Citation Senser

CS6604: Digital Libraries Course Project

PROJECT CLIENT: Dr. Godmar Back

Submitted by:
Prasad Gopal, Suraj Menon, Veena Basavaraj
Date: 12/5/2005

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This project aims at developing an add-on feature to LibX-A Firefox extension for libraries, developed by our client Dr. Godmar Back and Annette Bailey. This feature will sense citations in any ad hoc web page and parse it to construct an OpenURL link for the citation entry. This project also aims at evaluating a few of the other OpenURL resolvers for the VT Library.

The major tasks of this project are:

- *To sense citations in web pages that follows certain standard styles.*
- *To parse the citations for key fields (metadata) like the author(s), title, publication, and year.*
- *To generate OpenURL links for the parsed metadata according to the OpenURL standards.*

The design goals of this project are summarized in order of their priority.

- *A browser solution that does not hinder the performance or user experience on the web while sensing and parsing the page contents.*
- *A solution that will avoid false positives and does not detect non-citation entries.*
- *A solution that will try to minimize the false negatives. The solution should parse atleast the standard citation styles and formats.*

This report provides a background on the existing literature and implemented solutions for sensing citation information on the web and also parsing the citation entries for metadata. The remaining sections of the report discuss the design, implementation, problems and tradeoffs we encountered in developing this solution in a browser environment and finally the evaluation results for the current implementation.

1. INTRODUCTION

This project aims at developing an add-on feature to LibX-A Firefox extension for libraries [23], developed by our client Dr. Godmar Back and Annette Bailey. LibX provides direct access to an institutions library resources via a firefox browser extension. The add-on feature will sense citations in any ad hoc web page and parse it to construct an OpenURL link for the citation entry. This project also aims at evaluating a few of the other OpenURL resolvers for the VT Library.

Citations can have one or more of the following parts. Comma, space or period separator is used between the parts

.

- [Serial Number]
  e.g. [1], [CLR90a], [Shaf98a]
- [Author(s) Name]
- [Title of Periodical or Book]
- [Publication]
- [Volume Number]
- [Number of Pages]
- [Date]

Certain standard citation style guidelines are available for various disciplines and are followed by users in those disciplines. Each style format includes the same basic parts of the citation listed above but are organized differently. Each specific style also has variations for different types of articles. Here is the list of different types of articles supported. They are:

- journal or magazine article
- newspaper article
- article from an Internet database
- book
- book article or chapter
- encyclopedia article
- web articles.

2. PROBLEM STATEMENT

Most often web articles or user's personal web pages that refer to (or cite) various literature and scholarly information do not have access to the actual cited documents and there are no hyperlinks to them. In some instances, even though a link to the cited resource is provided, all users may not have direct access to it. OpenURL standards are a solution to such a problem. OpenURL links contain the access information along with the citation metadata. They are directed to the OpenURL resolvers who take into account the identity of the user and provide access to the resources specified in the metadata.

3. SOLUTION APPROACH

The approach taken to solve the problem of providing access to cited documents based on users' access rights can be divided into three parts namely:

- To sense citations in ad hoc web pages that follows certain standard styles.
- To parse the citations for key fields (metadata) like the author(s), title, publication, and year.
- To generate OpenURL links for the parsed metadata according to the OpenURL standards.

Figure 1 is a concept map that describes the overall problem solution of this project.

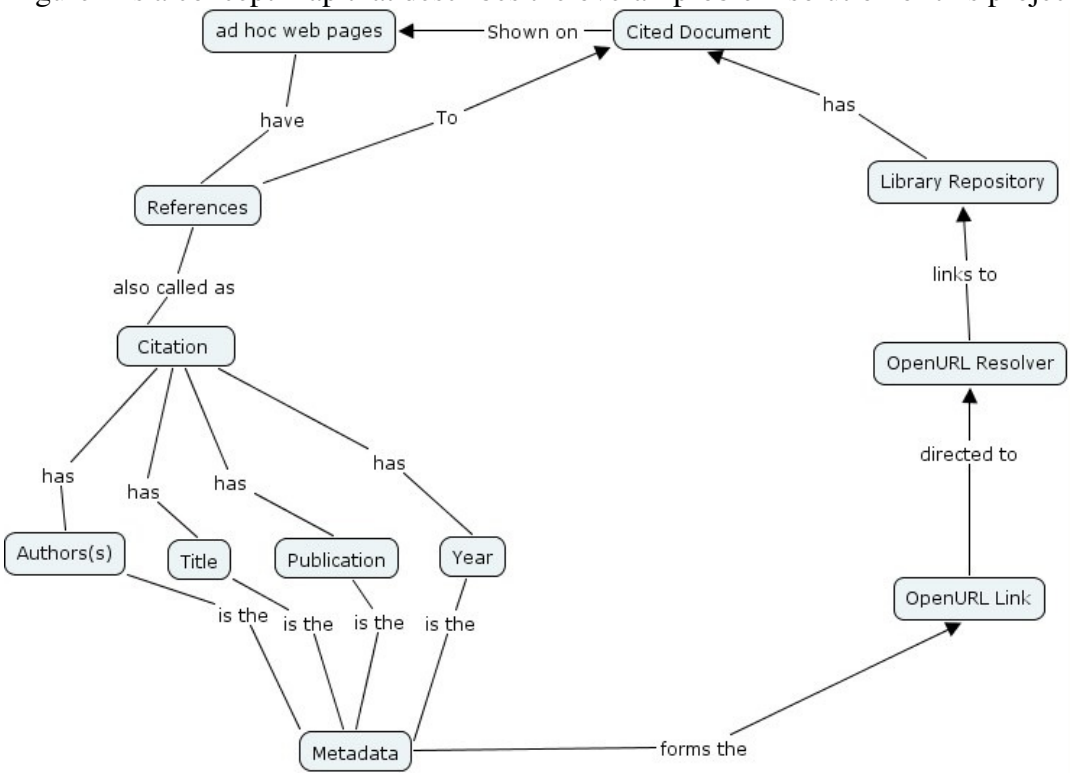


Figure 1: Concept map describing the Citation Senser

4. BACKGROUND

The section provides a review and brief explanation of the techniques and solutions that exist currently for citation sensing and parsing.

CITATION SENSING:

RECORD BOUNDARY EXTRACTION:

In order to parse the citations, we have to extract the records of interest from a web page. Once we narrow down to the correct sections of the web pages, we can apply

parsing to obtain metadata from them. We investigated certain existing record boundary extraction techniques to sense the citations in web pages.

[19] explains how we could get sections of interest from a web document. It does so by finding the section that could potentially contain what we are looking for and then applying few heuristic tests on those to extract the desired part. It applies this technique to find car advertisements and obituaries from newspaper websites. In this record boundary extraction technique we first find the node with the maximum fan out and assume that the maximum fan out node should have the records of interest. Then we apply heuristics to score the potential sections and infer whether each of these sections is of interest or not. This technique employs heuristic tests like ontological matching, repeating tag pattern test, standard deviation heuristic, identifiable separator tags and highest count tags and then applies combined heuristics to integrate the results from the above heuristic tests. The final scoring results indicate whether the records under consideration are the valid ones or not. In ontological matching the candidate sections are searched for related keywords and constants. The repeating-tag pattern heuristics test checks for patterns of tags that are repeated. The standard deviation heuristics test measures the length of text between two tags of the same name and then measures the standard deviations in these lengths for all types of tags present in the web document. The lower the deviations found for a type of tag, the better are the chances that the records are bounded by that tag. In the "identifiable separator tags" test we look for tags that are commonly used for separating records from each other. In the "highest count tags" test the tags are ranked based on the number of times they are present. The results of these tests are tested for various types of data and then a certainty factor is computed for each heuristics test. This certainty factor is used to integrate the results of individual tests. The final score decides whether the node defines the boundary for the records of interest.

However, [19] completely rules out cases where records are contained in repeated tree patterns. [20], [21] and [22], of Bing Liu, emphasizes on having a generalized idea on extracting records from a web page. They create a tree out of the html tags and then find patterns in trees and extract records at their correct boundaries. [20] extracts records present in the form of simple patterns. [21] uses a enhanced technique. It does a "simple tree matching" and defines the concept of "seed tree". Seed tree keeps track of all the tree patterns that occur in a tree. It is used as a reference tree for matching with the tree structures that are found while scanning the web page tree. [22] also handles cases where records have interleaving occurrence in the tree.

CITATION PARSING:

CORNELL DIGITAL LIBRARY research work:

In [1], reference linking is the term given to the process of dynamically linking to the cited documents from the citing paper. In the past, similar efforts have been taken up as part of the CrossRef [7] project which involved the IEEE, ACM and many others. The references to archival journals and proceedings in their digital libraries were cross-linked to aid researchers. Automatic reference linking lies between the Citation Index's static link discovery and the web author's manually inserted links. The work at Cornell is more generic and looks into irregularly formatted online documents (web pages) with a variety of reference styles and uses heuristics to extract the metadata.

The primary task is to analyze the entire document and find reference strings. This work not only analyses the reference strings but also matches the reference anchors in the body of the document to the tags in the reference strings (found in the references section). The online HTML documents are normally preprocessed to convert them into XHTML or ASCII so that parsing is easier. ResearchIndex [5] also known as the CiteSeer converts PDF/PS to ACSII using the PStotext utility. The Opcit project at Southampton [6] converts archived PDF to ASCII format before parsing using another utility. At Cornell, the Jtidy package is used to convert the HTML to XHTML. Once the XHTML is obtained, readily available XML parsers are used to analyze the text. In order to locate the list of references and extract metadata for each of the reference strings, the deciter package from Southampton [6] is used. The document is scanned for section headings like references, bibliography. The Southampton deciter routine outputs the metadata obtained from the reference string in the form of an XML. Then a DOM parse of XML is done to extract the information of interest and an URN (Uniform Resource Name) is synthesized.

The Deciter routine is similar to adddoc Perl module of Research Index [5] and uses clever heuristics. The details of how an author's name is parsed using deciter can be obtained from the Text::BibTex::Name Perl module. Existing tools find it best not to parse the reference string from left to right, but to first isolate date and page range if present from the reference string. Then parse the string for author names until the (") or (') character is encountered. This is considered as the starting point for title. The text obtained from this point till the publication date is parsed as the title.

The secondary task is to link reference strings to full-text documents using OpenURL resolvers like SFX. This is dynamic and is done at presentation time rather than analysis time like in [5]. It treats reference link resolution as a separate task from reference link analysis. Both static and dynamic linking is supported. Static linking is done by collecting information from each of the analyzed documents and is stored in the database for later use. An object oriented (java) API has been built at Cornell [2, 3] that creates a surrogate object for each document. The first time a surrogate object is instantiated for a document, the surrogate parses it and collects reference linking information. This information becomes part of the data encapsulated in the surrogate. This data is distributed through the API for other client programs to perform static link resolution.

PARATOOLS:

Paratools is a set of Perl modules for handling citations. It includes:

- Reference parsing modules and templates
- Document parsing modules
- OpenURL creation and processing routines

The main modules developed as part of [6] are the reference parser, reference resolver, web query and web service interface and the OpenURL query interface. The reference parser is also called the ParaCite parser or Paratools [6]. This is written entirely in Perl, with its regular expression support making up the core of the parsing functionality. This tool uses a high-level template matching scheme. Here template means a reference or citation style. A collection of reference templates exist and these are being compared against the reference strings and the best fitting template is used to split the reference string information into metadata. Each field in the reference information,

i.e., the author, publisher, and title, are weighted for each of the template using matches with its corresponding regular expressions. The template with the highest overall score is chosen and metadata is constructed as per the template.

ParaCite has been integrated with the EPrints.org software. It is used to find free online versions of existing journals. The authors of [6] also claim that the web service interface can be used to create plug-ins for existing software, such as web browsers and document editors and this would automatically create OpenURLs for references.

AUTOBIB:

AutoBib [8] is another system that caters to extracting bibliographic information on the WWW. Rather than relying on pattern matching heuristics alone, statistical techniques are also used. The AutoBib mainly has 3 steps to resolving the citations. A one time first step is to create a seed database from an existing bibliographic source. This seed database will contain structured records. Structured records are created from sample reference strings after resolving each field of the citation data and identifying its type. For instance, consider the following citation. "Pankaj K. Agarwal and Pavan K. Desikan. An efficient algorithm for terrain simplification. In Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 139-147, New York / Philadelphia, January 5-7 1997. ACM / SIAM".

They do a space-delimited tokenization of the reference text. The tokens obtained from this process are annotated with keywords like author, title, date, etc. ("Pankaj"), ("K"), ("Agarwal") are identified as author, ("An"), ("efficient"), ("algorithm") are identified as words in the title and so and so forth. These are stored in the seed database. This is just a one time process.

The second step is to extract raw webpage's from similar bibliographic sources (DBLP, CCSB). The contents of the web pages are analyzed using record-boundary techniques and the output of this process are tokenized strings. The tokens contain numbers, delimiters, tags and words. Once the tokens are created, matching is done with the structured records in the database using direct string matching and a set of three heuristics. They are the contains-in heuristic, grouping heuristic and acronym heuristic. At the end of this there are references that have tokens that are annotated as author, title, date, etc. This annotated data is used as training data for the HMM parser. The references that did not match the structured records are fed into the HMM parser.

The last step is to parse the unknown fields into known fields. For this a HMM is created. A HMM is defined by five things, which are as follows:

- set of known states
- set of observed symbols
- transition probability matrix from state i to state j
- emission probability matrix of emitting a symbol j when in state q and initial probabilities of each state.

The next job is find out the optimal sequence of states that led to the current sequence of observed symbols (in this context, the sequence of observed symbols refers to the sequence of raw tokens). The optimal solution is obtained using the dynamic programming method called the viterbi algorithm. This algorithm selects one optimal sequence of states that has highest probability value. Once the sequence of states is

determined; it is easy to identify the type of the unknown token. For instance if the 5th token is unknown, the 5th state in the optimal sequence would identify the type of the unknown field.

TEMPLATE MINING:

One other system [10] uses template mining for extracting citation information. Four templates have been created using template mining, one for extracting information from the citing articles, and the other three for extracting information from the cited articles (which is the citation list). The details of the template are available in [10]. Template mining is used to extract data from the text based on observed patterns. When the text matches a template the corresponding data is extracted according to the instructions associated with that template. The templates were created using an iterative process of template specification, evaluation and modification. Many journal articles which strictly follow the citation style guidelines were analyzed and patterns were identified. The citing article template was derived by analyzing 43 articles with 1067 citations. The cited article template was derived by analyzing 34 articles with 1112 citations. There is no concrete information on the citation styles of these entries though the names of the 12 journals from which these samples were taken are given. The paper concludes that the proposed templates are just initial prototypes and require further work. Also the template's ability to recognize the citation components will be determined by the sequence and matching process that will be used.

GLR PARSER:

LR parsers are not effective in handling ambiguous situations [11]. Masaru Tomita brought up the concept of Generalized LR parser which could be used for ambiguous grammars especially in case of natural language processing. The parser used a GLR stack (variation of LR stack) which necessitated operations very complex in nature. [12, 13, 14] describe the efforts made to reduce this complexity by using a probabilistic model to parse string data. These methods use probabilities that are known for each action of LR tables in order to determine the shift or reduce actions based on the LR state. The work in [15] also describes a probabilistic model, but it uses information from the LR stack of the partially parsed tree for the contextual information. This idea is called "Condition Access Model". The work in another paper [16] claims to make this better by using the structural characteristics of the parsed tree. This is represented by "Surface Phrasal Types". The authors of [17] explain an application of the GLR parser in natural language processing. It uses "grammar partition" and "parser composition" approaches to improve the efficiency of GLR parsing.

Grammar Partitioning is done by dividing the main grammar into sub grammars. Each of the sub grammars is divided into sub grammars and is assigned level numbers. Sub grammars that need to interact with each other have the same level. Thus a hierarchy of sub grammars is formed under the main grammar. This is possible by introducing the concept of virtual terminals. Virtual terminals act as terminals for a sub grammar, even though they are non-terminals. Cascaded parsing and predictive pruning are the two main techniques of Parser Composition.

The GLR parser introduced in [18] has the capability to skip unrecognizable parts in a sentence and parse any text. It is used in parsing spontaneous speech. However this idea could be used in cases where we would need to skip unwanted parts found on citations on web pages such as unwanted HTML tags of tables or hyperlinks in between

them. The authors of [18] explain few heuristics that could be used to evaluate partial parse trees generated by the GLR parser.

CiteUlike:

CiteUlike is a free service to help academics to share, store, and organize the academic papers they are reading. CiteUlike automatically extracts the citation details from specific sites and a personal online library of reference information can be created. The system is not open source, but there is a possibility of using such online citation reserves to implement a server support based Citation Senser. One main issue with such an implementation would be performance.

5. COMPARISON OF DIFFERENT DESIGN APPROACHES

We can broadly classify the different systems studied under the following categories.The different approaches are compared using few variables. A brief explantion of each of the following is as follows.

The rule or heuristic based approach uses regular expression matching and heuristics to identify parts of the citation.The GLR parser based approach is based on grammar definition and is efficient.The statistical approach uses the probability based HMM model.The server support based solution relies on external servers which maintain citation data for resolving parts of the citation.

Parameters	***Rule or Heuristic Based Solution***	***Parser Based Solution***	***Statistical Based Solution***	***Server Support Based Solution***
Example System(s)	*Paratools, Cornell's work, Template mining*	*GLR parser used in NLP*	*AutoBib using the Hidden Markov model*	*Using external system like CiteUlike*
Citation Styles supported	*Around 400 (limited)*	*Depends on the grammar constructed*	*Depends on the training data*	*Depends on the results from the external server*
System trainable	*No*	*No*	*Yes*	*No*
Efficiency/Complexity of parsing	*Sequential and not efficient*	*Efficient parsing*	*Efficient*	*Efficient*
Performance/Scalability	*Slow, not scalable for large datasets*	*Scalable*	*Scalable but not for browser based implementations*	*Scalable (The request /response time might increase.)*
Can it be implemented	*Yes*	*Yes*	*Yes, Limited by*	*Yes*

on the browser			*the size of training data set*	
Complexity of Implementation w.r.t to team skills	*Low*	*High*	*Medium*	*Low*
Track record of tool used	*Citebase citation analysis of arXiv physics papers*	*Normally used for parsing spontaneous speech*	*AutoBib an academic research work only for CS related citation data*	*CiteUlike has an online repository of citation data, but there is no facility to access this data*

6. CITATION-SENSER DESIGN

DESIGN OVERVIEW:

The Citation Senser has the following parts.

- ***Citation Record Boundary Detection*** - find the sections of the web pages that contain the citations.
- ***Citation Record Paring*** – The raw citation data is parsed to identify its different parts in order to construct the metadata for the OpenURL.

Figure 2 provides a high level overview of the Citation Senser design using concept maps**.**

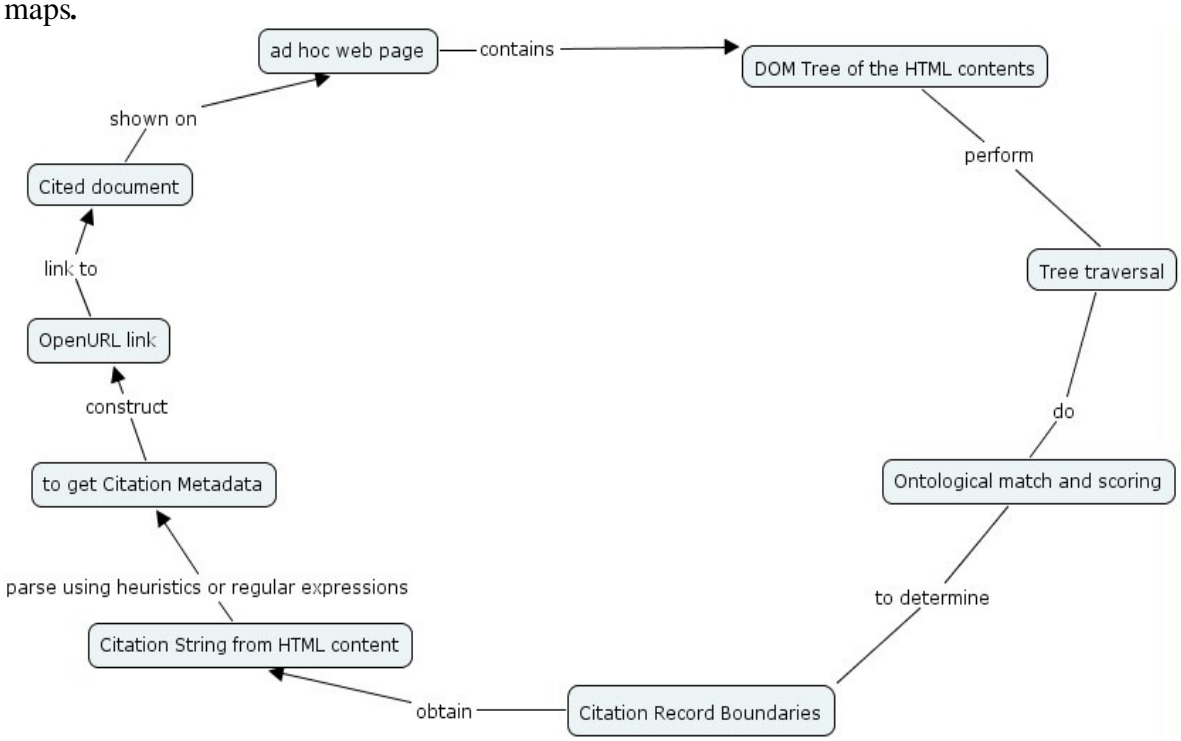


Figure 2: Citation Senser Design Overview

DESIGN GOALS:

The design goals of this project are summarized in order of their priority.

- A browser solution that does not hinder the performance or user experience on the web while sensing and parsing the page contents.
- A solution that will avoid false positives and does not detect non-citation entries.
- A solution that will try to minimize the false negatives. The solution should parse atleast the standard citation styles and formats.

DESIGN DETAILS:

<u>Record Boundary Detection:</u>

A DOM (Document Object Model) tree of the web page is available. It is a tree representation of the html tags (called nodes) and the text within them. There are numerous patterns in which the citation data exists in web pages. Moreover it could be embedded at any hierarchy of the DOM tree. Records are normally found in the following ways: Figure 3 shows the repeated patterns of tags in web pages.

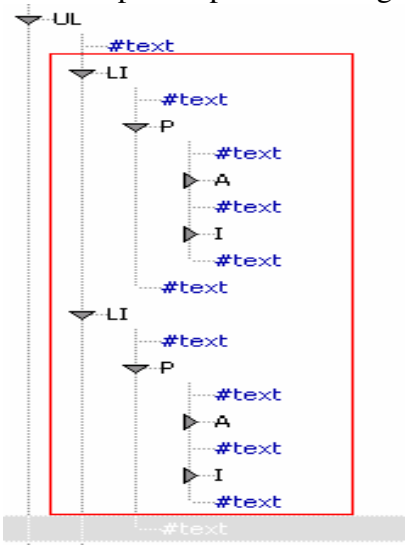
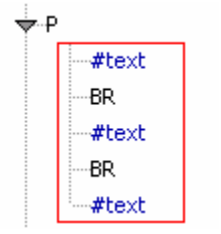


Figure 3: A Repeated tag pattern DOM Tree

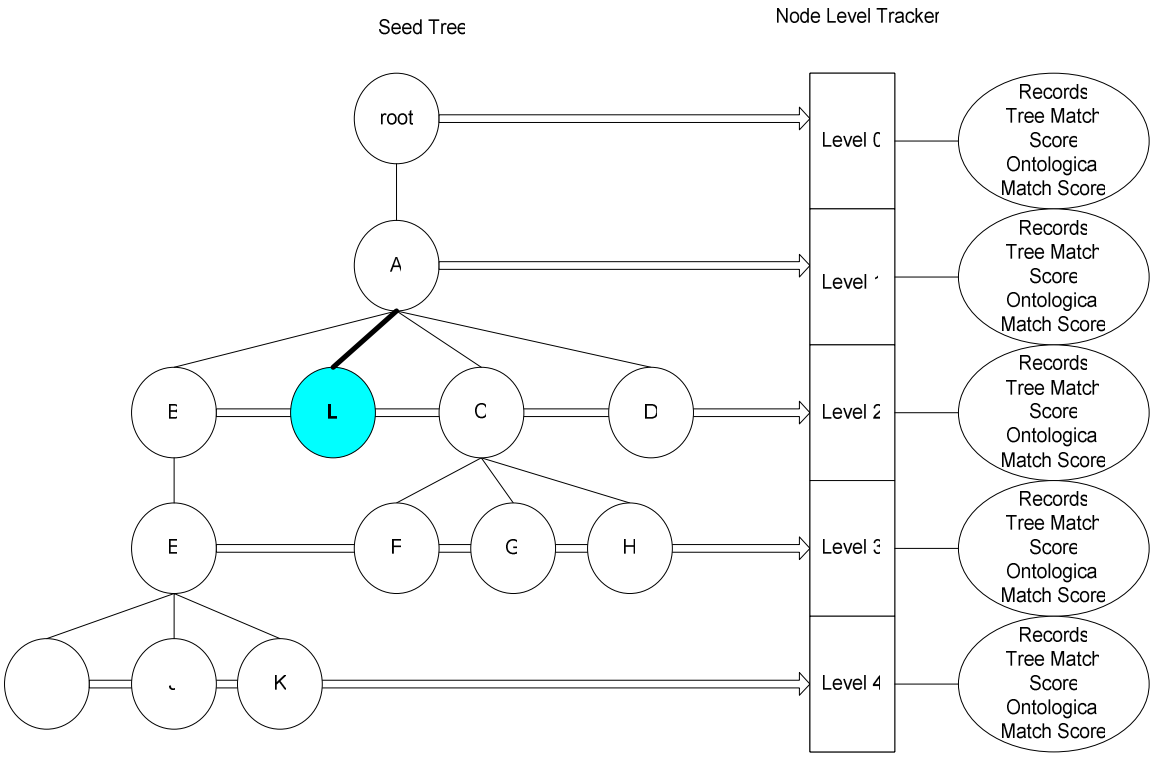


Repeated Pattern tag

From these extracted patterns, we have to make sure that we extract the correct records from their exact boundaries. Our work is very much based on the efforts of [20] & [21].

The steps involved in this process are given. The details of the design are also depicted in the concept map Figure 5.

- First we find all the heading tags in a DOM tree in their order of occurrence. We did this by using XPath expressions.
- Ontological matching for keywords such as Publication, Journal, Report, Conferences, and References is performed on the text content of these heading tags. If matched we boost the score of the records found in subsequent operations.
- For each heading tag we do a post order traversal of the part of the DOM tree that contains text which is usually below the heading nodes.
- As we do this post order traversal, we create a "universal pattern tree" (on similar lines of "seed tree" in [21]) that keeps track of the patterns that are occurring within that section of the DOM tree till the point we have completed our post order traversal of the DOM tree.



- As shown in the above figure, the uncolored part of the tree is the seed tree that we obtained till the point of addition of new node "L". The pattern root>A>L was not encountered by the seed tree till then. Hence it gets added to the tree. From this point, any more occurrence of pattern root>A>L would contribute to the tree-

match score of node A at level 1. Now at every level we also do an ontological test, where we look for certain keywords and score every node for that.

- The node-level tracker keeps track of all these scores and the records that are found at each level. After saving all possible records at a particular level we move back to the upper level to do the same.
- Text length<50 are not considered as citation records. This eliminates the false positives that even pass the ontological matching.
- Finally we scan from the top, for records that has a cumulative score greater than a predefined threshold value.
- For all such records extracted, we discard the records that are saved at lower levels of node-level tracker for the current record.
- We record each and every node that has already been scanned to avoid repeated traversing.

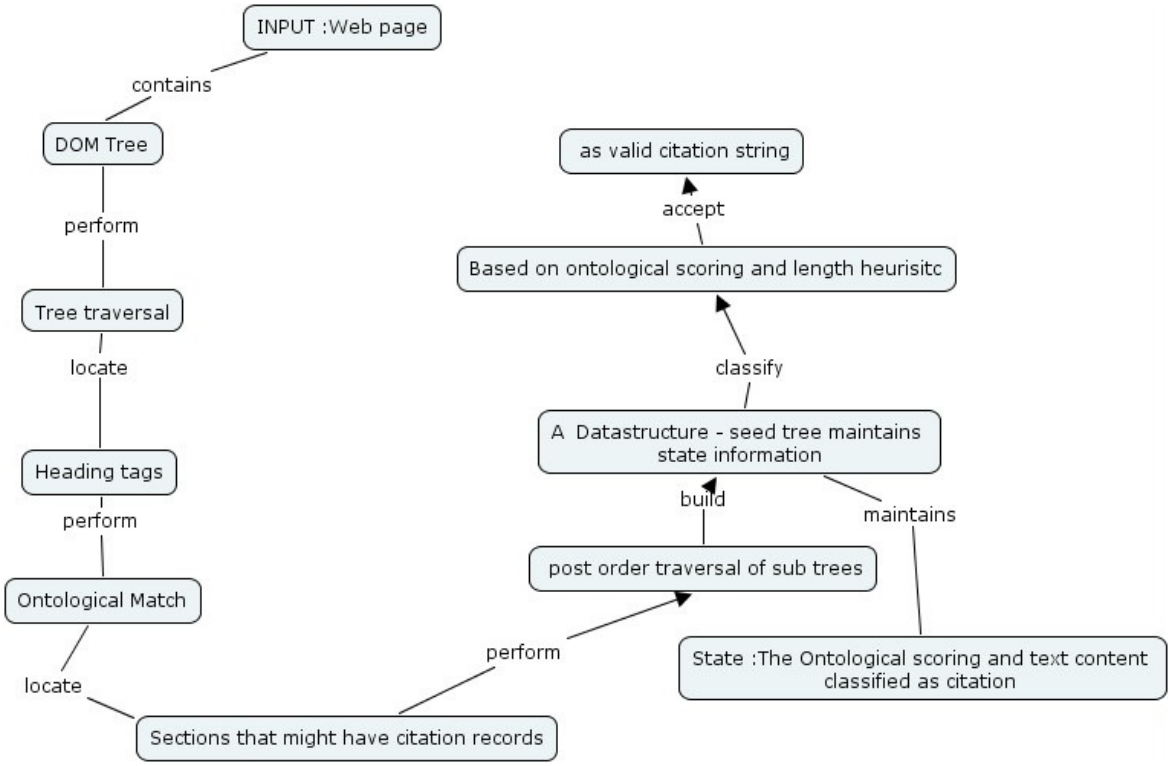


Figure 4: Record Boundary Detection

Citation Record Parsing:

Approach 1: This uses ontological matching and heuristics to extract the metadata. The boundaries that divide the key fields are determined using heuristics. The details of the design are depicted in the concept map (Figure 5). Few of the heuristics used are:

- The title occurs between decimeters such as " " ,' ' and within <b> ,<i>html tags.
- The authors usually occur in the beginning or in the initial two-thirds of the citation string.

Ontological matching for keywords such as "publication" ,"journal" ,"proceedings" ,"volume" ,"pages" provide information on the boundary for the occurrence of the

publication field in the citation string. Regular expression is used to determine the author(s).

- In cases where we don't get any metadata from the citation record that could be given to the VT Article Linker Open URL resolver, we create a Google Scholar link out of a part of the citation record. We expect that the Google Scholar results from this link should give an Open URL that is already been provided by the LibX extension.

Merits and Demerits of Approach 1:

- *Faster and more suitable for a browser implementation*
- *False negatives are likely due to use of heuristics*
- *Ontological matching can trigger false positives*

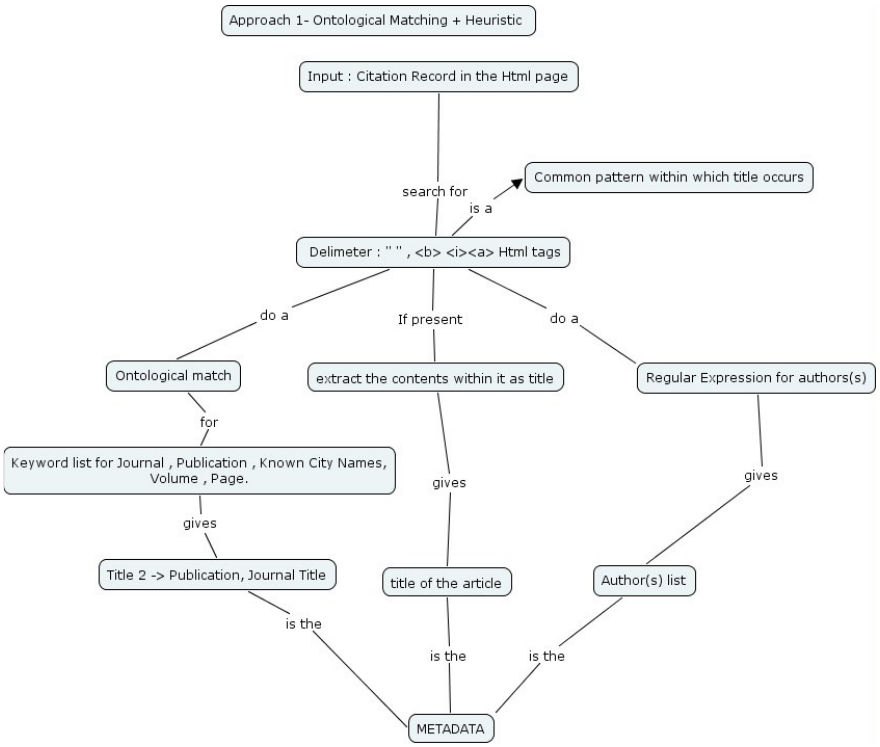


Figure 5: Concept Map for Citation Parser - Approach 1

Approach 2: This uses the regular expression based template matching and scoring. The details of the design are depicted in the concept map (Figure 6). The salient features of this approach are:

- Templates represent citation styles for journal articles, books e.t.c. An example of a template is 'AUTHORS_ (_YEAR_) Precis of "_TITLE_". _PUBLICATION_ _VOLUME_(_ISSUE_)',
- These templates contain tokens (for instance ,YEAR, TITLE, ISSUE) which map to the key fields of the citation such as author , title , publication , volume , pages e.t.c

- The tokens in the template are represented by regular expressions. Each token has a reliability score.
- The Reliability scores associated with a few tokens are shown below.:
  "_ISSN_"=> 0.95,"_AUTHORS_"=> 0.65,"_EDITOR_"=> 0.6,"_DATE_"=> 0.95,
- The citation string in sequentially matched with each of the available templates represented by regular expressions to determine the format of the citation. Since this can result in more than one match, reliability and concreteness scoring is done.
- *Reliability scoring*: This returns a value that is an indicator of the likelihood of a template matching correctly. Fields such as page ranges, URLs, etc., have high likelihoods (as they follow rigorous patterns), whereas titles, publications, etc. have lower likelihoods.
- *Concreteness scoring*: This return a value that indicates the number of non-field characters in the template. The more 'concrete' a template, the higher the probability that it will match best. For example, '_PUBLICATION_ Vol. _VOLUME_' is a better match than '_PUBLICATION_ _VOLUME_', since _PUBLICATION_ is likely to subsume 'Vol.' in the second case.

Merits and Demerits of Approach 2:

- Sequential matching, slower , less suitable for a browser implementation
- Accuracy is low for multiple author citation strings
- Generic templates like _TITLE_, YEAR trigger false positives. Examples of such problems are discussed in the following section.

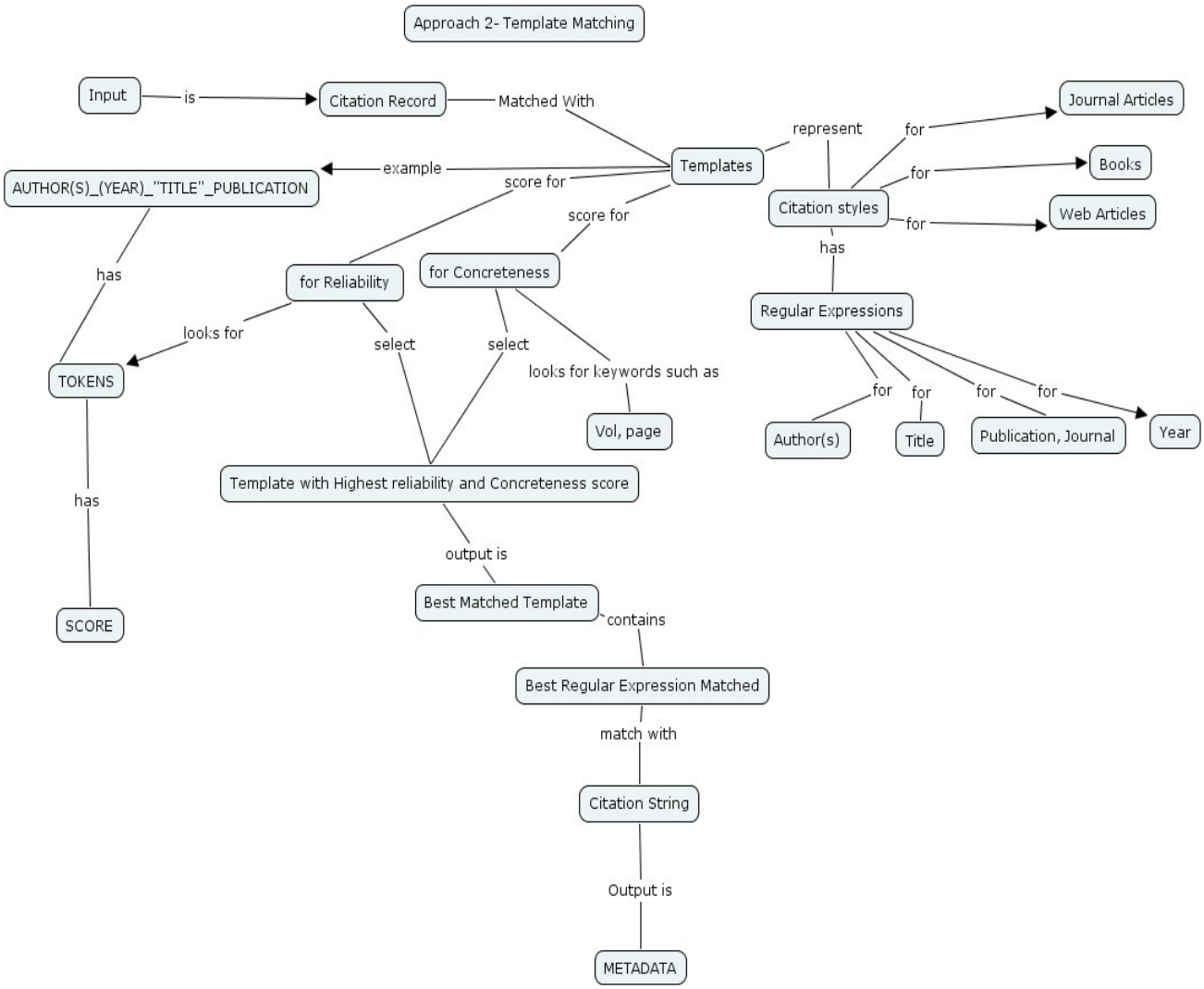


Figure 6: Concept Map for Citation Parser - Approach 2

7. PROBLEMS AND TRADEOFFS IN IMPLEMENTATION

This section outlines the problems encountered during the implementation. It also discusses the tradeoffs used to meet the design goals.

Problem 1: *There are no standards to creating web page content. The citation text can be embedded in any tag and at any hierarchy. Users have their own styles of writing html content using different tools to create them. False positives were generated in many web pages. Record Boundary technique was the bottleneck.*

Figure 7 is a snapshot of the false positives from the naïve Record Boundary. The keyword "Conferences" in the heading tag is picked up as a section heading that contains citation records.

Figure 7: Snapshot of false positives

Solution:

- Enhanced Record boundary as described in Figure 4.The Record Boundary recursively searches through all levels to avoid false positives and also minimize false negatives.

Consequence: Performance is affected for longer web pages with large text content.

Problem 2: *Performance is a major concern in the browser extension. Both space and time optimal solution is the prime requirement. The user experience on the web should not be throttled due to the extension.*

Solution:

- The approach-2 parser implementation described in Section 6 caters to many citation styles, but the sequential regular expression matching slows down the parsing process. In order to eliminate this, approach -1 was implemented which is based on heuristic and ontological matching.

Consequence:

- False negatives. Figure 9 is a snapshot of the parsing extension results on a faculty page. The existing heuristics fail to sense most of the citation entries that do not have the ontological entries we are maintaining.

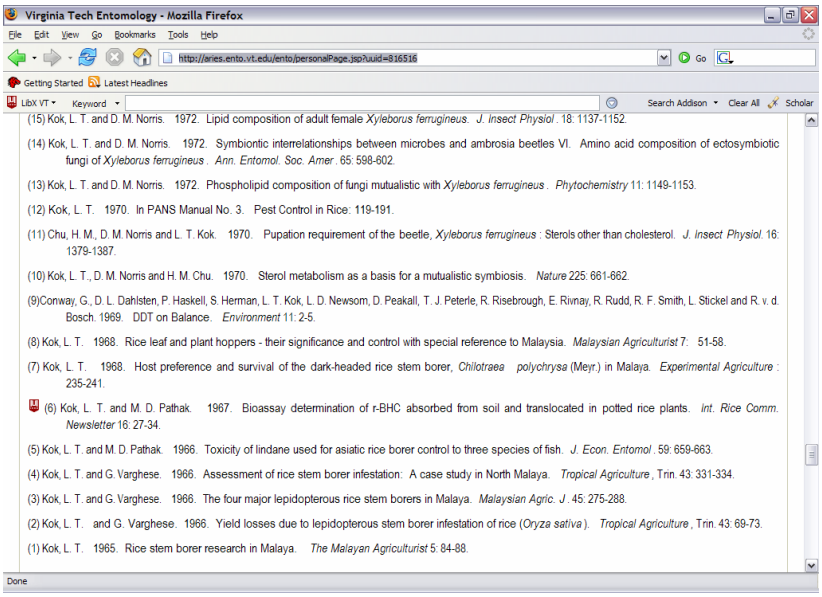


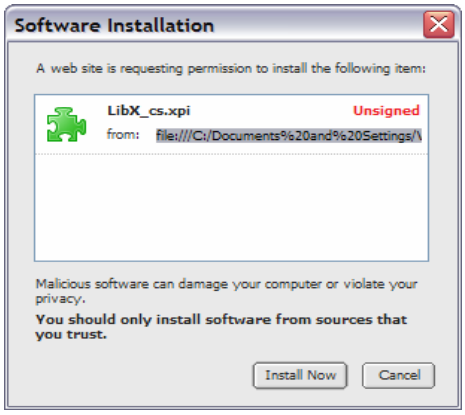
Figure 8 : Snapshot of false negatives

8. USER MANUAL

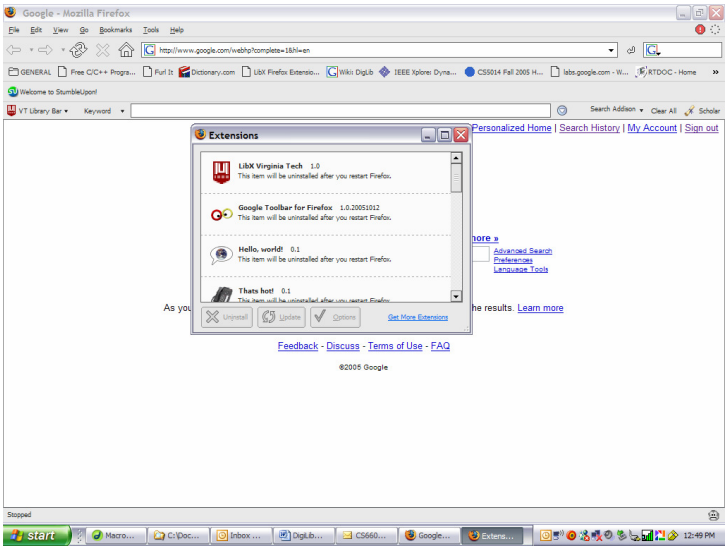
Firefox extensions are enhancements to the existing browser features. Our implementation has been tested on the Firefox 1.0.x. it has not been tested on the latest Firefox 1.5 version.

Installing the Extension: One way to install the extension is:

- Go to the Firefox browser File menu -> open File -> LibX_cs.xpi. The following screen is displayed.



- Press on Install Now. The following screen is displayed



- Close and Restart Firefox for the extension to be activated.

Uninstall the Extension: Go to the menu Tools -> Extensions. Select the entry in the list. Click "Uninstall" at the bottom left (it looks like an X). Restart firefox.

9. DEVELOPER MANUAL

The following is a brief description of few of the important JavaScript classes developed for the firefox extension.

Class: ontoLogicalTest

ontologicalTest class is responsible for ontological matching of any piece of text with certain keywords that are related to citation records. An instance of this class could score text based on presence of keywords related to citation records. It can also parse the text of citation records for extracting metadata out of it.

Constructor:

ontoLogicalTest()

Creates an instance of the class

Attributes:

keywords	Contains array of regular expression for matching keywords
treg	Regular expression that looks for title in double quotes
Authors	Regular expression that matches most of the styles of occurrence of author in a citation record

Methods

doMatch(text)	matches the input text for keywords and score the piece of text accordingly parameter: String *text*
doAdvancedMatch(text)	parses the text with certain heuristics and extracts metadata out of the text parameter: String *text*
getScore()	returns the average score
getMaxScore()	returns the maximum score
getComprehensiveScore()	returns the score from advanced scoring mechanism
getMatchedComponents()	returns array of with indexes that are types of parts of the citations
getMatchedComponentString()	returns string equivalent of types of the parts of the citations

Class: record

Represents a record.

Constructor: record(nodes, score):

Parameter: *nodes* - array of HTMLElement
text - String containing the citation record.
creates an instance of records with nodes and the text in those nodes.

Attributes:

cumulativeScore	Total score after ontological match and tree match
nodeArray	Array of nodes that make the record
textContent	Total text contained in the record
title	Title extracted from the text in record
journal	Journal name extracted from the text in record
author	Author name extracted from the text in record

Methods:

isAlreadyIncluded()	Returns true if *this* is already included in a record at higher level.

accepted()	Signals that the current record is accepted. It updates the attributes *title*, *author* and *journal* by parsing the text.

Class : levelTracker

An instance of this class keeps track of the treematch score and records formed at each level of the *seed tree.*

Constructor: levelTracker(l)

Parameter: *l* – level number

Attributes:

level	Level that is getting monitored
records	Array of *record*
nodes	Array of HTMLElement that are been added before creating a records out of them
nodeTracker	Array that uses its index to log tree matching.
THRESHOLD	The minimum value that a record must have to qualify as a valid record
recordExtractedFromNextLevel	True if records are extracted from the next level.

Methods:

createRecords()	creates record from the *nodes* after computing their score.
addNode (stNode, previousNodeTracker)	Logs the presence of the type of HTMLElement in the current level. Parameter *stNode*: seedTreeNode *previousNodeTracker:* nodeTracker at previous level.
LeaveCurrentLevel	Creates record and reinitializes the score and *nodes* array.
computeCumulativeScore	Computes the cumulative score for making *record* been made.

Class : seedTreeNode

A node in the seed tree.

Constructor: seedTreeNode(node)

Parameter: node – HTMLElement

Creates an instance of seedTreeNode of type of *node.* If node is null for root of the tree.

Attributes:

name	Name of the type of the node
ontoScore	Ontological score obtained for the node
node	HTMLElement that instantiated the seed tree node
levelTrackerArray	Array of level tracker and is used only if the current node is the root node.

Methods:

addChild(stNode, stRoot)	Adds a seed Tree node stNode as a child Parameter : *stNode* – seedTreeNode
leave( )	Signals the traversal of the node is complete and hence updates the levelTracker Parameter *stNode*: seedTreeNode *previousNodeTracker:* nodeTracker at previous level.

Class: seedTree

Seed Tree created out of patterns in tree.

Constructor: seedTree(node)

Parameter: node – HTMLElement

Creates a seed Tree with node of type of HTMLElement node.

Attributes:

root	Root of the tree

Methods:

addChild(node)	Adds a seedTreeNode of type of HTMLElement node. Parameter : *node* – HTMLElement
getRecordsLength	Returns the length of records at the level

getRecords( )	Extracts records from the level

Class: recordBoundaryTest

Extracts citation record boundaries and places an icon with an Open URL link before them.

Constructor: recordBoudaryTest(d)

Parameter: d – HTMLDocument

Creates an instance of recordBoundaryTest that would extract records boundaries for the HTMLDocument d

Attributes:

doc	HTMLDocument
tagList	Array that indicates types of tags.
regExps	Regular Expressions for doing ontological Match on heading tags.

Methods:

nextValidSiblingNodes()	Returns array of nodes from which we have to create a seed Tree and extract records
applyTest()	Applies the record boundary test and makes required changes in the DOM tree in case records are found.

Class: CitationParser

Parses citation record using templates (as in Paratools) and scores for each of the template matched

Constructor: CitationParser (rec)

Parameter: rec– CitationRecord

Creates an instance of CitationParser that would match all the templates and give the best-match template and finally the metadata

Attributes:

templates	Array that contains the citation styles
matches	Array that contains the regExps for the tokens in the templates
factors	Array that contains the reliability score for each of the tokens in the matches

Methods:

convertToRegExps(Templates)	Substitutes and replaces the template tokens with the regExps defined in matched
doMatch(rec,Templates, bestMatch, bestOrig	Matches the rec with each of the substituted templates and returns the bestmatch template
getReliability( )	Scores each template for reliability
getConcreteness()	Scores each template for concreteness
extractMetaData(rec)	Returns an Array containing the metadata of the given record
getMetaData(rec)	Called by extractMetaData which inturn handles multiple authors and extracts the field boundaries
mapToOpenURL(rec)	Creates a OpenURL link of the metadata

10. EVALUATION

We have tested our current extension on 50 web pages. Most of these web pages were home pages (specifically those containing resumes) of professors in Computer Science field. At this stage the extension does not cater to other domains and requires to be ontologically trained for other domains. The following table gives an account of our evaluation.

	URL	Extracted records	Parsed correctly	VT Article Linker gives correct OpenURL
1.	http://people.cs.vt.edu/~gback/	✔	✔	✔
2.	http://www.nvc.cs.vt.edu/%7Ebohner/sbohner-cv.htm	✔	✔	✔
3.	http://people.cs.vt.edu/%7Ebowman/	✔	✔	✔
4.	http://people.cs.vt.edu/%7Eirchen/	✔	✔	✔
5.	http://www.cs.vt.edu/info/people/vitae/Egyhazy.html	✘	✘	✘
6.	http://www.teacherbridge.org/public/users/dunlapd/Resume	✘	✘	✘
7.	http://frakes.cs.vt.edu/frakespubs.html	✔	✔	✔
8.	http://fox.cs.vt.edu/cv.htm	✔	✔	✔
9.	http://people.cs.vt.edu/~vchoi/	✔	✔	✔
10.	http://people.cs.vt.edu/%7Eedwards/	✔	✔	✔
11.	http://people.cs.vt.edu/%7Earthur/	✔	✔	✔
12.	http://europa.nvc.cs.vt.edu/%7Eathma	✔	✘	✔

	n/			
13.	http://people.cs.vt.edu/~barnette/	✔	✘	✘
14.	http://people.cs.vt.edu/%7Ekafura/	✔	✔	✔
15.	http://europa.nvc.cs.vt.edu/~ctlu/	✔	✔	✔
16.	http://people.cs.vt.edu/%7Eheath/vita/cv_html/cv_html.html	✔	✘	✘
17.	http://people.cs.vt.edu/~onufriev/publications.html	✔	✘	✘
18.	http://www.cs.vt.edu/info/people/vitae/Hartson.html	✔	✘	✘
19.	http://aero-comlab.stanford.edu/juanjo/	✔	✘	✘
20.	http://aero-comlab.stanford.edu/jameson/publication_list.html	✘	✘	✘
21.	http://www.cse.buffalo.edu/faculty/miller/Biographical.htm	✔	✔	✔
22.	http://www.cse.buffalo.edu/faculty/selman/	✘	✘	✘
23.	http://www.cedar.buffalo.edu/~rohini/publications.html	✔	✔	✔
24.	http://www.cse.buffalo.edu/%7Eshambhu/publications.htm	✔	✔	✔
25.	http://www.cse.buffalo.edu/faculty/mbeal/papers.html	✔	✔	✔
26.	http://www.cse.psu.edu/~acharya/	✔	✔	✔
27.	http://www.cse.buffalo.edu/faculty/alphonce/Publications/	✔	✔	✔
28.	http://www.egr.msu.edu/~nrm/research/pub.html	✔	✔	✔
29.	http://www.cse.buffalo.edu/~xwang8/	✔	✘	✘
30.	http://www.cs.sunysb.edu/people/faculty/ArieKaufmanAB.html	✔	✔	✔
31.	http://www.cs.sunysb.edu/~rtjohnso/papers.html	✔	✔	✔
32.	http://www.cs.sunysb.edu/people/faculty/XianfengGuAB.html	✔	✘	✘
33.	http://www.cs.sunysb.edu/people/faculty/DimitrisSamarasAB.html	✔	✔	✔
34.	http://www.cs.sunysb.edu/people/faculty/MichaelTashbookAB.html	✔	✔	✔
35.	http://www-dsg.stanford.edu/Publications.html	✔	✘	✘
36.	http://www.cs.sunysb.edu/people/faculty/AlexanderMohrAB.html	✔	✔	✔
37.	http://www.cs.purdue.edu/homes/li/publications.html	✘	✘	✘
38.	http://www.cc.gatech.edu/~ammar/pby.html	✘	✘	✘
39.	http://www.bell-labs.com/user/rastogi/pub.html	✔	✔	✔
40.	http://blrc.edu.cn/blrcweb/publication.htm	✔	✘	✔

41.	http://www.isr.umd.edu/~baras/	✔	✘	✘
42.	http://www.cs.virginia.edu/brochure/profs/batson.html	✔	✔	✔
43.	http://www.cs.umd.edu/%7Ebederson/papers/	✔	✔	✔
44.	http://www.bell-labs.com/user/risbood/	✘	✘	✘
45.	http://www.cs.purdue.edu/homes/cmikkels/monalisa/monalisa.html	✔	✔	✔
46.	http://www.cc.gatech.edu/~goodman/	✔	✘	✘
47.	http://www.cc.gatech.edu/%7Edellaert/	✘	✘	✘
48.	http://www.cc.gatech.edu/fac/Blair.MacIntyre/papers.html	✔	✘	✘
49.	http://www.cc.gatech.edu/~jpierce/	✔	✔	✔
50.	http://www.cs.washington.edu/research/embedded.intro.html	✔	✔	✔

11. EVALUATION RESULTS

We found that we were able to find the correct record boundaries in most cases. If the extension detects few of the records listed in the web page, we consider the record boundary extraction to be successful. From the results till now, we have success-rate of more than 80% for the citation sensing. For 70 % of web pages where record boundaries were extracted successfully, the heuristics based parser was successful and parsed correctly. Hence it was also able to provide the correct Open URL link. In few other cases it was able to provide an appropriate Google Scholar link. This URL results in a page that contains the Open URL link for the scholarly article (a feature that was already implemented in the LibX extension by Dr. Godmar Back and Annette Bailey [23].

12. FUTURE WORK

Though the prototype implementation demonstrates a working solution in the browser environment, there is lot of scope for improving both the record boundary detection and parsing. The following is a list of future improvements.

- *There is scope for improving the parser to support variety of citation styles using better heuristics. This will reduce the false negatives.*
- *Look-up dictionaries of authors and title keywords can be integrated into the heuristic parser solution. The dictionaries are updated with author and title contents each time OpenURL resolver successfully resolves to a cited document in the library repository. This can improve the performance.*
- *Ontology should cater to different domains. The Ontological scoring can be improved by comprehensively testing for various webpage's that have different citation formats. This will reduce false positives and false negatives.*
- *Another direction to improve the parsing is to explore the GLR parsing technique*
- *Server support for parsing citations with google scholar support is also a good alternative to improve the accuracy of parsing.*

REFERENCES

1. Bergmark, D. 2000 Automatic Extraction of Reference Linking Information from Online Documents. Technical Report. UMI Order Number: TR2000-1821., Cornell University

2. Bergmark, D. and Lagoze, C. 2001 An Architecture for Automatic Reference Linking (Extended Version). Technical Report. UMI Order Number: TR2001-1842., Cornell University.

3. Bergmark, D. and Lagoze, C. 2001 Reference Linking the Web"s Scholarly Papers. Technical Report. UMI Order Number: TR2001-1835., Cornell University

4. Sergey Brin. Extracting patterns and relations from the world wide web. In WebDB, Workshop at EDBT '98, 1998

5. C. Lee Giles, Kurt Bollacker, and Steve Lawrence CiteSeer: An automatic citation indexing system. In Ian Witten, Rob Akscyn, and Frank M. Shipman III, editors, Digital Libraries 98 - The Third ACM Conference on Digital Libraries, pages 89--98, Pittsburgh, PA, June 23--26 1998. ACM Press

6. ParaTools: http://paracite.eprints.org/developers/

7. CrossRef: http://www.crossref.org/

8. Junfei Geng; Jun Yang; AUTOBIB: automatic extraction of bibliographic information on the Web, Database Engineering and Applications Symposium, 2004. IDEAS '04. Proceedings. International 7-9 July 2004 Page(s):193 - 204 Digital Object Identifier 10.1109/IDEAS.2004.1319792

9. Min-Yuh Day; Tzong-Han Tsai; Cheng-Lung Sung; Cheng-Wei Lee; Shih-Hung Wu; Chorng-Shyong Ong; Wen-Lian Hsu. A knowledge-based approach to citation extraction. Information Reuse and Integration, Conf, 2005. IRI -2005 IEEE International Conference on. Aug. 15-17, 2005 Page(s): 50- 55

10. G.G. Chowdhury. Template mining for information extraction from digital documents. Library Trends. 48(1), 1999, 181-207.

11. Masaru T (1987) An efficient augmented-context-free parsing algorithm. Comput. Linguist. 13:31-46

12. Ted B, John C (1993) Generalized probabilistic LR parsing of natural language (Corpora) with unification-based grammars. Comput. Linguist. 19:25-59

13. Inui Kentaro, Virach Sornlertlamvanich, Tanaka Hozumi and Tokunaga Takenobu.1998. Probabilistic GLR parsing: a new formalization and its impact on

parsing performance. In Journal of Natural Language Processing, Vol.5.No.3,pages33-52.

14. Tobias R (2000) A context-sensitive model for probabilistic LR parsing of spoken language with transformation-based postprocessing. In: Proceedings of the 18th conference on Computational linguistics - Volume 2. Association for Computational Linguistics, Saarbrücken, Germany

15. Yong-Jae Kwak, So-Young Park, Young-Sook Hwang, Hoo-Jung Chung, Sang-Zoo Lee and Hae-Chang Rim, "GLR Parser with Conditional Action Model(CAM)," Proc. of the 5th Natural Language Processing Pacific Rim Symposium, pp. 359-366, 2001.

16. Helen M, Po-Chui L, Kui X, Fuliang W (2002) GLR parsing with multiple grammars for natural language queries. ACM Transactions on Asian Language Information Processing (TALIP) 1:123-144

17. GLR Parser with Conditional Action Model using Surface Phrasal Types for Korean Yong-Jae Kwak, So-Young Park, and Hae-Chang Rim NLP Lab., Dept. of CSE, Korea

18. Alon L (1994) An integrated heuristic scheme for partial parse evaluation. In: Proceedings of the 32nd annual meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Las Cruces, New Mexico

19. Embley DW, Jiang Y, Ng YK (1999) Record-boundary discovery in Web documents. In: Proceedings of the 1999 ACM SIGMOD international conference on management of data. ACM Press, Philadelphia, Pennsylvania, United States, Pages: 467 - 478 , ISSN:0163-5808

20. Bing Liu, Robert Grossman, Yanhong Zhai. "Mining Data Records in Web Pages." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2003), Washington, DC, USA, August 24 - 27, 2003.

21. Yanhong Zhai, and Bing Liu. "Web Data Extraction Based on Partial Tree Alignment" To appear in Proceedings of the 14th international World Wide Web conference (WWW-2005), May 10-14, 2005, in Chiba, Japan

22. Bing Liu and Yanhong Zhai. "NET - A System for Extracting Web Data from Flat and Nested Data Records." Proceedings of 6th International Conference on Web Information Systems Engineering (WISE-05), 2005

23. Libx: http://libx.org/

FURL ENTRIES

- *Papers and presentations: OpCit project link*
  Rated: 3 in Citation Senser ; @ Fri, 4 Nov 2005 03:49:38 GMT
- *CPAN link*
  Rated: 3 in Citation Senser ; @ Fri, 4 Nov 2005 03:49:11 GMT
- *SFU Library - Citation Finder link*
  Rated: 3 in Citation Senser ; @ Fri, 4 Nov 2005 03:47:48 GMT
- *WebSPHINX: A Personal, Customizable Web Crawler link*
  Rated: 3 in Citation Senser ; @ Fri, 4 Nov 2005 03:44:57 GMT
- *Statistical Data Mining Tutorials link*
  Rated: 3 in Citation Senser ; @ Fri, 4 Nov 2005 03:43:53 GMT
- *File IO - MozillaZine Knowledge Base link*
  Rated: 3 in Citation Senser ; @ Fri, 4 Nov 2005 03:42:55 GMT
- *Packaging Firefox/Thunderbird Extensions link*
  Rated: 3 in Citation Senser ; @ Fri, 4 Nov 2005 03:41:06 GMT
- *Reference Linking in a Hybrid Library Environment Part 1: Frameworks for Linking link*
  Rated: 3 in Citation Senser ; @ Fri, 4 Nov 2005 03:40:43 GMT
- *CiteULike: A free online service to organise your academic papers link*
  Rated: 4 in Citation Senser ; @ Fri, 4 Nov 2005 03:37:58 GMT
- *AutoBib Project Homepage link*
  Rated: 3 in Citation Senser; @ Fri, 4 Nov 2005 03:36:47 GMT

- *Project related useful web links and bookmarks*

  - http://www.furl.net/members/gprasad6604
  - http://www.furl.net/members/veenabs6604
  - http://www.furl.net/members/ssmenon_6604

- *Project related BLOG's*

  - http://diglib.portspaces.com/gprasad6604/blog3

ACKNOWLEDGEMENTS

We express our sincere gratitude to Dr. Edward Fox for the guidance and suggestions he has provided us during the course of this project. We are fortunate to work with our client, Dr. Godmar Back on his novel idea - Citation Senser as a firefox extension. We express our sincere gratitude to Dr. Godmar Back for guiding us throughout the project and motivating us to explore new ideas and approaches to tackle this problem. And last but not least we like to thank all our classmates for their support and help.

Client Contact Details:

Dr. Godmar Back
Assistant Professor
Dept. of Computer Science, Virginia Tech
gback@cs.vt.edu
Office: 2160A Torgersen Hall
Phone: 540-231-3046