.8	99.3	55.7
pages	96.5	67.9	99.9	98.6	99.5	96.1
publisher	98.7	58.3	99.3	76.1	99.6	88.1
tech	98.6	51.4	99.4	86.7	99.5	85.4
title	90.1	81.1	98.9	98.3	97.9	95.6
volume	97.9	62.5	99.9	97.8	99.4	91.4
Average F_1-measure		62.0		91.5		85.4

Table 8 Extraction results (Accuracy measured in tokens)

Table 8 above shows the experimental results of tagging citations using HMM and CRF compared to ParsCit's baseline parse. Ten-fold cross validation to eliminate bias towards any training set is done on each method of tagging:

- HMM using QTAG
- ME model using opennlp.maxent with ParsCit features F1 to F19

Accuracy values for citations tagged using CRF are taken from a very recent paper (McCallum and Peng, 2004) also based on the same training set **C**. However, it can only be used as a rough comparison as the test was done on 150 randomly chosen citations, with the remaining 350 used for training.

Out of the 500 citation lines tagged under cross validation, only 22 lines have had all their fields correctly extracted for a HMM-based citation parser (4.4% instance accuracy). ParsCit's baseline parser achieves overwhelming increase in performance as compared to this method, with an instance accuracy of 45.6%, and has comparable performance with the CRF method in terms of overall token accuracy.

4.3 Evaluation of ParsCit Repairs

	ParsCit (baseline)		ParsCit (repair)	
Overall acc.	92.51%		94.20%	
Instance acc	45.6%		60.8% (+33.3%)	
	Acc (%)	**F_1 (%)**	**Acc (%)**	**F_1 (%)**
author	97.8	96.2	99.3	98.7
booktitle	97.7	90.3	97.8	90.9
date	99.1	94.7	99.4	96.2
editor	98.5	72.8	98.4	90.9
institution	99.4	85.9	99.5	87.1
journal	98.4	81.3	98.5	82.0
location	99.1	83.8	99.1	83.4
note	99.3	55.7	99.4	59.0
pages	99.5	96.1	99.4	95.9
publisher	99.6	88.1	99.6	89.0
tech	99.5	85.4	99.4	84.5
title	97.9	95.6	98.4	96.7
volume	99.4	91.4	99.5	92.1
Average F_1-measure		85.4		88.2

Table 9 Performance comparison between baseline parse and repairs

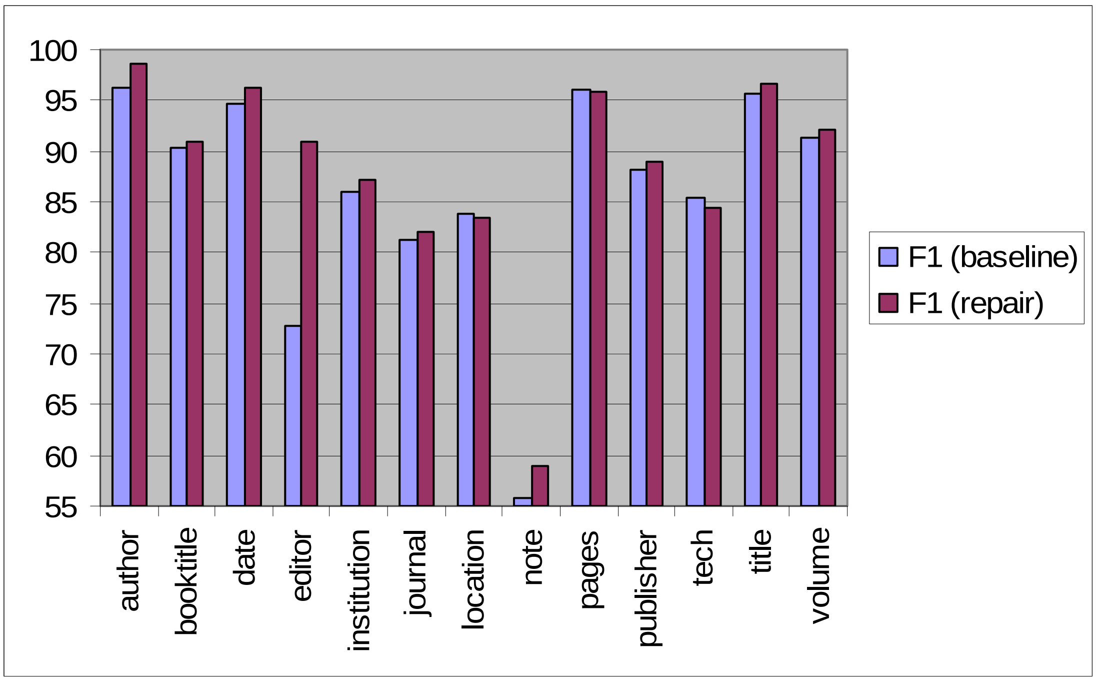


Figure 11 F_1 comparison between baseline parse and repairs

Table 9 above shows a performance comparison of ParsCit System before and after repair. There is an increase in per-field F_1-measure for almost all fields as depicted in the graphical representation Figure 11. Repairs on the baseline parse gives a significant improvement of 33.3% in instance accuracy.

58.9% of the citations parsed incorrectly from a ParsCit baseline parse contain only single errors, which ParsCit is capable of correcting. 449 tokens categorized as single errors out of 801 (56.1%) were successfully corrected using repairs mentioned in the previous chapter.

The same set of repairs is done on the baseline parse of the HMM-tagged output on **C** to prove its effectiveness. By defining *good parses* as citations from which at most one complete field is not extracted correctly, we have the following comparison in Table 10.

	Field Acc.	**Instance Acc.**	**Good Parse**
HMM (baseline)	53.7%	4.4%	24.0%
HMM (repairs)	57.8% (+7.6%)	6.6% (+50.0%)	31.0% (+29.2%)
ParsCit (baseline)	85.1%	45.6%	78.6%
ParsCit (repairs)	90.0% (+5.86%)	60.8% (+33.3%)	88.0% (+12.0%)

Table 10 Evaluation of repairs done on HMM and ParsCit baseline

Given the effectiveness of repairs done by the system, it is believed that these repairs will also improve the performance of a CRF-based citation parser, to gain an estimated instance accuracy of 85%, when used together with features developed in this thesis, i.e. out of 20 citations parsed, 17 will have all their fields extracted successfully.

5 Conclusion and Future Work

ParsCit is an citation parser which eliminates manual effort to hand-annotate citations of a literature work by using a statistical machine learning technique, maximum entropy. ParsCit is different from previous citation parsers in that the parsing process does not require citations to fit into any of a database of fixed templates. To my knowledge, it is the first citation parser that repairs *single errors* and leverages of contextual template information when parsing citations from the same bibliography. This information is utilized in repairing citations parsed with one repeated field.

I have shown in this thesis the success in the features and repairs incorporated into the parser, achieving a field accuracy of 89.95%. Due to the nature of the ME model, ParsCit is also expected to perform with considerable accuracy for citations of various templates and disciplines, even though the training set is small.

With each citation a bibliography parsed successfully, fields such as 'authors', 'title' , etc. extracted can be used to build a citation database. This process is critical for citation analysis like citation indexing. A citation index (Garfield, 1979) indexes the link between the articles published and the cited material. As summarized in (Lawrence et al, 1998), citation indexing can

> *"improve scientific communication by revealing relationships between articles, drawing attention to important corrections or retractions of published work, identifying significant improvements or criticisms of earlier work, and helping limit the wasteful duplication of prior research."*

Research paper search engines such as CiteSeer (Lawrence et al, 1998) Cora (McCallum et al, 2000) provide various detailed analyses based on citation indices, providing great convenience to researchers. ParsCit, with its flexibility and portability, allows a database to be built and makes it possible for citation indexing even for a researcher's personal use.

Repairs mentioned in this thesis can be employed on similar natural language processing tasks which demands structures or metadata to be extracted from a sequence of text. Such tasks include the extraction of various common fields from headers of research papers, classified ads, etc.

Inherent of all supervised machine learning techniques, a large, labeled training set helps to improve the performance of ParsCit. A web-based annotation system has been set up to allow volunteers to hand-annotate a random set of citations to further expand the existing labeled training dataset. Web spidering techniques may also be applied to automatically search for literature works from which ParsCit will attempt to parse. Parses which are deemed to be good will then be appended to its existing training set.

Additional supporting databases (such as a journal name database, publisher database, country name database, etc.) can also be collected to help the system identify and extract fields.

References

Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. (1998). Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. In Proceedings of the Sixth Workship on Very Large Corpora.

Cameron R. D., (1997). A Universal Citation Database as a Catalyst for Reform in Scholarly Communication. First Monday. Vol. 2, No. 4.

Darroch, J. N. and Ratcliff, D. (1972). Generalized Iterative Scaling for Log-linear Models. Annals of Mathematical Statistics, Vol.43, No.5, pp. 1470-1480.

Garfield, E. (1979). Citation Indexing: Its Theory and Application in Science, Technology, and Humanities.

Huang, I.A., Ho, J.M., Kao, H.Y. and Lin, W.C. (2004). Extracting Citation Metadata from Online Publication Lists Using BLAST.

Jewell, M. (2002). ParaCite: An Overview.

Lafferty, J., McCallum, A., Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proceedings of International Conference on Machine Learning, pp. 282-289

Lawrence, S., Giles C. L. and Bollacker K. D. (1998). CiteSeer: An Automatic Citation Indexing System. Digital Libraries 98 - Third ACM Conference on Digital Libraries, pp. 89-98.

Lawrence, S., Giles C. L., and Bollacker K. D. (1999). Distributed Error Correction. Fourth ACM Conference on Digital Libraries.

McCallum, A., Nigam, K., Rennie, J. and Seymore, K. (2000). Automating the Construction of Internet Portals with Machine Learning. Information Retrieval Journal Vol.3, pp. 127-163.

McCallum, A. and Peng, F. (2004). Accurate Information Extraction from Research Papers using Conditional Random Fields.

Nigam, K., Lafferty, J., McCallum, A. (1999). Using Maximum Entropy for Text Classification.

Oren Patashnik (1988) BibTeXing

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE **77,** pp. 257-286.

Ratnaparkhi, A. (1996). A Maximum Entropy Model for Part-Of-Speech Tagging. In Conference on Empirical Methods in Natural Language Processing, University of Pennsylvania, pp. 133-142.

Ratnaparkhi, A. (1998). Maximum Entropy Models for Natural Language Ambiguity Resolution.

Reynar, J. C. and Ratnaparkhi, A. (1997). A Maximum Entropy Approach to Identifying Sentence Boundaries.

Appendix A

A1 – Sample from Training dataset C'

<start>_<start> A_author%HEAD ._author%BODY Cau_author%BODY ,_author%BODY R_author%BODY ._author%BODY Kuiper_author%BODY ,_author%BODY and_author%BODY W_author%BODY ._author%BODY -_author%BODY P_author%BODY ._author%BODY de_author%BODY Roever_author%BODY ._author%TAIL Formalising_title%HEAD Dijkstra_title%BODY '_title%BODY s_title%BODY development_title%BODY strategy_title%BODY within_title%BODY Stark_title%BODY '_title%BODY s_title%BODY formalism_title%BODY ._title%TAIL In_editor%HEAD C_editor%BODY ._editor%BODY B_editor%BODY ._editor%BODY Jones_editor%BODY ,_editor%BODY R_editor%BODY ._editor%BODY C_editor%BODY ._editor%BODY Shaw_editor%BODY ,_editor%BODY and_editor%BODY T_editor%BODY ._editor%BODY Denvir_editor%BODY ,_editor%BODY editors_editor%BODY ,_editor%TAIL Proc_booktitle%HEAD ._booktitle%BODY 5th_booktitle%BODY ._booktitle%BODY BCS_booktitle%BODY -_booktitle%BODY FACS_booktitle%BODY Refinement_booktitle%BODY Workshop_booktitle%BODY ,_booktitle%TAIL 1992_date%HEAD ._date%TAIL <end>_<end>

<start>_<start> M_author%HEAD ._author%BODY Kitsuregawa_author%BODY ,_author%BODY H_author%BODY ._author%BODY Tanaka_author%BODY ,_author%BODY and_author%BODY T_author%BODY ._author%BODY Moto_author%BODY -_author%BODY oka_author%BODY ._author%TAIL Application_title%HEAD of_title%BODY hash_title%BODY to_title%BODY data_title%BODY base_title%BODY machine_title%BODY and_title%BODY its_title%BODY architecture_title%BODY ._title%TAIL New_journal%HEAD Generation_journal%BODY Computing_journal%BODY ,_journal%TAIL 1_volume%HEAD (_volume%BODY 1_volume%BODY)_volume%BODY ,_volume%TAIL 1983_date%HEAD ._date%TAIL <end>_<end>

<start>_<start> Alexander_author%HEAD Vrchoticky_author%BODY ._author%TAIL Modula_title%HEAD /_title%BODY R_title%BODY language_title%BODY definition_title%BODY ._title%TAIL Technical_tech%HEAD Report_tech%BODY TU_tech%BODY Wien_tech%BODY rr_tech%BODY -_tech%BODY 02_tech%BODY -_tech%BODY 92_tech%BODY ,_tech%BODY version_tech%BODY 2_tech%BODY ._tech%BODY 0_tech%BODY ,_tech%TAIL Dept_institution%HEAD ._institution%BODY for_institution%BODY Real_institution%BODY -_institution%BODY Time_institution%BODY Systems_institution%BODY ,_institution%BODY Technical_institution%BODY University_institution%BODY of_institution%BODY Vienna_institution%BODY ,_institution%TAIL May_date%HEAD 1993_date%BODY ._date%TAIL <end>_<end>

A2 – Legend for Fields

author	**title**	editor	*booktitle*	date
journal	volume	tech	institution	pages
location	publisher	note		

Field	**Color/Font**
author	red
title	bold
editor	yellow
booktitle	black italics
date	gray
journal	silver
volume	blue
tech	olive
institution	maroon
pages	black
location	green
publisher	brown
note	orange

Project Information

Project Type	:	Honours Year Project
Project Area	:	Software Systems
Project Title	:	Citation Parsing Using Maximum Entropy and Repairs
Project No	:	H79040
Student’s Name	:	Ng Yong Kiat
Project Advisor	:	Assistant Professor Kan Min-Yen
Date of Completion	:	November 2004
Deliverables	:	Report 1 Volume
Implementation Software and Hardware	:	DELL PC Intel Pentium 4 CPU 1.60GHz with 256 MB RAM, Windows XP, Perl, HTML, Java, MAXENT Package, pdftotext from Xpdf

Abstract

This thesis presents ParsCit, a system which parses citations from a publication or article, and label parts of citations with their corresponding field names. As citation styles differ between disciplines, publishers and authors, the task of citation parsing is difficult and inherently ambiguous. Whereas traditional citation parsers use a manually compiled set of rules to perform parsing, ParsCit adopts the framework of machine learning, learning rules for parsing from annotated data. A maximum entropy framework is employed to create a basic citation parse, and a series of repairs is performed to improve system accuracy. Given as few as 500 training examples, ParsCit is able to achieve a token accuracy of 94.20%, comparable to related machine learning approaches which lack the use of features applicable to the context of citation parsing. In this paper, I shall also emphasize the efficiency of the system’s set of repairs, which is absent in citation parsers seen to date.

Subject Descriptors:

I.2.6 Learning

I.2.7 Natural Language Processing

Keywords:

Text processing, Maximum Entity, Citation Parser