Published in final edited form as:
*Int J Doc Anal Recognit*. 2010 June 1; 13(2): 107–119. doi:10.1007/s10032-009-0105-9.

# Locating and parsing bibliographic references in HTML medical articles

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

## Abstract

The set of references that typically appear toward the end of journal articles is sometimes, though not always, a field in bibliographic (citation) databases. But even if references do not constitute such a field, they can be useful as a preprocessing step in the automated extraction of other bibliographic data from articles, as well as in computer-assisted indexing of articles. Automation in data extraction and indexing to minimize human labor is key to the affordable creation and maintenance of large bibliographic databases. Extracting the components of references, such as author names, article title, journal name, publication date and other entities, is therefore a valuable and sometimes necessary task. This paper describes a two-step process using statistical machine learning algorithms, to first locate the references in HTML medical articles and then to parse them. *Reference locating* identifies the reference section in an article and then decomposes it into individual references. We formulate this step as a two-class classification problem based on text and geometric features. An evaluation conducted on 500 articles drawn from 100 medical journals achieves near-perfect precision and recall rates for locating references. *Reference parsing* identifies the components of each reference. For this second step, we implement and compare two 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, followed by a search algorithm that 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**

HTML document analysis; Document Object Model (DOM); Reference parsing; Support Vector Machine (SVM); Conditional Random Field (CRF)

## 1 Introduction

The automatic extraction of bibliographic data from medical journal articles 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 world-wide. The references that typically appear at the end of such articles provide valuable information not only for generating bibliographic data items, such as Comment-On/Comment-In articles (commentary article pairs) [19], but also for systems that

jzou@mail.nlm.nih.gov .

index articles by automatically assigning Medical Subject Headings to them [1], as well as many other applications.

Hence, the analysis of these references is an important preprocessing step. Our method to accomplish this is a two-step process:

1. *Locate references*: to identify the reference section in an article, and then decompose the section into individual references.
2. *Parse references*: to extract entities from individual references. Our goal is to extract the following 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 in the reference are labeled as *Other* (<O>). Most of these "Other" words are those such as "crossref", "medline", or "scopus" placed at the end of the references to provide quick access to external links. Other typical words in this category include editors, publishers, city names and institute names, which cannot be readily categorized into the above-mentioned 7 entities. Table 1 illustrates some typical "Other" words, which are marked with <O> tag.

Several well-known citation-indexing systems, e.g., CiteSeer [23], ISI Web of Knowledge [41] and Google Scholar [42], implement algorithms for locating and parsing references. These systems usually use Web search engines to crawl the Web and download PDF and PostScript articles. After converting these to text, they first locate the reference section and then parse each reference to extract fields such as title, author and year of publication.

In contrast, our focus is on HTML-formatted medical journal articles, which differ from PDF/PS-converted text files in several ways. One problem with HTML-formatted texts is that visually similar pages can be implemented with completely different HTML codes. We therefore choose to use statistical machine learning approaches, rather than relying on HTML-tag-based heuristic rules.

While the most straightforward method for locating references in HTML journal articles is to use HTML tags, the HTML syntax is overly flexible and is designed for displaying and manipulating, rather than to semantically understand, the HTML pages. Consequently, references in these pages can be implemented by completely different HTML codes, leading to incorrect results when using predefined HTML tags for reference locating [39].

First, we observe the following with regard to bibliographic references:

1. They contain distinctive text, e.g., author names, abbreviated journal names, pagination, publication year;
2. They have similar geometric features, e.g., they occur at the end of the article, have similar width and height;
3. All references are consecutive neighbors, adjacent ones being separated by a line break.

These observations suggest formulating reference location as a two-class classification. The procedure would be the following: after rendering the HTML article in a browser, segment the pages into zones, extract geometric and text features from each zone, and use an SVM classifier to classify each zone as either a *reference* or a *non-reference zone*.The third observation listed earlier is a useful constraint that can expedite the process and increase its reliability.

Formulating a procedure for parsing references is challenging because of the variety of formats appearing in the 5,200 journals indexed by NLM. Table 1 is a partial list of these 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.

Shown in each part of the table is a reference as it appears in an article with HTML tags removed, and this reference labeled in an XML-like format. We see that these references vary considerably in style, and many variations are listed in this paragraph. For example, (a) has a Citation Number, but reference (b) is identified by a combination of first author and publication year. Some other references, on the other hand, have neither a Citation Number nor any indication of sequence. 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 following different formats, e.g., (d). In most cases, the Article Title exists, but, sometimes not, as in (e). Most Journal Titles are abbreviated, while some are not. Publication Year may or may not be inside a parenthesis. Pagination may be in the full format, e.g., 495–499 in (a), an 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 by nothing. Page numbers may also contain non-digits, e.g., H1145-H1152 in (f). There are also several different volume–page combinations. The eight entities of interest to us (listed earlier) may vary in the order they appear. Most references cite journal papers, but variations in citations appear for books, e.g., (h), reports, e.g., (i) and edited book chapters, e.g., (j). Occasionally, the “authors” may be organizations, e.g., (k). In addition to all these stylistic differences, there are many minor variations in the use of commas, spaces, semicolons or periods to separate different entities. Some references have all words in the Article Title capitalized and others just the first word; and so on.

These variations pose challenges to the accurate parsing of the references. For this purpose, we have implemented and compared two algorithms, each based on a state-of-the-art machine learning technique. One uses the Conditional Random Field (CRF), a statistical sequence model, to model the word sequence of a reference.

The other involves local word classification and is itself a two-step process. The first step is a multi-class (in our case, 8-class) classification, which assigns an entity label to each word in the reference. We examine local features of each word, including the attributes of the word itself and those of its adjacent neighbors.

In addition, there are rules that always hold, regardless of the many styles and variations. For example:

- Citation Number (<N>) is always the first entity, if it exists.
- “pp.” or “p.”, if it appears and is labeled as pagination, has to be followed by at least one other pagination word (usually the actual page numbers).

The complete set of such rules is listed in Sect. 4.2.2. These rules prove to be useful global constraints with which the label sequence must comply. In the second step of the algorithm, labels exhibiting low confidence are systematically corrected if the entire label sequence violates these global rules.

This paper is organized as follows: in Sect. 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 Sects. 3 and 4. Experimental evaluation is presented in Sects. 5 and 6 and a summary is given in Sect. 7.

## 2 Related work

### 2.1 Related work on reference locating

Existing algorithms for HTML page understanding are typically designed in a straightforward way, i.e., they depend heavily on the HTML tags. For example, Buyukkokten et al. and Kaasinen et al. use the <P>, <TABLE> and <UL> tags to divide Web pages for subsequent conversion and summarization [4,18]. Diao et al. use four types of tags, including <P>, <TABLE>, <LI>/<UL> and <H1>~<H6>, to detect paragraph, table, list and headings, respectively [11].

For specifically locating references in HTML articles, there does not appear to be any reported work. A related problem however, which has been 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, mostly depending on string matching HTML-tag sequences to detect data records [25]. Zhai and Liu extended this work, using visual information and tree matching to detect data records, and then a partial tree alignment algorithm to align data records, and extract information from each one [38]. Reis et al. assumed that certain groups of Web pages share common format and layout characteristics and designed a tree-matching algorithm to extract content from news pages [34].

These data record–mining algorithms have been used to extract consumer product reviews, news, Internet forum postings and several other applications. These algorithms are also mostly based on the HTML DOM (Document Object Model) tree and HTML tags. The occurrence of similar DOM tree structures in the Web page is the primary cue for locating and aligning data records and for extracting information from them.

On the other hand, to extract data from scanned documents, geometric features are by far the most important. Scanned document layout analysis, which includes geometric and logical layout analyses, has been extensively documented in the literature. Geometric layout analysis, as its name suggests, concentrates on analyzing document images based on their geometric features. Most of these algorithms follow either a top-down or a bottom-up approach. Top-down algorithms recursively divide a whole page into smaller zones. The process terminates when certain criteria are met. Typical top-down methods include the X-Y cut [14,27], shape-directed-covers-based algorithms [2] and several others. Bottom-up algorithms start with the image pixels, cluster them into connected components, then into words, lines and finally zones. Typical bottom-up methods include Docstrum [29], Block Adjacency Graph (BAG) [17]. Hybrid methods combining split and merge strategies have also been proposed in [15,32]. Logical layout analysis is used to analyze the logical components of the scanned document, though most algorithms also consider geometric features [20,21]. In other scanned document analysis algorithms designed for special purposes, such as name extraction [24], geometric features (sometimes also called visual cues), such as gaps between text zones, are also extensively utilized. A review of methods for scanned document image analysis is given in [28].

Our approach to locating references from HTML articles uses both text and geometric features. By rendering the HTML articles in a Web browser (e.g., Microsoft Internet Explorer), both text and geometric information can be extracted. The text features include orthographic and binary features indicating whether or not certain words appear. The geometric features extracted are the normalized locations and sizes of the zone bounding boxes. These features, listed in Table 2, are much more reliable cues compared to HTML tags, and we therefore formulate reference locating as a two-class classification based on these features.

### 2.2 Related work on reference parsing

In contrast to reference locating, reference parsing has received considerable attention in the literature. Existing approaches fall into two categories: *rule-based* methods and those based on *machine learning*. Rule-based methods rely on rules based on a domain expert's observation. For example, Chowdhury [6] and Ding et al. [12] have manually crafted templates 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 the text matches the templates, the data are extracted according to the rules. Day et al. [9,10] extended the template-aided 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 [16].

Journal publishers usually require authors to strictly follow predefined citation styles, and carefully edit submitted material before publication. Therefore, for a homogeneous set of journals, rule-based methods can be very successful. On the other hand, these methods require domain experts not only to handcraft the rules, but also maintain them over time. The rigidity of the rules prevents adaptability and makes it difficult to tune the system for a heterogeneous set of journals. In MEDLINE, citations are drawn from articles in over 5,200 journals from hundreds of publishers, leading to a large variation of citation styles. This poses a challenge to rule-based approaches for reference parsing.

In contrast, by automatically learning the knowledge from training samples, machine learning approaches exhibit good adaptability 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 [31]. Besagni et al. took a bottom-up approach based on Part-of-Speech (PoS) tagging [3]. In this approach, basic PoS 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.

Cortze et al. propose a knowledge-based approach for reference parsing, called FLUX-CiM [7]. This is an unsupervised approach based on a frequency-tuned lexicon and includes four stages: blocking, matching, binding and joining.

Hidden Markov Model (HMM) and Conditional Random Field (CRF), as successful machine learning tools for information extraction from sequences, have also been studied for parsing references. For example, Takasu applied HMM for parsing erroneous references [37], and Councill et al. used CRF to implement an open source reference-parsing package [8]. Since CRF has recently been reported to outperform HMM [33], we pick CRF as one of our reference-parsing methods.

Another frequently adopted machine learning method for information extraction is the Support Vector Machine (SVM) classifier. For example, Okada et al. combined SVM and HMM for bibliographic component extraction [30]. In one of our reference-parsing algorithms, therefore, we use the SVM to classify each individual word. We make the intuitive assumption that adjacent words in a reference are more likely than not to belong to the same entity. We exploit this important local dependency by using 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 (Document Object Model) 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 subtrees.

Following the DOM convention, we use <> to indicate element nodes and use # to indicate text nodes. The two references, as shown in Fig. 1a, are simple text lines, but correspond to complicated DOM sub-trees, shown in Fig. 1c. (Dashed bounding boxes indicate zone sub-trees and are explained below.) HTML DOM tree is the starting point for our reference locating algorithm, but a preprocessing step is required for pruning over-segmented sub-trees that are unnecessary, such as the DOM sub-trees in the lower two bounding boxes.

To explain our method, we first define two types of HTML tags:

*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>.

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 a single line-break node. Examples are shown in Fig. 1c, in which dashed bounding boxes correspond to zone nodes. Two child zones are formed due to the line-break <BR> nodes. Their parent is a zone corresponding to the <TD> DOM node. This is the only step in our algorithm that uses the HTML-tag information. After this step, the text lines without line breaks are usually formed into one zone. Subsequent steps of the algorithm are independent of HTML tags and are conducted on the zone tree.

From each zone node containing non-space text, 59 geometric 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 are binary features which indicate whether the specified words appear in the text. These 50 words are selected by the GSS measure [13].

GSS is named after the three authors who proposed a method for selecting informative words. In a survey of text categorization by Sebastiani [35], the GSS measure is recognized as one of the best methods for this purpose. Specifically for our two-class classification, we make a slight modification and define a joint GSS measure for each word $t_k$ to be:

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

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 medical 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 “cross-ref”, “medline”, “scopus” and “infotrieve” are also highly ranked. These are usually placed at the end of references to provide quick access to external links. Intuitively, they are informative words for detecting references.

We used LibSVM [5], 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 $\gamma$ (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 three-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 greater than 0.5);
2. Under this parent, we search for the best locations of the first and the last reference zones:

$$\left[t_F^*, t_L^*\right] = \arg \max_{t_F, t_L} \prod_{t_F \le i \le t_L} P(c_i = R) \times \prod_{0 \le j < t_F, t_L < j \le N} \left(1 - P\left(c_j = R\right)\right),$$

   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 no more than 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.

These steps are necessary for two reasons. One is that for some journals, the footnotes and references share the same list, with footnotes appended below the reference section in an article. We would like to remove those footnote siblings. The other much more serious reason is that the visual layout of an HTML page can be very different from the structure layout of the underlying HTML code. Even though some reference sections may visually appear to be stand-alone sections, in the actual HTML implementation, they may share the same DOM parent with non-reference zones. The steps given earlier are indispensable for handling these cases.

## 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 and label 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 following sub-sections, and compare them in Sect. 6.

### 4.1 CRF for reference parsing

CRF is a probabilistic model designed for labeling sequence data [22,36]. It is defined as the conditional probability of a state sequence, $s = \{s_1, s_2, \ldots, s_N\}$, given an input observation sequence $o = \{o_1, o_2, \ldots, o_N\} : p(s|o) \propto \exp\left(\Sigma_{t=1}^{N} F(s, o, t)\right)$, 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, $g_j(\cdot)$, 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 $\lambda_i$ and $\lambda_j$, i.e., the weights of feature functions. The trained CRF model can then be used to assign labels to unknown sequences.

In order to extract observation features, we generated word dictionaries for Author Names, Article Titles and Journal Titles from 10 years of MEDLINE data. There are a total of 236,748 Author Name words, 108,484 Article Title words and 6,909 Journal Title words. The observation feature 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 exists 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 [26], a machine learning JAVA library for language processing, developed by McCallum and colleagues, 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 in a reference using an SVM classifier. This classifier is trained on a set of manually labeled references, and then applied to test references in which it classifies (labels) every word. Our method follows this step by ensuring that the sequence of the class labels does not violate certain heuristic rules (Sect. 4.2.2) that are observed to always hold. If the label sequence violates these rules, a search algorithm is used to find a sequence which complies with them at the highest probability (Sect. 4.2.3).

**4.2.1 Single word classification using SVM—**The SVM uses 15 features from each word. The first 14 are the same ones 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 have a higher probability of belonging to the same entity. In order to exploit these local contextual dependencies, the features used for the classification are extracted not only from the word itself, but also from its neighbors.

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

**4.2.2 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 least 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 "Other" entity (defined in Sect. 1), the remaining entities can only consist of consecutive words, and appear at most once in the reference.
- A Citation Number must be the first entity, if it exists.
- Author entity must appear before Article Title and Journal Title, if these exist.
- Article Title entity must appear before Journal Title, if this exists.
- Journal Title must appear before Volume and Pagination, if they exist.
- Volume must appear before Pagination, if this exists.

These global rules are very strong and useful constraints, but most of them are difficult to model with statistical models. We therefore explicitly check whether the label sequences obey the rules with a search algorithm described below.

**4.2.3 A search algorithm for finding the 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 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, $\{w_1, w_2, \ldots, w_N\}$, and $M$ (in our case, $M = 8$) entity labels, $\{c^1, c^2, \ldots, c^M\}$, single word classification calculates an $M \times N$ probability matrix $\mathbf{P}$. An element of $\mathbf{P}$, $p(c^j | w_i)$, represents the posterior probability of word $w_i$ belonging to entity $c^j$, which is the output from the single word SVM classifier. To avoid computational overflow, log-probability, $l(c^j | w_i) = \ln p(c^j | w_i)$, is used in the following discussions.

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

The process of finding the most likely and valid label sequence then becomes a search for 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 Fig. 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>. Assuming that the cost of <2,1> is larger than the cost of <3>, we place it below <3> in the column. In Column 4, the previous single-token modification is <3>, and all other possible previous modifications include <1>, <2> and <2,1>, so we have three multi-token modifications. They are arranged according to their costs as shown in Column 4. In this case, the cost of <3,1> is smaller than that of <4>, and therefore it is placed above <4>. The costs of <3,2> and <3,2,1> are larger than that of <4>, and therefore they are placed below <4>. Let us assume that <1> and <3> are the modifications to the same word, so the modifications <3,1> and <3,2,1> are meaningless. We mark them with dashed circles and abandon them. Similarly, we create Columns 5, 6 and so on. In this example, <1>, <3>, <5> are assumed to be single-token modifications of the same word, and <2> and <4> are single-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 following columns. However, a key observation is that the modifications in an upper column must be smaller than those in the lower column and the following 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.
        iii. Stop when single word modification <K> is tested.
    b. 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 collected a random set of 1,000 articles from the top 100 journals cited in the MEDLINE 2006 database. We randomly selected 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 biased toward the class label with more training samples [40], we retain the same number of reference and non-reference zones in our training set for a total of 43,418 zones. These zones are used to find the 50 most informative words using the GSS measure to train the SVM classifier.

Our reference locating method is very reliable. From 22,147 reference zones in 500 test articles, the algorithm achieves near-perfect precision and recall rates, producing only 6 false positives and 2 false negatives.

## 6 Evaluation of reference parsing

To evaluate reference parsing, we manually labeled 2,400 references of which 600 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 level, i.e., the percentage of the entity chunks[1] correctly identified.

### 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, we randomly selected 5 different sets of training sequences and repeated the experiments 5 times. For 100 and 300 sequences, we randomly selected 3 different sets of training sequences and repeated the experiments 3 times. For 600 sequences, we have used all available training sequence, and can conduct the experiment only once. The averaged results of the repeated experiments 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.

### 6.2 Method based on SVM and global rule correction

**6.2.1 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

[1]An entity chunk consists of a set of consecutive words that share the same entity label. For example, the reference in Table 1a 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”.

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.

**6.2.2 Evaluation of global rule correction—**All the above-mentioned experiments are continued with the global rule correction algorithm described in Sect. 4.2.3, 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.

### 6.3 Evaluation on FLUX-CiM data

We conducted an evaluation with the 2000 health science references from the publicly available FLUX-CiM data set, using our method combining SVM classification and global rule correction. No retraining is conducted using FLUX-CiM data. All FLUX-CiM references are similar in style and have no Citation Number and Other entities. Results appear in Table 8. Besides the word and chunk-level accuracies, which can be compared to Tables 4, 5, 6 and 7, we also include chunk-level precision, recall and F-measure. These three measures have been used in Table 3 of [7] for the same FLUX-CiM dataset. We have included precision, recall and F-Measure reported in [7] in the last three rows of Table 8, for easy comparison.

Compared to the performance reported in [7], our method shows lower performance for the Author and Journal entity chunks,[2] but higher performance on the other four entity chunks. Compared to Tables 5 and 7, the chunk-level accuracy of Table 8 is lower. The following may explain the performance of our algorithm: (1) we classify 8 entities, while FLUX-CiM health science references have only 6 entities. The more classes, the more difficult the problem; (2) we do not retrain our algorithm with FLUX-CiM references. There are some noticeable ground-truth labeling discrepancies between our ground-truth labeling and FLUX-CiM ground-truth labeling. For example, in FLUX-CiM data, publishers (e.g., Macmillan Publishing Company) and sometimes addresses (e.g., New York) are labeled as Journal, but we label them as "Other". We have labeled 84 entities as "Other," which contributes to the errors.

### 6.4 Discussion

We summarize the following observations from our experiments. First of all, the strong local contextual dependencies among reference words should be exploited 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 features extracted from the immediate left and right neighbors are combined (45 features). Combining features from two additional adjacent neighbors (75 features), on the other hand, achieves only slight accuracy improvements, as shown consistently in each cell of Table 6. 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.

[2]The word-level accuracy of Author and Journal chunks is still high, which indicates that most of the chunk-level errors are due to mislabeling a few words in the chunk.

Secondly, we find that global rule correction is effective as shown in Table 6. We believe that global rule correction is a good practical heuristic to correct minor errors. When it fails, it also serves as a good indicator that the parsing may be incorrect, and requires operator attention.

The article title contains the most heterogeneous text, and therefore is the most difficult entity to extract accurately. Both CRF-parsing and SVM-parsing yield 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.

We see from a comparison of Tables 5 and 7 that both reference-parsing methods (CRF and SVM) 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., have varying features. It is possible that CRF, by modeling the entire sequence, performs better with such text. We anticipate that overall performance for all entities may be 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). Our methods make few errors for this kind of “standard” references; even organizational authors (Examples (k) in Table 1) can usually be successfully labeled. However, in our collection, a 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 Other (<O>) entities. For the edited books especially (Examples (j) in Table 1), the long word sequence naming the editors sometimes confuses the algorithms. Further research is needed to solve this problem.

## 7 Summary

We have presented approaches for locating and parsing references in HTML-formatted 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. An SVM classifier based on these features has achieved near 100% accuracy.

The first-order correlation between reference words is important contextual information and is 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.

Our algorithm has been applied to medical journal articles only. However, we expect that it could be easily applied to other domains by collecting a set of ground-truth samples and re-training the SVM and CRF models.

## Acknowledgments

We thank Loc Tran for collecting journal articles used for training and evaluation of our algorithms and Dr. Jong Woo Kim for generating the author name dictionary from MEDLINE historic data. We also thank Eli Cortez from Universidade Federal do Amazonas, Manaus, Brazil for sharing the FLUX-CiM data with us. In addition, we thank the two anonymous reviewers for their very constructive comments. This research was supported by the Intramural

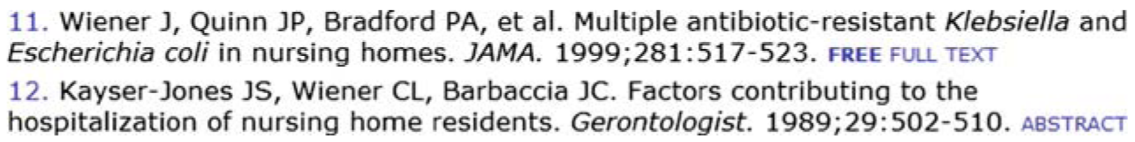
Research Program of the National Institutes of Health (NIH), National Library of Medicine and Lister Hill National Center for Biomedical Communications.

## References

1. Aronson, AR.; Bodenreider, O.; Chang, HF.; Humphrey, SM.; Mork, JG.; Nelson, SJ.; Rindflesch, TC.; Wilbur, WJ. The NLM indexing initiative; Proceedings of AMIA Symposium; 2000; p. 17-21.
2. Baird, HS.; Jones, SE.; Fortune, SJ. Image segmentation by shape-directed covers; Proceedings of International Conference Pattern Recognition; 1990; p. 820-825.
3. Besagni D, Belaïd A, Benet N. A segmentation method for bibliographic references by contextual tagging of fields. Proc. ICDAR 2003;1:384–388.
4. Buyukkokten, O.; Garcia-Molina, H.; Paepche, A. Accordion summarization for end-game browsing on PDAs and cellular phones; Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 2001; p. 213-220.
5. Chang, C-C.; Lin, C-J. LIBSVM: a library for support vector machines. 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
6. Chowdhury G. Template mining for information extraction from digital documents. Libr. Trends 1999;48(1):182–208.
7. Cortez E, da Silva AS, Goncalves MA, Mesquita F, de Moura ES. A flexible approach for extracting metadata from bibliographic citations. J. Am. Soc. Inf. Sci. Technol 2009;60(6):1144–1158.
8. Councill, IG.; Giles, CL.; Kan, M-Y. ParsCit: an open-source CRF reference string parsing package; Proceedings of the 6th International Language Resources and Evaluation; 2008;
9. Day, M-Y.; Tsai, T-H.; Sung, C-L.; Lee, C-W.; Wu, S-H.; Ong, C-S.; Hsu, W-L. A knowledge-based approach to citation extraction; IEEE International Conference Information Reuse and Integration; 2005; p. 50-55.
10. Day M-Y, Tsai RT-H, Sung C-L, Hsieh C-C, Lee C-W, Wu S-H, Wu K-P, Ong C-S, Hsu W-L. Reference metadata extraction using a hierarchical knowledge representation frame-work. Decis. Support Syst 2007;43(1):152–167.
11. Diao, Y.; Lu, H.; Chen, S.; Tian, Z. Toward learning based web query processing; Proceedings of International Conference on Very Large Databases; 2000; p. 317-328.
12. Ding, Y.; Chowdhury, G.; Foo, S. Template mining for the extraction of citation from digital documents; Proceedings of the 2nd Asian Digital Library Conference; 1999; p. 47-62.
13. Galavotti, L.; Sebastiani, F.; Simi, M. Experiments on the use of feature selection and negative evidence in automated text categorization; Proceedings of ECDL; 2000; p. 59-68.
14. Ha, J.; Haralick, R.; Phillips, I. Recursive X-Y cut using bounding boxes of connected components; Proceedings 3rd International Conference Document Analysis and Recognition; 1995; p. 952-955.
15. Hauser SE, Le DX, Thoma GR. Automated zone correction in bitmapped document images. Proc. SPIE: Document Recognit. Retr.VII 2000;3976:248–258.
16. Huang, I-A.; Ho, J-M.; Kao, H-Y.; Lin, W-C. Extracting citation metadata from online publication lists using BLAST; Proceedings of the 8th Pacific–Asia Conference on Knowledge Discovery and Data Mining; 2004; p. 26-28.
17. Jain AK, Yu B. Document representation and its application to page decomposition. IEEE Trans. Pattern Recognit. Mach. Intell 1998;20(3):294–308.
18. Kaasinen, E.; Aaltonen, M.; Kolari, J.; Melakoski, S.; Laakko, T. Two approaches to bringing internet services to WAP devices; Proceedings of the 9th International World Wide Web Conference; 2000; p. 231-246.
19. Kim, I.; Le, D.; Thoma, GR. Identification of “comment-on sentences” in online biomedical documents using support vector machines; Proceedings of the SPIE Conference on Document Recognition and Retrieval; 2007; p. X1-X9.
20. Kim, J.; Le, D.; Thoma, GR. Automatic labeling in document images; Proceedings of the SPIE Conference on Document Recognition and Retrieval; 2001; p. 111-122.
21. Klink S, Kieninger T. Rule-based document structure understanding with a fuzzy combination of layout and textual features. Int. J. Document Anal. Recognit 2001;4:18–26.

22. Lafferty, J.; McCallum, A.; Pereira, F. Conditional random fields: probabilistic models for segmenting and labeling sequence data; Proceedings of the ICML; 2001; p. 282-289.
23. Lawrence S, Giles CL, Bollacker K. Digital libraries and autonomous citation indexing. IEEE Comput 1999;32(6):67–71.
24. Likforman-Sulem L, Vaillant P, de Bodard A. Automatic name extraction from degraded document images. Pattern. Anal. Appl 2006;9(2):211–227.
25. Liu B, Grossman R, Zhai Y. Mining Web pages for data records. IEEE Intell. Syst 2004;19(6):49–55.
26. McCallum, AK. MALLET: a machine learning for language toolkit. 2002. http://mallet.cs.umass.edu
27. Nagy G, Seth S, Viswanathan M. A prototype document image analysis system for technical journals. Computer 1992;25:10–22.
28. Nagy G. Twenty years of document image analysis in PAMI. IEEE Trans. Pattern Anal. Mach. Intell 2000;22(1):38–62.
29. O'Gorman L. The document spectrum for page layout analysis. IEEE Trans. Pattern Recognit. Mach. Intell 1993;15:1162–1173.
30. Okada, T.; Takasu, A.; Adachi, J. Bibliographic component extraction using support vector machines and hidden Markov models; Proceedings of the ECDL; 2004; p. 501-512.
31. Parmentier, F.; Belaïd, A. Logical structure recognition of scientific bibliographic references; Proceedings of the ICDAR; 1997; p. 1072-1076.
32. Pavlidis T, Zhou J. Page segmentation and classification. Graph. Models Image Process 1992;54:484–496.
33. Peng, F.; McCallum, A. Accurate information extraction from research papers using conditional random fields; Proceedings of Human Language Technology Conference; 2004; p. 329-336.
34. Reis, DC.; Golgher, PB.; Silva, AS.; Laender, AF. Automatic web news extraction using tree edit distance; Proceedings of the WWW; 2004; p. 502-511.
35. Sebastiani F. Machine learning in automated text categorization. ACM Comput. Surv 2002;34(1):1–47.
36. Sutton, C.; McCallum, A. An introduction to conditional random fields for relational learning. In: Getoor, L.; Taskar, B., editors. Introduction to statistical relational learning. MIT Press; Cambridge: 2006.
37. Takasu, A. Bibliographic attribute extraction from erroneous references based on a statistical model; Proceedings of the JCDL; 2003; p. 49-60.
38. Zhai Y, Liu B. Structure data extraction from the Web based on partial tree alignment. IEEE Tran. Knowl. Data Eng 2006;18(12):1614–1628.
39. Zou, J.; Le, D.; Thoma, GR. Structure and content analysis for HTML medical articles: a hidden markov model approach; Proceedings of the DocEng; 2007; p. 119-201.
40. Zou J, Le D, Thoma GR. Extracting a sparsely-located named entity from online HTML medical articles using support vector machine. Proc. Document Recognit. Retr 2008;68150:P1–P10.
41. http://www.isiwebofknowledge.com/
42. http://scholar.google.com/

**(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)**

<A name="#REF-JOC60062-11">11.</A> Wiener J, Quinn JP, Bradford PA, et al. Mu ltiple 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&amp;journalCode=jama&amp;resid =281/6/517"><FONT face="verdana, arial, helvetica, sans-serif" size=1><NOBR><B>FREE</B> FULL TEXT</NOBR></FONT></A><BR>

<A name="#REF-JOC60062-12">12.</A> Kayser-Jones JS, Wiener CL, Barbacci a JC. Factors contributing to the hospitalization of nursing home residents. <I>Gerontologist.</I> 1989;29:502-510. <A href="/cgi/ijlink?linkType=ABST&amp;journalCode=thegeron&amp;resid=29/4/502"><FONT face="verdana, arial, helvetica, sans-serif" size=1>ABSTRACT</FONT></A><BR>

**(c)**

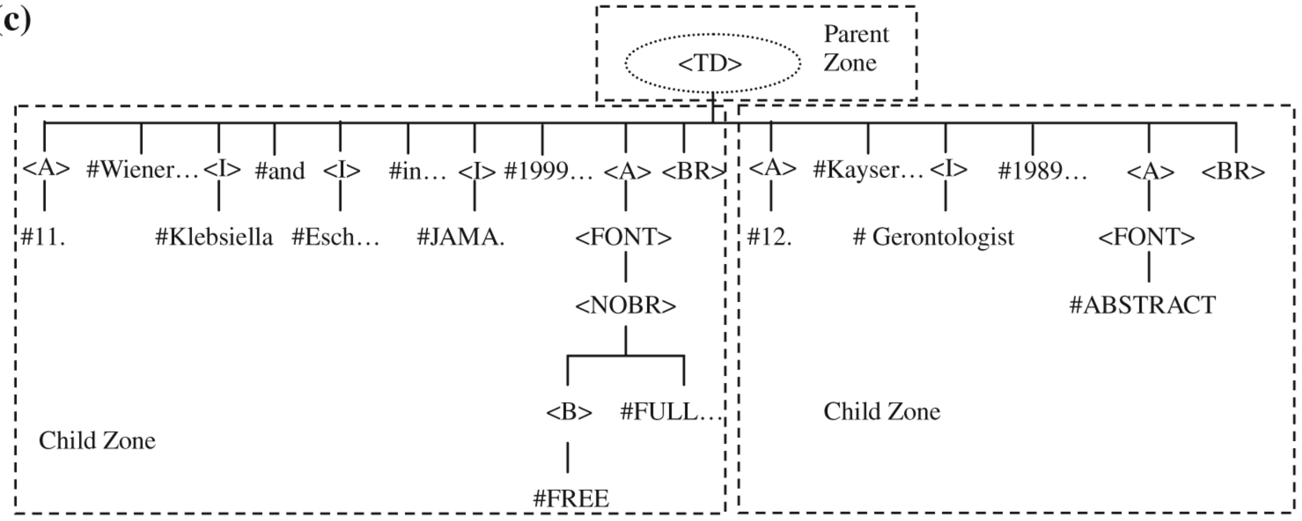


**Fig. 1.**

Two examples of references. **a** As displayed in the browser; **b** HTML code; **c** DOM sub-tree and zone sub-tree (marked with dashed bounding boxes). <TD> is the parent DOM node of the two references, but also of all references in the article. Since it is not included in the HTML code in (**b**), we use a dotted ellipse

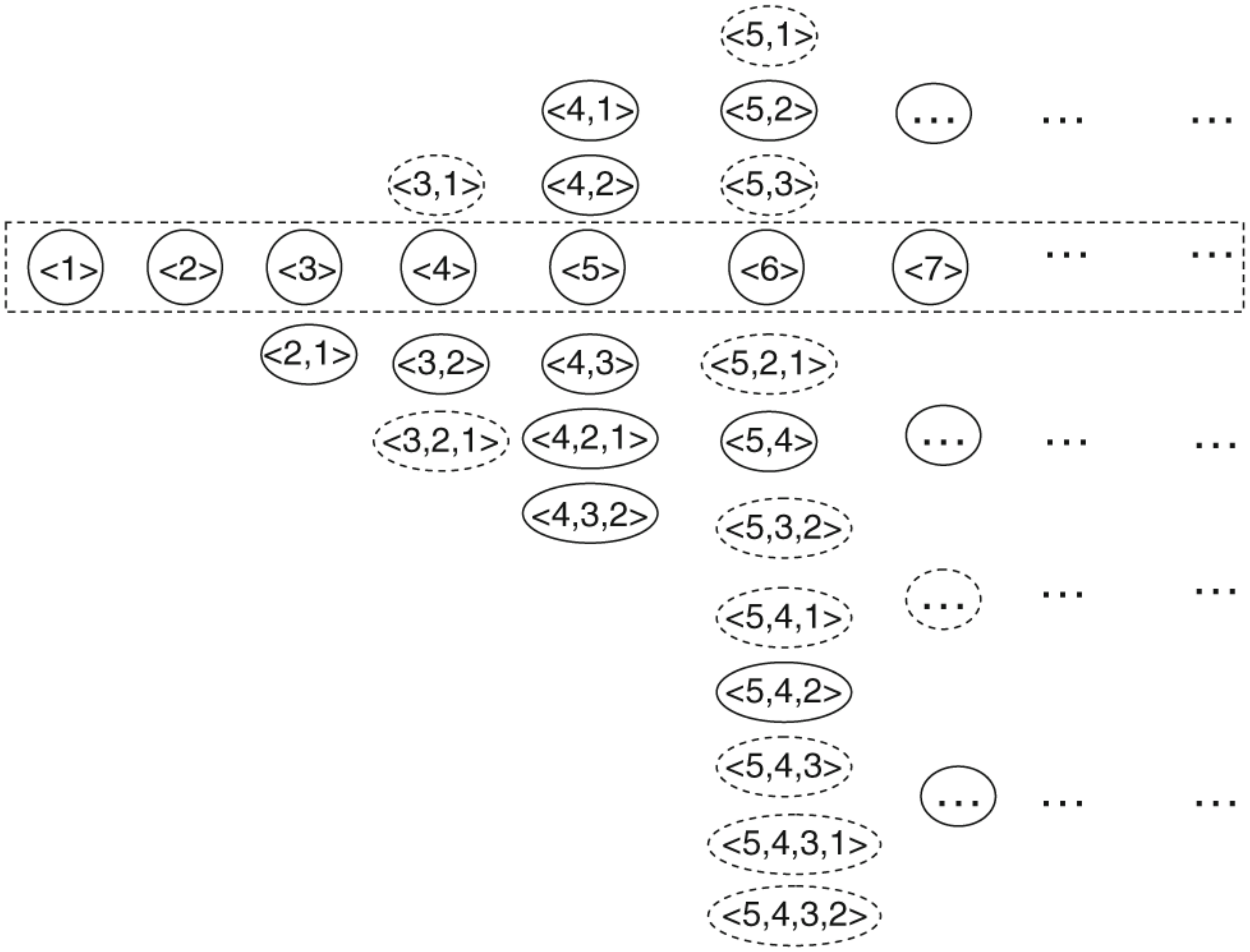


**Fig. 2.**
Illustration of the algorithm for searching for valid and most likely reference label sequence

**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>. <O> Full Text via CrossRef \| Abstract + References in Scopus \| Cited By in Scopus </O> |
| (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> <O> Abstract + References in Scopus \| Cited By in Scopus </O> |
| (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> <O> [CrossRef][ISI][Medline]</O> |
| (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>. <O>[Abstract/Free Full Text]</O> |
| (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> <O>[Abstract/Free Full Text]</O> |
| (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> <O>[PubMed] [Free Full Text]</O> |
| (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> <O>[CrossRef] [ISI] [Medline]</O> |
| (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> <O>Hoboken: John Wiley and Sons</O> <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> <O>Washington, DC: National Institute of Justice;</O> <Y>2000</Y> <O>NCJ 183781</O> |
| (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><O>In P.S., Sarin and R.C., Gallo (Eds.). </O><J>Inhibitors of DNA and RNA Polymerases </J> <O> Oxford Pergamon Press </O> <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>

**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 author 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., J, in the first reference of Fig. 1. |
| 9. # of double-upper-case-letter words | The number of double-upper-case-letter words, e.g., JS, in the second reference of Fig. 1. |
| 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, bio-chem, 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. |

**Table 3**

14 binary features extracted from individual words

| Feature | True if |
|---|---|
| 1.Author Name feature | the word is in Author Name dictionary. |
| 2. Article Title feature | the word is in Article Title dictionary. |
| 3. Journal Title feature | the word is in Journal Title dictionary. |
| 4. Pagination pattern | the word is in pagination format, e.g., 200-5, H100-H105. |
| 5. Name Initial pattern | the word is in name initial pattern, e.g., J.Z., J.-Z. |
| 6.Four digit year pattern | the word is in four-digit year pattern, e.g., 2005. It must not be later than the current year. |
| 7. et, al | the word is “et” or “al”, or “et.”, or “al.”. |
| 8. pp., p. | the word is “pp.”, or “p.” or “pp”, or “p”. |
| 9. Ended with “.” | the word ends with “.”. |
| 10. Upper case first char | the first character of the word is upper case. |
| 11. Letter only | the word contains letters only. |
| 12. Digit only | the word contains digits only. |
| 13. Digit and letter | the word contains both digits and letters. |
| 14. Digit and letter only | the word contain digits and letters only. |

**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

| | Cit. number | Author | Title | Journal | Volume | Year | Pagination | Other | 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 |

**Table 6**

Word-level accuracy of single word classification and after global rule correction

| Samples | Number 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 |

**Table 7**

Chunk-level accuracy of SVM-parsing (after global rule correction) with 600 training sequences

| | Cit. number | Author | Title | Journal | Volume | Year | Pagination | Other | 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 |

**Table 8**

Accuracy of SVM-parsing (after global rule correction) on 2000 FLUX-CiM health science references

| | Author | Title | Journal | Volume | Year | Pagination |
|---|---|---|---|---|---|---|
| Word-level accuracy (%) | 99.2 | 98.7 | 96.0 | 98.6 | 99.9 | 99.8 |
| Chunk-level accuracy (%) | 93.6 | 86.9 | 92.7 | 99.1 | 99.9 | 99.7 |
| Chunk-level precision (%) | 92.8 | 87.0 | 92.7 | 99.4 | 100.0 | 99.8 |
| Chunk-level recall (%) | 93.6 | 86.9 | 92.7 | 99.1 | 99.9 | 99.7 |
| Chunk-level F-Measure (%) | 93.2 | 86.9 | 92.7 | 99.3 | 100.0 | 99.7 |
| Chunk-level precision (%) in [7] | 98.6 | 84.9 | 97.2 | 96.4 | 99.8 | 99.7 |
| Chunk-level recall (%) in [7] | 99.0 | 85.1 | 89.3 | 98.7 | 99.5 | 99.2 |
| Chunk-level F-Measure (%) in [7] | 98.8 | 85.0 | 93.1 | 97.6 | 99.7 | 99.5 |