Locating and Parsing Bibliographical References in HTML Medical Articles

Jie Zou*, Daniel Le, George R. Thoma
Lister Hill National Center for Biomedical Communications, National Library of Medicine
8600 Rockville Pike, Bethesda, MD 20894

ABSTRACT

Bibliographical references that appear in journal articles can provide valuable hints for subsequent information extraction. We describe our statistical machine learning algorithms for locating and parsing such references from HTML medical journal articles. *Reference locating* identifies the reference sections and then decomposes them into individual references. We formulate reference locating as a two-class classification problem based on text and geometric features. An evaluation conducted on 500 articles from 100 journals achieves near perfect precision and recall rates for locating references. *Reference parsing* is to identify components, e.g. author, article title, journal title etc., from each individual reference. We implement and compare two reference parsing algorithms. One relies on sequence statistics and trains a Conditional Random Field. The other focuses on local feature statistics and trains a Support Vector Machine to classify each individual word, and then a search algorithm systematically corrects low confidence labels if the label sequence violates a set of predefined rules. The overall performance of these two reference parsing algorithms is about the same: above 99% accuracy at the word level, and over 97% accuracy at the chunk level.

Keywords: Reference Parsing, HTML Document Analysis, Document Object Model (DOM), Support Vector Machine (SVM), Conditional Random Field (CRF).

1. INTRODUCTION

Automatic metadata extraction from medical journals is key to the affordable creation of citations in MEDLINE®, the flagship database of the U.S. National Library of Medicine (NLM), containing over 17 million records and searched over 3 million times per day worldwide. Analyzing references, which are citations usually placed at the end of scientific publications, is an important preprocessing step for generating several MEDLINE bibliographic data items, e.g., identifying Comment-On/Comment-In articles (commentary article pairs)[11], assigning MeSH (Medical Subject Heading) indexing terms[1] through analyzing the MeSH terms already assigned to the cited articles, and many others.

With a rapidly increasing number of articles published online in the HTML format, we concentrate on analyzing references in such articles. It is a two-step process:

- *Locate references*: to identify reference sections, and then decompose them into individual references.
- *Parse references*: to extract entities from the references. Our goal is to extract 7 entities: *Citation Number* (<*N*>), *Author Names* (<*A*>), *Article Title* (<*T*>), *Journal Title* (<*J*>), *Volume* (<*V*>), *Pagination* (<*P*>) and *Publication Year* (<*Y*>). All remaining words are labeled as *Unknown* (<*U*>). [The notation inside each parenthesis is the abbreviated entity label.]

The most straightforward method for locating references is to use HTML tags. However, the HTML syntax is overly flexible, and is designed for displaying and manipulating, rather than for semantic understanding of, the HTML pages. Consequently, references can be implemented by completely different HTML codes. Using predefined HTML tags for reference locating can therefore produce misleading results[24].

We observe the following with regard to bibliographical references: (1) They contain distinctive text, e.g., author names, abbreviated journal names, pagination, publication years, etc.; (2) They have similar geometric features, e.g., occurring at end of the article, having similar width and height, etc.; (3) All references are consecutive neighbors, and adjacent ones are separated by a line-break.

*jzou@mail.nlm.nih.gov; phone 1 301 435-3148; fax 1 301 402-0341;

We therefore formulate reference location as a two-class classification. After rendering the HTML article in a Browser, geometric and text features are extracted from the zones in the HTML article, and an SVM classifier is used to classify these zones as either *reference zones* or *non-reference zones*. The third observation in the previous paragraph is a useful constraint which can expedite the process and increase its reliability.

Parsing references is also challenging, because NLM indexes over 5,200 journals from hundreds of publishers, who follow many different citation formats. Table 1 is a partial list of reference styles. While only shorter references are shown in the table for brevity, their lengths vary from less than 10 to more than 100 words.

Table 1: Examples of reference styles collected from MEDLINE-indexed medical articles.

(a)	2. M.F. Perutz, Nature of haem-haem interaction, Nature 237 (1972), pp. 495–499. Full Text via CrossRef | Abstract + References in Scopus | Cited By in Scopus <N>2.</N><A>M.F. Perutz,</A><T>Nature of haem-haem interaction,</T> <J> Nature</J><V>237</V><Y>(1972)</Y> <P> pp. 495–499</P>. <U> Full Text via CrossRef | Abstract + References in Scopus | Cited By in Scopus </U>
(b)	Cao et al., 2002a X. Cao, C. Tang and Y. Luo, Effect of nerve growth factor on neuronal apoptosis after spinal cord injury in rats, Chin. J. Traumatol. 5 (2002), pp. 131–5. Abstract + References in Scopus | Cited By in Scopus <N> Cao et al., 2002a </N><A>X. Cao, C. Tang and Y. Luo,</A> <T> Effect of nerve growth factor on neuronal apoptosis after spinal cord injury in rats,</T> <J> Chin. J. Traumatol. </J><V>5</V> <Y>(2002)</Y> <P> pp. 131–5.</P><U> Abstract + References in Scopus | Cited By in Scopus </U>
(c)	Saha, S., et al. (2002) Using the transcriptome to annotate the genome. Nat. Biotechnol, 20, 508–512[CrossRef][ISI][Medline]. <A>Saha, S., et al.</A> <Y>(2002)</Y> <T> Using the transcriptome to annotate the genome.</T> <J> Nat. Biotechnol,</J> <V>20</V> <P>508–512</P> <U>[CrossRef][ISI][Medline]</U>
(d)	Paddock, C. D., and J. E. Childs. 2003. Ehrlichia chaffeensis: a prototypical emerging pathogen. Clin. Microbiol. Rev. 16:37-64.[Abstract/Free Full Text] <A> Paddock, C. D., and J. E. Childs.</A> <Y>2003.</Y> <T> Ehrlichia chaffeensis: a prototypical emerging pathogen.</T> <J>Clin. Microbiol. Rev.</J> <V>16</V>: <P>37-64</P>. <U>[Abstract/Free Full Text]</U>
(e)	Wagner, A. F., Frey, M., Neugebauer, F. A., Schäfer, W., and Knappe, J. (1992) Proc. Natl Acad. Sci. U. S. A. 89, 996–1000[Abstract/Free Full Text] <A>Wagner, A. F., Frey, M., Neugebauer, F. A., Schäfer, W., and Knappe, J.</A> <Y>(1992)</Y> <J> Proc. Natl Acad. Sci. U. S. A.</J> <V>89</V>, <P>996–1000</P><U>[Abstract/Free Full Text]</U>
(f)	23. Ytrehus K, Liu Y, Downey J M. Am J Physiol. 1994;266:H1145–H1152. [PubMed] [Full Text] <N>23.</N><A> Ytrehus K, Liu Y, Downey J M.</A><J>Am J Physiol.</J><Y>1994</Y>;<V>226</V>:<P> H1145–H1152</P> <U>[PubMed] [Free Full Text]</U>
(g)	25. M. Huse, J. Kuriyan, Cell 109, 275 (2002). [CrossRef] [ISI] [Medline] <N>25.</N><A> M. Huse, J. Kuriyan,</A><J> Cell </J><V>109</V><P>275</P><Y>(2002)</Y> <U>[CrossRef] [ISI] [Medline]</U>
(h)	Roe, BA.; Crabtree, JS.; Khan, AS. DNA Isolation and Sequencing. Hoboken: John Wiley and Sons; 1996. <A>Roe, BA.; Crabtree, JS.; Khan, AS.</A><T>DNA Isolation and Sequencing</T><U>Hoboken: John Wiley and Sons</U><Y>1996</Y>
(i)	1. Tjaden P, Thoennes N. Full Report of the Prevalence, Incidence and Consequences of Violence Against Women: Research Report. Washington, DC: National Institute of Justice; 2000. NCJ 183781. <N>1.</N><A> Tjaden P, Thoennes N</A><T> Full Report of the Prevalence, Incidence and Consequences of Violence Against Women: Research Report.</T><U> Washington, DC: National Institute of Justice;</U><Y>2000</Y><U> NCJ 183781</U>
(j)	Lindell, T.J. (1980) Inhibitors of mammalian RNA polymerases In P.S., Sarin and R.C., Gallo (Eds.). Inhibitors of DNA and RNA Polymerases, Oxford Pergamon Press pp. 111–141. <A>Lindell, T.J.</A><Y>(1980)</Y><T> Inhibitors of mammalian RNA polymerases </T><U> In P.S., Sarin and R.C., Gallo (Eds.). </U><J> Inhibitors of DNA and RNA Polymerases </J><U> Oxford Pergamon Press </U><P> pp. 111–141</P>
(k)	25 Collaborative Computational Project Number 4, The CCP4 suite: programs for protein crystallography, Acta Crystallog. sect. D 50 (1994), pp. 760–763. <N>25</N><A> Collaborative Computational Project Number 4</A><T> The CCP4 suite: programs for protein crystallography </T><J> Acta Crystallog. sect. D </J><V>50</V><Y>(1994)</Y><P>pp. 760–763</P>

Shown in each part of the table are the original references with HTML tags removed, and labeled in an XML-like format. These references vary considerably in style. For example, (a) and (b) have Citation Numbers, but in completely different formats. Some other references, on the other hand, do not have Citation Numbers. There are also many different formats for Author Names: initials followed by last names, e.g., (a); last names followed by initials, e.g., (e); not all authors listed, e.g., (c); and the first author and the remaining authors in different formats, e.g., (d). In most cases,

the Article Title exists, but, sometimes not, as in (e). Most Journal Titles are significantly abbreviated, while some are not. Publication Years may or may not be inside a parenthesis. Pagination may be in the full format, e.g., 495-499 in (a), abbreviated format, e.g., 131-5 in (b) or only indicate the starting page, e.g., 275 in (g). They may be preceded by "pp.", "p.", or nothing. They can also contain non-digits, e.g., H1145-H1152 in (f). There are also several different volume-page combinations. The order in which the eight entities appear may also vary. Most references are citations to journal papers, but also to books, e.g., (h), reports, e.g., (i) and edited book chapters, e.g., (j). Occasionally, the authors may be organizations, e.g., (k). There are also many minor variations in the use of commas, spaces, semicolons or periods to separate different entities; in capitalizing all title words or just the first one; and so on.

We have implemented and compared two reference parsing algorithms, each based on a state-of-the-art machine learning technique. One focuses on sequence modeling and uses the Conditional Random Field (CRF), a statistical sequence model, to model the word sequence of a reference.

The other algorithm involves local word classification and is a two-step process. The first step is a multi-class (in our case, 8-class) classification. We call this *single-word classification*, since each word in the reference is assigned an entity label. We concentrate on examining local features of each individual word. These local features include the attributes of the word itself and its adjacent neighbors.

In addition, we find rules that always hold, in spite of the many styles and variations in the references. For example:

- Citation Number (<N>) is always the first entity, if it exists.
- "pp." or "p.", if labeled as pagination, has to be followed by at least another pagination word.

The complete set of such rules is listed in Section 4.2. These rules are useful global constraints with which the label sequence must comply. In the second step of the algorithm, labels with low confidence are systematically corrected if the entire label sequence violates the global rules.

This paper is organized as follows: in Section 2, we review existing methods for both locating and parsing references. We also briefly discuss the rationale and novelty of our approach. We discuss our methods in detail in Sections 3 and 4. Experimental evaluation is presented in Sections 5 and 6, and conclusions and future work constitute Section 7.

2. RELATED WORK

CiteSeer is a well-known and successful citation indexing system developed at NEC Research Institute[13]. CiteSeer uses Web search engines and heuristics to crawl the Web and download PDF and PostScript articles. After converting to text, CiteSeer uses heuristics to locate the reference section, and then parses each reference to extract fields such as title, author, year of publication, and so on. Similar systems include ISI Web of Knowledge[26] and Google Scholar[27]. We focus on HTML articles. By rendering the HTML articles in a Web browser (e.g. Microsoft Internet Explorer), geometric information (locations and sizes of zones) can be extracted, and these are important features for reliably locating bibliographical references.

There does not appear to be work reported on specifically locating references appearing in HTML articles. A related problem, which has been carefully studied recently by several researchers, is *mining data records from Web pages*. Data records are a list of similarly structured items, e.g., a list of products on sale. Liu et al. exploit the Web page structure and mostly depend on string matching of HTML tag sequences to detect data records[14]. Zhai and Liu extended this work, and used visual information and tree matching to detect data records, and then designed a partial tree alignment algorithm to align data records, and extract information from each one[23]. Reis et al. assumed that certain Web page groups share a common format and layout characteristics, and designed a tree matching algorithm to extract news content from news pages[19].

These data record mining algorithms have been used to extract consumer product reviews, news, Internet forum postings, and several other applications. These algorithms are mostly based on HTML DOM (Document Object Model) tree and HTML tags. The duplication of similar DOM tree structures is the primary cue for locating and aligning data records and for extracting information from them. In our reference locating problem, the text is a much more reliable feature compared to HTML tags. We therefore formulate the reference locating as a two-class classification based on geometric and text features.

Reference parsing, on the other hand, has received far more attention. Existing reference parsing methods can be generally divided into two categories: *rule based* methods and those based on *machine learning*. Rule based methods usually rely on a set of rules based on a domain expert's observation. Chowdhury[4] and Ding et al. [7] have used *template mining* techniques. Templates are manually crafted to summarize the recognizable patterns formed by either the data and/or text surrounding the data. A set of rules is usually associated with the templates, and when text matches to the templates, the data are extracted according to the rules. Day et al. [5, 6] extended the template mining approach, and used INFOMAP, a hierarchical framework, for knowledge (template) representation. Huang et al. used a gene sequence alignment tool, BLAST (Basic Local Alignment Search Tool), to extract citation metadata[10].

Journal publishers usually require authors to strictly follow predefined citation styles, and careful editorial checking and correction are usually conducted before publishing. Therefore, for a small set of journals, rule-based methods can be very successful. On the other hand, rule-based methods require domain experts to design the rules and maintain them over time. This approach also prevents adaptability and it is difficult to tune the system due to the rigidity of the rules. As mentioned, for MEDLINE data, over 5,200 journals from hundreds of publishers need to be processed. Hence, automatic reference parsing through rule-based methods poses a challenge due to the large variation of citation styles.

In contrast, machine learning approaches exhibit good adaptability by automatically learning the knowledge from training samples, and have therefore attracted a great deal of interest. Parmentier and Belaïd developed a *concept network* to hierarchically represent and recognize structured data from bibliographic citations[16]. Besagni et al. took a bottom-up approach based on Part-of-Speech (PoS) tagging[2]. In this approach, basic tags, which are easily recognized, are first grouped into homogeneous classes. Confusing tokens are then classified by either a set of PoS correction rules or a structure model generated from correctly detected records.

Hidden Markov Model (HMM), a successful machine learning tool for information extraction from sequences, has also been studied for parsing references, e.g., Takasu applied HMM for parsing erroneous references[22]. Conditional Random Field, another popular sequence model, is recently reported to achieve better performance compared to HMM[18]. We have therefore included CRF as one of our reference parsing methods.

Another frequently adopted machine learning method for information extraction is the Support Vector Machine (SVM) classifier. Han et al. took a two-stage approach for metadata extraction from the header part of research papers[9]. Okada et al. combined SVM and HMM for bibliographic component extraction [17]. We have implemented a reference parsing algorithm, which uses the SVM to classify each individual word. Intuitively, adjacent words in a reference usually are more likely to belong to the same entity. To exploit this important local dependency, we use not only the features extracted from the word itself, but also those extracted from its neighbors.

3. REFERENCE LOCATING

Our method begins by rendering the HTML article in Internet Explorer, and then creating an HTML DOM tree. DOM tree is a well-defined model published by W3C (World Wide Web Consortium) for accessing and manipulating HTML documents. However, DOM tree usually over-segments the HTML article. Figure 1 illustrates the HTML codes of two consecutive references, their rendering results and their corresponding DOM sub-trees. Following the DOM convention, we use <> to indicate element nodes and use # to indicate text nodes. Two references, shown in Figure 1(b), are simple text lines, but correspond to complicated DOM sub-trees, shown in Figure 1(c). (Dashed bounding boxes indicate zone sub-trees and will be explained below.) HTML DOM tree is the starting point for our reference locating algorithm, but a preprocessing step is required for pruning those unnecessary sub-trees, such as all DOM sub-trees in the lower two bounding boxes.

All HTML tags can be divided into two categories. *Inline tags* are those that do not introduce line breaks. A complete inline tag list in our algorithm includes: <A>, <ACRONYM>, <ABBR>, <B>, <BIG>, <CITE>, <CODE>, <DEL>, <DFN>, <EM>, <FONT>, <I>, <IMG>, <INPUT>, <INS>, <NOBR>, <KBD>, <Q>, <SAMP>, <SMALL>, <SPAN>, <STRONG>, <SUP>, <SUB>, <TT>, <U>, <VAR>. *Line-break tags* are the remaining tags, which do introduce line breaks, e.g., <P>, <TABLE>, <DIV>, <H1>, etc.

We merge all consecutive inline DOM nodes. This generates another tree structure that we call a *zone tree*. Each zone node contains either a set of consecutive inline DOM nodes, or one line-break node. Examples are shown in Figure 1(c). Dashed bounding boxes correspond to zone nodes. Two child zones are formed due to the line-break
 nodes. Their

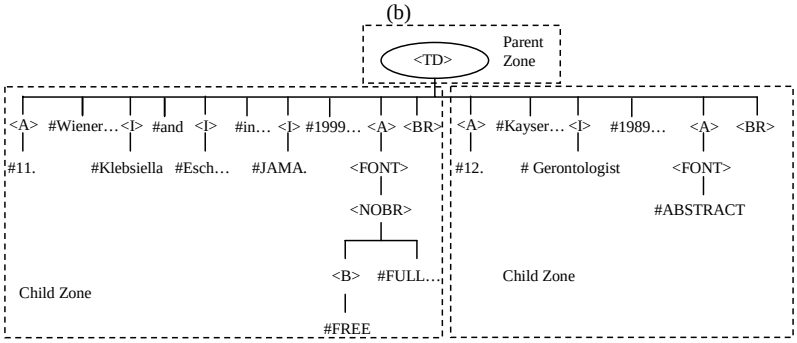
<A name="#REF-JOC60062-11">11.</A> Wiener J, Quinn JP, Bradford PA, et al. Multiple antibiotic-resistant <I>Klebsiella</I> and <I>Escherichia coli</I> in nursing homes. <I>JAMA.</I> 1999;281:517-523. <A href="/cgi/ijlink?linkType=ABST&journalCode=jama&resid =281/6/517"><FONT face="verdana, arial, helvetica, sans-serif" size=1><NOBR><B>FREE</B> FULL TEXT</NOBR></FONT></A>
<A name="#REF-JOC60062-12">12.</A> Kayser-Jones JS, Wiener CL, Barbaccia JC. Factors contributing to the hospitalization of nursing home residents. <I>Gerontologist.</I> 1989;29:502-510. <A href="/cgi/ijlink?linkType=ABST&journalCode=thegeron&resid=29/4/502"><FONT face="verdana, arial, helvetica, sans-serif" size=1>ABSTRACT</FONT></A>

(a)

11. Wiener J, Quinn JP, Bradford PA, et al. Multiple antibiotic-resistant *Klebsiella* and *Escherichia coli* in nursing homes. *JAMA.* 1999;281:517-523. FREE FULL TEXT

12. Kayser-Jones JS, Wiener CL, Barbaccia JC. Factors contributing to the hospitalization of nursing home residents. *Gerontologist.* 1989;29:502-510. ABSTRACT

(b)



(c)

Figure 1: Two reference examples. (a): HTML code; (b): Displayed in the Browser; (c): DOM sub-tree and zone sub-tree (marked with dashed bounding boxes).

parent is a zone corresponding to the <TD> DOM node. After this step, the non-break text lines are usually formed into one zone. The following process is conducted on the zone tree.

For each zone node, containing non-space text, 59 geometry and text features are extracted. The first 9 features with brief explanations are listed in the first 9 rows of Table 2. The remaining 50 features are binary features which indicate whether the specified words appear in the text. The 50 words are selected by the GSS measure[8].

Table 2: 59 features for reference locating.

Features	Comments
1. left	The left position of the zone bounding box normalized by the page width
2. top	The top position of the zone bounding box normalized by the page height
3. width	The width of the zone bounding box normalized by the page width
4. # of words	The number of words in the zone
5. 4-digit year pattern	Does the zone contain a word in four digit year pattern, e.g., 2005? The four digit year pattern must not be later than the current year.
6. pagination pattern	Does the zone contain a word in pagination pattern, e.g., 200-5, H100-H105?
7. # of name words	The number of name words in the zone. Our name dictionary contains 236,748 names, which are collected from 10 years of MEDLINE data
8. # of single-upper-case-letter words	The number of single-upper-case-letter words, e.g., D, in the zone.
9. # of double-upper-case-letter words	The number of double-upper-case-letter words, e.g., DL, in the zone.
10~59. Special word features	Does the zone contain the following words: j, crossref, abstract, full, text, medline, free, pp, via, scopus, cited, biol, amp, isi, infotrieve, microbiol, order, chem, mol, sci, no, res, proc, acad, biochem, natl, appl, al, et, acta, bacteriol, amino, environ, nature, summaryplus, links, rev, escherichia, biochemistry, vol, med, sect, crystallogr, immunol, biophys, crystallog, nat, clin, immun, nucleic.

GSS is named after the three authors who proposed the method. In a survey of text categorization by Sebastiani[20], the GSS measure is recognized as one of the best methods for selecting informative words. In our two-class classification, we define a joint GSS measure for each word t_k to be:

$$GSS(t_k) = \left| P(t_k, c_1)P(\bar{t}_k, c_0) - P(t_k, c_0)P(\bar{t}_k, c_1) \right|,$$

where, $P(\bar{t}_k, c_i)$ indicates the probability that, given a random zone, word t_k does not occur in the zone and that the zone belongs to category c_i. The GSS measure reflects the intuition that the best words are the ones distributed most differently in the reference and non-reference zones. $P(t_k, c_i)$ and $P(\bar{t}_k, c_i)$ are estimated by counting occurrences in the training samples, and the top 50 words with the highest GSS measures are selected and listed in the last row of Table 2. The words are listed in descending order of their GSS values. Because our training samples are medical articles, many of

the selected words are abbreviated journal titles. For locating references in general publications, the most informative word list can be easily created by following the same procedure. Also note that special words like "crossref", "medline", "scopus" and "infotrieve" are also highly ranked. They are usually placed at the end of the references to provide quick access to external links. Intuitively, they are informative words for detecting references.

We used LibSVM[3], an SVM library developed at the National Taiwan University, to implement our reference zone classification. We adopted Radial Basis Function (RBF) as the kernel function, and the values for two parameters, C (penalty parameter of the errors) and γ (RBF parameter), were selected through exhaustive grid-search using cross-validation on training samples.

This SVM classifier assigns each zone tree node a probability value for being a reference zone*. Because references are consecutive neighbors, they must be consecutive siblings in the zone tree. We use the following 3-step heuristic to label the reference zones: (1) We find a parent zone node, which has the most reference-like children (probability of being reference zone is larger than 0.5); (2) Under this parent, we search for the best locations of the first and the last reference zones: $[t_F^*, t_L^*] = \arg\max_{t_F, t_L} \prod_{t_F \le i \le t_L} P(c_i = R) \prod_{0 \le j < t_F, t_L < j \le N} (1 - P(c_j = R))$, where, t_F and t_L are the locations of the first and the last references, N is the total number of children zones, $P(c_k = R)$ is the probability of the k[th] child to be a reference zone. Since there are usually at most about 100 children zones under a parent, we simply use exhaustive search to find t_F^* and t_L^*; (3) We label all consecutive sibling zones between t_F^* and t_L^* reference zones.

4. REFERENCE PARSING

For the step following reference locating, we have implemented two reference parsing algorithms. One relies on sequence statistics and trains a Conditional Random Field (CRF) sequence model. The other focuses on local feature statistics and trains a Support Vector Machine (SVM) to classify each individual word, followed by a search algorithm that systematically corrects low confidence labels if the label sequence violates a set of predefined rules. We describe these in the sub-sections below, and compare them in Section 6.

Table 3: 14 binary features extracted from individual words.

Feature	Comments
1.Author Name Feature	Is the word in Author Name dictionary?
2. Article Title Feature	Is the word in Article Title dictionary?
3. Journal Title Feature	Is the word in Journal Title dictionary?
4. Pagination Pattern	Is the word in pagination format, e.g., 200-5, H100-H105?
5. Name Initial Pattern	Is the word in name initial pattern, e.g., J.Z., J.-Z?
6.Four Digit Year Pattern	Is the word in four digit year pattern, e.g., 2005? It must not be later than the current year.
7. et, al	Is the word "et" or "al", or "et.", or "al."?
8. pp., p.	Is the word "pp.", or "p.", or "pp", or "p"?
9. Ended With "."	Does the word end with "."?
10. Upper Case First Char	Is the first character of the word upper case?
11. Letter Only	Does the word contain letters only?
12. Digit Only	Does the word contain digits only?
13. Digit and Letter	Does the word contain both digits and letters?
14. Digit and Letter Only	Does the word contain digits and letters only?

4.1 CRF for reference parsing

CRF is a probabilistic model designed for labeling sequence data[12, 21]. It is defined as the conditional probability of a state sequence, $s = \{s_1, s_2, \cdots, s_N\}$, given an input observation sequence $o = \{o_1, o_2, \cdots, o_N\}$: $p(s \mid o) \propto \exp\left(\sum_{t=1}^{N} F(s, o, t)\right)$, where N is the length of the sequence, and $F(s, o, t)$ is the sum of CRF feature functions at position t. There are two types of CRF feature functions: edge feature functions, $f_i(\cdot)$, that characterize state-state transitions and state feature functions,

* In our implementation, many zone nodes are assigned a zero probability without going through the SVM classification in order to save computation time. These zones that are ignored contain less than 5 words or more than 400 words. Because articles typically have at least 2 references, we also ignore zone nodes which have no siblings.

$g_j(.)$, that characterize the observation-state relations. We use first-order Markov chain in our CRF model and the observations are extracted from the word itself and its immediate left and right neighbors. Therefore, our CRF feature functions can be written as: $F(s,o,t) = \sum_i \lambda_i f_i(s_{t-1}, s_t) + \sum_j \lambda_j g_j(o_{t-1}, o_t, o_{t+1}, s_t)$. The goal of training a CRF is to estimate the parameters λ_i and λ_j, i.e., the weights of feature functions. The trained CRF model can then be used to assign labels to unknown sequences.

We collected word dictionaries for Author Names, Article Titles and Journal Titles from 10 years of MEDLINE historical data. There are a total of 236,748 Author Name words, 108,484 Article Title words and 6,909 Journal Title words. The observation vector o_t at position t contains not only the word itself, but also 14 other binary features as well. The first 3 features of a word, Author Name Feature, Article Title Feature and Journal Title Feature, are binary features indicating whether the word is in the corresponding dictionaries. We also extract an additional 11 binary features. All 14 binary features and their brief explanations are listed in Table 3. We used MALLET[15], a machine learning library for language processing, developed by McCallum, to implement our CRF reference parsing algorithm.

4.2 Combining SVM and global rules for reference parsing

In our second algorithm, we treat reference parsing as a multi-class classification of each individual word. A set of local features is extracted from each word and its adjacent neighbors. An SVM classifier is trained on a set of manually-labeled references, and then applied to classify each word of a test reference. If the label sequence after single word classification violates a set of predefined rules, a search algorithm is used to find a label sequence, which obeys the global constraints and has the highest probability.

Single word classification using SVM

From each word, 15 features are extracted. The first 14 are the same binary features listed in Table 3. The 15th feature is the normalized position, i.e., the position of the word normalized by the total number of the words in the reference.

Intuitively, we expect adjacent words in a reference to usually have a higher probability of belonging to the same entity. In order to utilize these local contextual dependencies, the features used for the classification are extracted from not only the word itself, but also from its neighbors.

As done for reference locating, we adopted LibSVM with RBF kernel function for this single word classification. Similarly, the two parameters, C (penalty parameter of the errors) and γ (RBF parameter), were also selected through exhaustive grid-search using cross-validation on training samples.

Global rules for references

By inspection, we have found that the following rules always hold for references.

- "J", "J.", or "Journal" cannot be labeled as an isolated single Journal Title entity. At lease one of its adjacent neighbors must also be part of the Journal Title.
- "pp." or "p.", if labeled as pagination, has to be followed by at least another pagination word.
- Except for Unknown Entity, each of the other entities can only be composed of consecutive words, and appear at most once in the reference.
- There must be an Author entity.
- A Citation Number must be the first entity if it exists.
- Author entity must appear before Article Title and Journal Title, if they exist.
- Article Title entity must appear before Journal Title, if they exist.
- Journal Title must appear before Volume and Pagination, if they exist.
- Volume must appear before Pagination, if they exist.

These global rules are very strong and useful constraints, but some of them characterize long distance (high order) correlations, and therefore are difficult to model with statistical models. We choose to explicitly check whether the label sequences obey the rules.

A search algorithm for finding optimal label sequence which complies with the rules

Due to the high accuracy of single word classification, most references can already be correctly parsed. For those that do not pass the global rule test, nearly all of them are close to the correct label sequence with only a few words mislabeled. The goal is then to identify and correct those mislabeled words. We present a systematic search algorithm guaranteed to find a label sequence that is valid (obeys the global rules) and is most-likely (has the highest probability).

Given an N word reference, $\{w_1, w_2, \cdots w_N\}$, and M (in our case, $M = 8$) entity labels, $\{c^1, c^2, \cdots c^M\}$, single word classification calculates an $M \times N$ probability matrix **P**. An element of **P**, $p(c^j \mid w_i)$, represents the posterior probability of word w_i belonging to entity c^j. To avoid computational overflow, log-probability, $l(c^j \mid w_i) = \ln p(c^j \mid w_i)$, is used in the following discussions.

The log-probability of a label sequence, $L = \{c_1, c_2, \cdots, c_N\}$, where, $c_i \in \{c^1, c^2, \cdots c^M\}$ can then be calculated as: $l(L) = \sum_{i=1}^{N} l(c_i \mid w_i)$. The cost of changing a word's label in the sequence can also be calculated as: $Cost(c_i \to c_i' \mid w_i) = l(c_i \mid w_i) - l(c_i' \mid w_i)$. The cost of changing labels of K words, $K \le N$ in a label sequence is then: $Cost(L \to L' \mid w_1, w_2, \cdots, w_K) = \sum_{k=1}^{K} Cost(c_k \to c_k' \mid w_k)$.

The key to finding the most-likely and valid label sequence is then to search possible *label sequence modifications* in the ascending order of their costs. The search stops at the first label sequence, which obeys the global rules. Because there are $M^N - 1$ possible modifications, it is computationally prohibitive to calculate costs for all possible modifications and then sort them. We present an algorithm which enumerates sequence modifications in ascending order of their costs.

We first calculate the costs for all $N(M-1)$ possible *single-token modifications* (only one word's label is modified) and sort them in ascending order. This is not computationally expensive. We arrange these $N(M-1)$ single-token modifications in the middle line of Figure 2 (marked with a dashed bounding box) in ascending order of their costs. <1> indicates the single-token modification with the minimum cost, and so on. It is easy to see that the first and second sequence modifications must be the first two single-token modifications. In each subsequent column we list all possible *multi-token modifications*, which are all possible combinations of the previous single-token modification and all other previous single- and multi- token modifications. For example, in Column 3, the previous single-token modification is <2>, and there is only one other modification, i.e., <1>, so there is only one multi-token modification, i.e., <2,1>. Let us assume that <1> and <2> are the modifications to the same word, so the modification <2,1> is meaningless. We mark it with a dashed circle and abandon it. In Column 4, the previous single-token modification is <3>, and all other possible previous modifications are <1> and <2>, so we have two multi-token modifications, as shown in Column 4, <3,1> and <3,2>. Let us assume the cost of <3,1> is less than that of <4>, but the cost of <3,2> is greater than that of <4>, and therefore, we place <3,1> on top of <4> and <3,2> below <4>. Similarly, we create Columns 5, 6, and so on. In this example, <1>, <2>, <5> are assumed to be single-token modifications of the same word, and <3> and <4> are single-

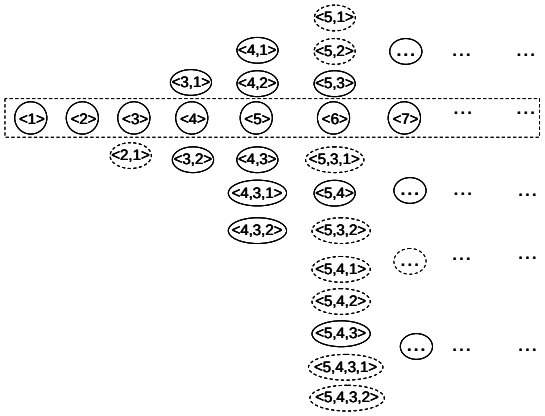


Figure 2: Illustration of the algorithm for searching for valid and most-likely reference label sequence.

token modifications of the other two words. Meaningless multi-token modifications are marked with dashed circles.

For each column, let us call the modifications above the single-token modification the *upper column*, and the modifications below the single-token modification the *lower column*. Although the modifications in each column are ordered, the modifications in the lower column may have higher cost than the modifications in the later columns. However, a key observation is that the modifications in an upper column must be smaller than those in the lower column and the later columns. This is the key for creating new columns dynamically and enumerating all modifications in ascending order of their costs. The algorithm is shown below:

1. Calculate costs for all $N(M-1)$ single-token modifications, and sort them in ascending order.
2. Test the first single-token modification. If it obeys the rules, go to the end, otherwise continue.
3. Test the second single-token modification. If it obeys the rules, go to the end, otherwise continue.
4. Create Column 3, and save all modifications into an ascending ordered list.
5. Repeat for K=3, 4,…, N(M-1)-1:
    a. Repeat:
        i. Pop up and test the first modification from the ordered list.
        ii. If it obeys the rules, go to the end, otherwise continue.
    b. Until single-word modification <K> is tested.
    c. Create Column K+1, and save the modifications into the ordered list.
6. Finish testing remaining ordered list.
7. End

It is clear that the algorithm is still an exhaustive search, but it searches from the label sequence generated by single word classification, which, in our case, is close to the correct solution. Most searches, therefore, terminate very quickly. Because the search is conducted in the ascending order of costs, it is guaranteed to find the most-likely modification that obeys the rules. In an actual implementation, it is of course better to set a limit on the maximum number of modifications to be tested to avoid lengthy computation. In our implementation, the search terminates after 10,000 modifications have been tested. In practical systems, if the search does not terminate when the limit is reached, this is an indication that the parsing may not be accurate.

5. EVALUATION OF REFERENCE LOCATING

To evaluate reference locating, we randomly collected 1,000 articles from the top 100 journals cited in the MEDLINE 2006 database (the articles from the most important 100 journals indexed by MEDLINE in 2006). We randomly select 500 of these articles as training samples, and the remaining 500 as test samples.

In the 500 training samples there are 21,709 references. On the other hand, there are significantly more non-reference zones. Because the SVM classifier is known to be biased toward the class label with more training samples[25], we retain only the same number of the randomly selected non-reference zones. A total of 43,418 zones therefore are used to find the 50 most informative words using the GSS measure, and to train an SVM classifier for reference zone classification.

Our reference locating method is very reliable. There are a total of 22,147 reference zones in those 500 test articles. The algorithm achieves very high precision and recall rates, producing only 6 false positives and 2 false negatives.

6. EVALUATION OF REFERENCE PARSING

To evaluate reference parsing, we randomly collected and manually labeled 2,400 references. 600 of them are randomly selected from the 500 training articles as training samples, and the remaining 1,800 are the test samples randomly selected from the 500 test articles. We evaluate the algorithm performance at two levels. One is at the word level, i.e., the labeling accuracy of individual words. The other is at a chunk or component level, i.e., the percentage of the entity chunks[†] correctly identified.

[†] An entity chunk consists of a set of consecutive words that share the same entity label. For example, the reference in Table 1(a) contains 8 entity chunks, where the first is the Number chunk consisting of a single word "2", and the second is the Author chunk consisting of two words, "M.F." and "Perutz".

6.1 CRF-based method

We conducted an evaluation of our CRF-based parsing algorithm by varying the number of training sequences using 10, 25, 50, 100, 300 and all 600 sequences. For 10, 25 and 50 sequences, the experiments were repeated 5 times, and for 100 and 300 sequences, the experiments were repeated 3 times. The results are shown in Table 4. There are 53,622 words in the 1800 test references, and the accuracy reported in Table 4 is the overall accuracy for all 8 entities. Higher accuracy is indeed achieved with more training samples. Table 5 shows the accuracy at chunk level for each entity with all 600 training sequences.

Table 4: Word level accuracy of CRF-parsing

Training Samples	10	25	50	100	300	600
Accuracy(%)	95.92	97.09	97.96	98.51	98.72	99.04

Table 5: Chunk level accuracy of CRF-parsing with 600 training sequences

	Number	Author	Title	Journal	Volume	Year	Pagination	Unknown	Overall
Total	627	1800	1308	1758	1735	1791	1751	1708	12478
Correct	622	1753	1211	1692	1720	1778	1731	1640	12147
Accuracy	99.2%	97.4%	92.6%	96.2%	99.1%	99.3%	98.9%	96.0%	97.3%

6.2 Combining SVM and global rule correction

Evaluation of single word classification

For the second approach, i.e., combining SVM and global rule correction, we first conducted a comprehensive evaluation of the single word classification by varying the number of training samples and the number of words from which the features are extracted. Following the same experimental protocol, we tested with 10, 25, 50, 100, 300, and all 600 training sequences. To vary the number of words from which the features are extracted, we tested with the word itself (15 features), the word and two adjacent neighbors (the immediate left and right words, giving 45 features), and the word and four adjacent neighbors (the immediate two left and two right words, amounting to 75 features). The experimental results are shown in the third column of Table 6.

Table 6: Word level accuracy of single word classification and after global rule correction

Samples	# of words	Accuracy of single word classification	Accuracy after global rule correction
10	The word itself, 15 features	89.91%	92.67%
	The word and 2 adjacent neighbors, 45 features	95.28%	96.67%
	The word and 4 adjacent neighbors, 75 features	95.79%	96.97%
25	The word itself, 15 features	91.65%	94.44%
	The word and 2 adjacent neighbors, 45 features	96.68%	97.57%
	The word and 4 adjacent neighbors, 75 features	97.27%	97.68%
50	The word itself, 15 features	92.32%	94.71%
	The word and 2 adjacent neighbors, 45 features	97.58%	98.36%
	The word and 4 adjacent neighbors, 75 features	98.17%	98.51%
100	The word itself, 15 features	92.98%	95.12%
	The word and 2 adjacent neighbors, 45 features	98.00%	98.55%
	The word and 4 adjacent neighbors, 75 features	98.51%	98.80%
300	The word itself, 15 features	93.36%	95.63%
	The word and 2 adjacent neighbors, 45 features	98.45%	98.88%
	The word and 4 adjacent neighbors, 75 features	98.91%	99.06%
600	The word itself, 15 features	93.35%	95.39%
	The word and 2 adjacent neighbors, 45 features	98.63%	98.98%
	The word and 4 adjacent neighbors, 75 features	99.07%	99.13%

Evaluation of global rule correction

All the experiments above are continued with the global rule correction algorithm described in Section 4.2, and the accuracies are reported in the fourth column of Table 6. We find that accuracies increase after the global rule correction. For chunk level evaluation, we conducted an experiment with all 600 training sequences and with 45 features. The chunk level accuracy of each entity is reported in Table 7.

Table 7: Chunk level accuracy of SVM-parsing with 600 training sequences

	Number	Author	Title	Journal	Volume	Year	Pagination	Unknown	Overall
Total	627	1800	1308	1758	1735	1791	1751	1708	12478
Correct	621	1757	1198	1686	1727	1788	1731	1640	12148
Accuracy	99.0%	97.6%	91.6%	95.9%	99.5%	99.8%	98.9%	96.0%	97.4%

6.3 Discussion

We summarize the following observations from the evaluations we have conducted on the two reference parsing algorithms.

First of all, there are strong local contextual dependencies among reference words and they must be utilized in reference parsing algorithms. This has been clearly demonstrated by the single word classification experiments. Regardless of the number of training samples, the accuracies are significantly improved if combining the features extracted from the immediate left and right neighbors (45 features). Combining features from an additional two adjacent neighbors (75 features), on the other hand, achieves only slight accuracy improvements. This is in agreement with many studies of statistical sequence models, where usually only the first-order correlation is modeled, and the first-order Markov Chain is the underlying graphic model.

Global rule correction is effective. We believe that the global rule correction is a good practical heuristic to correct minor errors. When it fails, it also serves as a good indicator for low confidence parsing.

The article title contains the most heterogeneous text, and therefore is the most difficult entity to extract. Both CRF-parsing and SVM-parsing achieve the lowest accuracy in Title chunk identification. On the other hand, both algorithms achieve high accuracy (around 99%) for entities having distinctive features, such as Number, Volume, Year and Pagination.

Comparing Tables 5 and 7, when training with 600 references, CRF-parsing and SVM-parsing essentially achieve the same overall performance: about 99% accuracy at word level and above 97% accuracy at chunk level. SVM-parsing missed only 3 Publication Years. SVM is a sophisticated classifier, which is expected to achieve better performance on entities having distinctive features. On the other hand, CRF achieves 1% higher accuracy on Title chunk identification. Titles contain heterogeneous text, i.e., having indistinctive features. It is likely that CRF, by modeling the entire sequence, has better chance to label them correctly. The performance may be further improved, if the advantages of SVM (sophisticated local classifier) and CRF (powerful sequence model) can be combined.

Most references in our collection are citations to journal papers (Examples (a)~(g) and (k) in Table 1). There are few errors in this kind of "standard" references; even organizational authors (Examples (k) in Table 1) can usually be successfully labeled. Only a very small percentage of references are citations to reports and books (Examples (h)~(j) in Table 1), and our current algorithm finds it difficult to label their Unknown (<U>) entities. For the edited books especially (Examples (j) in Table 1), the long word sequence of the editors sometimes confuses the algorithms. Further research is warranted to solve this problem.

7. CONCLUSIONS

We have presented approaches for locating and parsing references in HTML medical journal articles. We formulate reference locating as a two-class classification, and have demonstrated that text and geometry are very reliable for locating references, and an SVM classifier based on these features can achieve near 100% accuracy.

The first order correlation between reference words is important contextual information, and must be used in reference parsing algorithms. We implemented and compared two reference parsing algorithms. CRF-parsing focuses on modeling the word sequence with Conditional Random Fields, and SVM-parsing concentrates on local single word classification. The overall performance of these two approaches is about the same: above 97% accuracy at chunk level.

8. ACKNOWLEDGEMENT

We thank Loc Tran for collecting journal articles and Dr. Jong Woo Kim for collecting MEDLINE historic data for author names and article titles. This research was supported by the Intramural Research Program of the National Institutes of Health (NIH), National Library of Medicine, and Lister Hill National Center for Biomedical Communications.

REFERENCES

1. A.R. Aronson, O. Bodenreider, H.F. Chang, S.M. Humphrey, J.G. Mork, S.J. Nelson, T.C. Rindflesch, W.J. Wilbur, "The NLM indexing initiative," *Proc AMIA Symp.* pp. 17-21, 2000.
2. D. Besagni, A. Belaïd and N. Benet, "A segmentation method for bibliographic references by contextual tagging of fields," *Proc. ICDAR*, vol. 1, pp. 384-388, 2003.
3. C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
4. G. Chowdhury, "Template mining for information extraction from digital documents," *Library Trends*, vol. 48 no. 1, pp. 182-208, 1999.
5. M.-Y. Day, T.-H. Tsai, C.-L. Sung, C.-W. Lee, S.-H. Wu, C.-S. Ong, W.-L. Hsu, "A knowledge-based approach to citation extraction," *IEEE Int'l Conf. Information Reuse and Integration*, pp. 50-55, 2005.
6. M.-Y. Day, R.T.-H. Tsai, C.-L. Sung, C.-C. Hsieh, C.-W. Lee, S.-H. Wu, K.-P. Wu, C.-S. Ong, W.-L. Hsu, "Reference metadata extraction using a hierarchical knowledge representation framework," *Decision Support Systems*, vol. 43, no.1, pp. 152-167, 2007.
7. Y. Ding, G. Chowdhury and S. Foo, "Template mining for the extraction of citation from digital documents," *Proc. the 2nd Asian Digital Library Conference*, pp. 47-62, 1999.
8. L. Galavotti, F. Sebastiani and M. Simi, "Experiments on the use of feature selection and negative evidence in automated text categorization," *Proc. ECDL*, pp. 59-68, 2000.
9. H. Han, C.L. Giles, E. Manavoglu, H. Zha, Z. Zhang and E.A. Fox, "Automatic document metadata extraction using support vector machines," *Proc. Joint Conference on Digital libraries*, pp. 37-48, 2003
10. I.-A. Huang, J.-M. Ho, H.-Y. Kao, W.-C. Lin, "Extracting citation metadata from online publication lists using BLAST," *Proc. of the Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 26-28, 2004.
11. I. Kim, D. Le, G. R. Thoma, "Identification of "comment-on sentences" in online biomedical documents using support vector machines," *Proc. SPIE conference on Document Recognition and Retrieval*, pp.68150X(1-9), 2007.
12. J. Lafferty, A. McCallum and F. Pereira, "Conditional random fields: probabilistic models for segmenting and 13. labeling sequence data," *Proc. ICML*, pp. 282-289, 2001.
13. S. Lawrence, C. L. Giles and K. Bollacker, "Digital Libraries and Autonomous Citation Indexing," *IEEE Computer*, vol. 32, no. 6, pp. 67-71, 1999.
14. B. Liu, R. Grossman, Y. Zhai, "Mining Web pages for data records," *IEEE Intelligent Systems*, vol. 19, no. 6, pp. 49-55, 2004.
15. A.K. McCallum, *MALLET: A Machine Learning for Language Toolkit*, http://mallet.cs.umass.edu. 2002.
16. F. Parmentier and A. Belaïd, "Logical structure recognition of scientific bibliographic references," *Proc. ICDAR*, pp. 1072-1076, 1997.
17. T. Okada, A. Takasu and J. Adachi, "Bibliographic component extraction using support vector machines and hidden Markov models," *Proc. ECDL*, pp. 501-512, 2004.
18. F. Peng and A. McCallum, "Accurate information extraction from research papers using conditional random fields," *Proc. of Human Language Technology Conference*, pp. 329-336, 2004.
19. D.C. Reis, P.B. Golgher, A. S. Silva, A. F. Laender, "Automatic Web news extraction using tree edit distance," *Proc. WWW*, pp. 502-511, 2004.
20. F. Sebastiani, "Machine learning in automated text categorization", *ACM Computing Surveys*, vol. 34, no. 1, pp. 1-47, 2002.
21. C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," book chapter in *Introduction to Statistical Relational Learning*. Edited by L. Getoor and B. Taskar. MIT Press. 2006.
22. A. Takasu, "Bibliographic attribute extraction from erroneous references based on a statistical model," *Proc. JCDL*, pp. 49-60, 2003.
23. Y. Zhai, B. Liu, "Structure data extraction from the Web based on partial tree alignment," *IEEE Tran. Knowledge and Data Engineering*, vol. 18, no. 12, pp. 1614-1628, 2006.
24. J. Zou, D. Le, G.R. Thoma, "Structure and Content Analysis for HTML Medical Articles: A Hidden Markov Model Approach," *Proc. DocEng*, pp. 119-201, 2007.
25. J. Zou, D. Le, G.R. Thoma, "Extracting a sparsely-located named entity from online HTML medical articles using support vector machine," *SPIE Proc. Document Recognition and Retrieval (SPIE-DR&R)*, pp. 68150P(1-10), 2008.
26. http://www.isiwebofknowledge.com/
27. http://scholar.google.com/