CITATION PARSING IN IDEALS: FALL PRACTICUM 2007

CHAD CURTIS

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
GRADUATE SCHOOL OF LIBRARY AND INFORMATION SCIENCE

Date: December 2007.

Contents

Part 1. Overview and Approaches to Citation Parsing

1. Overview

One obstacle for wider adoption of UIUC's institutional repository, IDEALS, is the submission process. Currently, the user is responsible for entering required submission information into a series of input fields. One step in addressing the amount of work assigned to the submitter is the development of a modular citation parser for IDEALS. The citation parser will allow a user to paste a citation from their curriculum vitae and fields will be populated for them. The addition of this feature will shorten time spent during the submission process, with the ultimate goal to improve a user's experience by lessening his or her workload.

2. Approaches

There are currently two popular approaches to citation parsing: template matching and machine learning. A comparison of the two approaches can be found in an honours thesis, "Citation Parsing Using Maximum Entropy and Repairs" Kiat (2005). In order to achieve a high rate of accuracy Kiat applies the theory of maximum entropy to form a machine learning parser written in Java. Another academic paper on the subject is "Citation Senser: CS6604: Digital Libraries Course Project" Gopal et al. (2005). The task of the project is to add a feature to the LibX browser extension, created by Annette Bailey and Godmar Back (`http://www.libx.org/`). The function of the feature is essentially threefold: a citation is identified in an (X)HTML document, the citation is parsed, and an OpenURL is provided to connect the user to the object being cited. Much of the functionality seems to have been merged into the LibX project. Included in the report is a table comparison of citation parsing methods, which proves valuable when deciding on an approach.

From the sources cited above the advantages and disadvantages of two approaches to citation parsing are summed up in the following:

Template (heuristic) matching

Advantages:

- Reliable with simple or standard citation formats.
- Administrators can create templates to improve accuracy.
- A popular open-source "citation" package is available: (`http://search.cpan.org/~mjewell/Biblio-Citation-Parser-1.10/`)

Disadvantages:

- If there is not a matching template for a citation then parsing accuracy degrades substantially.
- Creation of templates is time consuming.
- Accuracy rate is debated, usually proved to be 65-70%, but some claims are lower.

Machine Learning

Advantages:

- High accuracy rate, cited as 94.20%, Kiat (2005).
- Uses statistical models which handle irregular citations better than heuristics.

- There is an open-source "maximum entropy" package that could be used to build a parser: (`http://maxent.sourceforge.net/`)

Disadvantages:

- Training time is necessary, which can slow down installation and development considerably.
- Complexity of theory (maximum entropy) and practice.
- No open-source "citation" package is in wide use or ready for implementation.

3. Relevant Projects

3.1. **Biblio-Citation-Parser-1.10.** The Perl module Biblio:Citation:Parser is a citation parsing framework available on CPAN: (`http://search.cpan.org/~mjewell/Biblio-Citation-P`
`10/`). One can use the standard package parser or use alternatives, such as Jiao. Templates used to match citations are included and can be edited and contributed back to the project. A popular project using this package is Paracite, a service developed for EPrints: `http://paracite.eprints.org/`.

3.2. **Prototype.** Prototype (`http://www.prototypejs.org/`) is a JavaScript library that abstracts code from a programmer in order to speed up AJAX development with standards-compliant JavaScript. Prototype has appeared in many established projects (`http://www.prototypejs.org/real-world`), but of interest to this project is the use of Prototype in BibApp: (`http://code.google.com/p/bibapp/`).

3.3. **Manakin.** Manakin (`http://di.tamu.edu/projects/xmlui/manakin/`) is a project which started at Texas A&M and will be the default user interface for DSpace 1.5. Previous versions of DSpace used JSP, which made it laborious and time consuming to make customizations that fit the institution. Using Apache Cocoon, Manakin creates an XML UI that essentially works on top of DSpace and is theme-able. Of special interest to institutions and academic departments is the ability to have communities and collections with individual themes.

Part 2. Implementation

After weighing the advantages and disadvantages it was decided to use the Perl modules Biblio-Citation-Parser-1.10 and CGI with the Prototype JavaScript library. The use of a template-based parser meant a quick installation based on tested modules, which could be configured for performance. The justification for not just using just the CGI module and adhoc JavaScript is the possibilty of expansion that Prototype provides. Such expansions could be feedback animation during processing, the ability to hide/reveal fields depending on submission type, and integration into help/error messages that will make a user's experience pleasant. Additionally, Protoype is used in BibApp, which may enhance development in both BibApp and this project.

4. Process Outline

(1) (X)HTML page is created through Apache Cocoon pipelines in Manakin.
(2) User pastes a citation into field and clicks the submit button.
(3) JavaScript function is called on click: parseCitation() within citeParse.js.

(4) parseCitation() forms a new Ajax.Request object instance from the Protoype library, myAjax.
(5) Request is made: a POST of the citation to a Perl script: parse.cgi.
(6) Perl script accepts the input through the use of a CGI object instance, $cgi, and passes the string to a Biblio-Citation-Parser object.
(7) Output is a hash, which is assigned to a variable, $metadata, for further processing.
(8) From the contents of $metadata a comma-delimited string is printed as output.
(9) The response of the Perl output is received by myAjax and the function fillFields() is called.
(10) fillfields() uses the JavaScript split() method to convert the comma-delimited string into an associative array.
(11) Each value in the associative array is matched to the appropriate element id in the (X)HTML page.

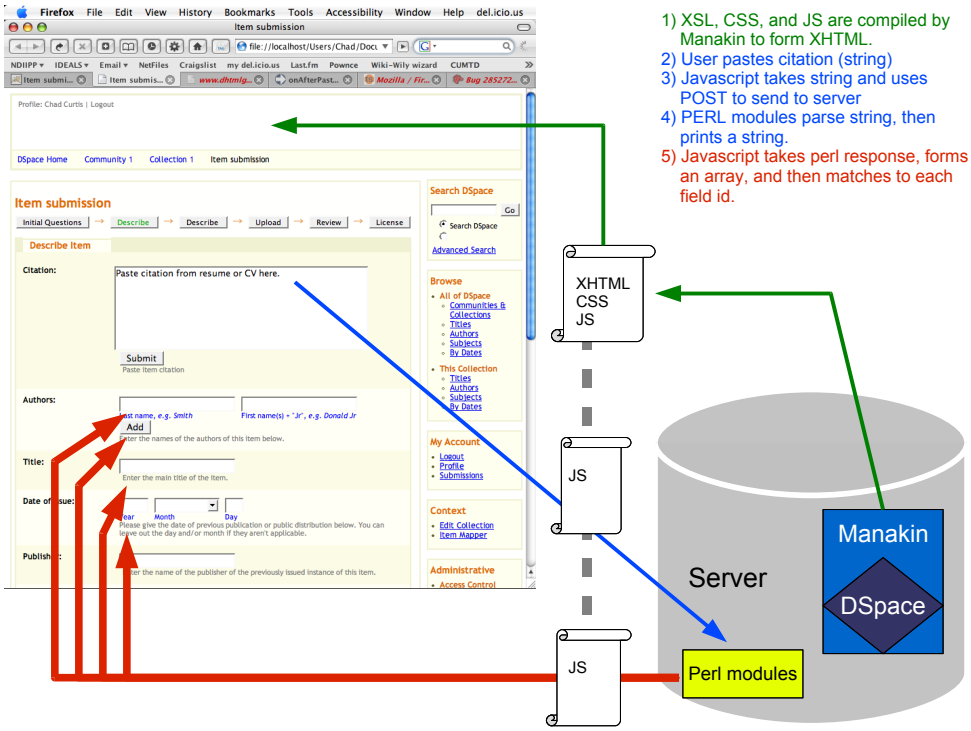

FIGURE 1. Citation Parser Process Overview

5. Static Demo

5.1. **Hosted example.** Copy and paste the following citation into the URL below:

```
Jewell, M (2002) Making Examples for Reference
# Parsers. Journal # of Example Writing 3:100-150.
```

Location of demo:
`http://echodep2.lis.uiuc.edu:8086/mockup/submit.html`

6. Package files

To replicate the static demo the following files are included in the package, static_demo.zip:
XHTML - submit.html
CSS - submit_style.css
JavaScript - prototype.js, citeParse.js
Perl - parse.pl

6.1. **submit.html and submit_style.css.** These are slightly modified files generated by Manakin.

6.2. **citeParse.js.** citeParse.js calls funtions from prototype.js file. As a reminder, to run it locally one must edit server file paths in the JavaScript.

6.3. **parse.cgi.** parse.cgi uses two modules: CGI and Biblio::Citation::Parser::Standard. Ensure that permissions are set as 755 and place this file in a directory of a web server that can execute CGI.

6.4. **Dependencies.**

(1) Perl v5.8.6 or higher
(2) Biblio::Citation::Parser - Follow official INSTALL file instructions included in the CPAN package. Dependencies for this module will vary, but most environments will only need the ones mentioned in documentation:
Text Uni
URI

7. Considerations, Issues, and Modifications

The intention of this demo is for the functionality to be added to Manakin's XSLT process. The forthcoming documentation for the dynamic module will be on the project wiki:
`https://services.ideals.uiuc.edu/wiki/bin/view/IDEALS/Internal/ChadCurtis`

There are currrently two citation input fields in the static demo. The current solution is that any citation pasted into the first field is put copied into the second field below. The dynamic module may want to use only one citation input field.

Code needs to be modified to fill in the forms in a smarter manner. Most likely the Perl templates need to be edited more than JavaScript. For example, pasting the following citation, puts the last name, Huber, in the title field:

```
Michael E. Huber. Marine Biology. 4th ed. Boston: McGraw-Hill, 2003.
```

8. Conclusion

Using existing Perl modules and JavaScript libraries, a citation parsing feature can be added to the IDEALS DSpace Submission process. The implementation of a template-based citation parser enables the administrator to get the feature installed quickly and allows future modification. The static demo demonstrated in this paper can be integrated into a dynamic module that works with the existing pipeline process in Manakin.

References

Gopal, P., Menon, S., and Basavaraj, V. (2005). Citation senser cs6604: Digital libraries course project. Technical report, Virginia Tech.

Kiat, N. Y. (2005). Citation parsing using maximum entropy and repairs. Technical report, National University of Singapore.