# Enriching the VT ETD-db System with References

**Sung Hee Park**
Department of Computer Science
Virginia Tech
shpark@cs.vt.edu

**Edward A. Fox**
Department of Computer Science
Virginia Tech
fox@cs.vt.edu

**ABSTRACT**

This project aims to enrich the Virginia Tech's ETD-db digital library system to handle references found in ETDs, typically in the final References section, but sometimes as footnotes or end of chapter references. One objective is to extend ETD-MS so that references can be included in the metadata. Another objective is to have automatic methods to extract these references from ETDs. A third objective is to manage the references inside ETD-db, providing browse, search, and presentation services. We first extract the reference section (or other portions with references) from an ETD, store the references into the metadata database for ETDs, and show the reference information in the reference tab of the ETD "splash" page, pulling the reference information from the metadata database. Since reference sections can be located at each chapter, or as footnotes, as well as at the end of theses, new methods for intelligent and reliable reference extraction are being devised. These make use of machine learning, heuristics, knowledge bases, and text processing techniques. As a result of this project, software and scripts for reference section extraction will be available. Metadata records with reference information included in the metadata are generated, and represented according to the extended version of ETD-MS. Those in the scholarly community who use ETDs could have an easier time accessing and giving credit to prior research publications.

**Keywords**

ETD-MS, Reference section extraction, machine learning

**INTRODUCTION**

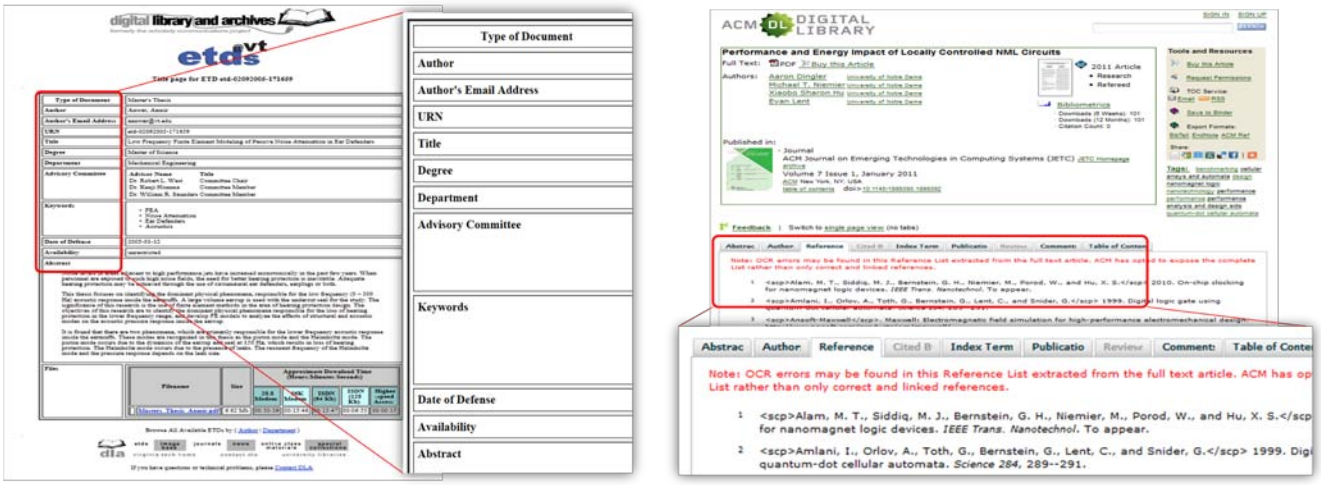
A thesis or dissertation is one of the scholarly works that shows a partial fulfillment of requirements of a degree that a student pursues in higher education. Virginia Tech has been a leader in electronic thesis and dissertation (ETD) initiatives since 1987. As a result of collecting finished theses and dissertations from graduate students and doctoral candidates, the collection has grown to exceed 18,000 manuscripts[1]. ETDs are described, managed, published, and searched using several types of metadata, e.g., descriptive metadata (including bibliographic information), administrative metadata, technical metadata, etc.

To facilitate our research on extending use of the ETD database, the reference sections of manuscripts need to be extracted and included as part of the browsing page for each ETD. Extraction by hand would require countless hours of work, so automation is required. In scholarly works (e.g., journal articles, conference papers, and technical reports), "reference" may refer to a section of document specifying resources referred in the text. A "References" section is included in the ETDs for description of relations in the text between other scholarly works. Regarding presenting such data and relations, we were inspired by the ACM Digital Library (DL). Figure 1 shows the VT ETD "splash" page vs. the ACM Digital Library "reference tab". The former has no references as metadata, while the ACM DL enables users to browse references.

There can be three types of reference in the text: references at the end of the document, chapter references, and footnotes. References should be extracted by some reference extraction techniques (e.g., regular expression, rule based approach, or machine learning approach) to parse and analyze them. Reference section extraction can be considered as an information extraction or document segmentation problem. To solve this problem, classification techniques, commonly used in pattern recognition and data mining, can be exploited.

Typically, brute force techniques using regular expressions have been found to be inadequate for this task because of the various different types of references that can be found in a thesis or dissertation. Therefore, we adopt machine learning techniques to improve the efficiency and accuracy of reference extraction over naïve methods. The technique we have created is able to robustly extract reference sections from ETDs.

[1] ETD collection: Browse Available ETDs by Author, http://scholar.lib.vt.edu/theses/browse/by_author/index.html




**(a) VT ETD-db "splash" page** **(b) ACM DL reference tab**

**Figure 1. (a) VT ETD-db "splash page" vs. (b) ACM DL reference tab**

### Research Questions

The above-mentioned problems lead to the following research questions:

- How can we implement metadata schema for bibliographic information?
- What machine learning methods are effective to extract reference sections including footnotes and chapter references?

This paper is organized as follows. Section 2 describes related work; Section 3 proposes an extension to ETD MS to include bibliographic information; Section 4 explains automatic reference section extraction; Section 5 covers evaluation of the extended ETD-MS and the automatic extraction; Sections 6 and 7 include discussion, conclusions, and future work.

## RELATED WORK

In this section, we review previous work on metadata schema for references, and information extraction from documents.

### Reference section extraction

General information extraction (IE) techniques from documents have been widely researched.

Naomi Sager directed an early IE system in the Linguistic String Project, focused on the medical domain (Sager, 1981). The Message Understanding Conference (MUC), sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA), encouraged IE research from 1987 to 1998 (Grishman & Sundheim, 1996). At MUC-7, there was evaluation of extraction of useful information from news messages about *Airplane crashes and Rocket/Missile Launches*. MUC encouraged a focus on extracting four elements: named entities, co-references, template elements, and template relations.

Subsequently, the Automatic Content Extraction (ACE) evaluation project was organized by the National Institute of Standards and Technology (NIST) from 2000 to 2008. An aim of the ACE program was development of technologies that extract entities from language data and then infer relations among them. Those two programs, MUC and ACE, contributed to the development of a variety of indicators for deep evaluation.

Regarding metadata extraction, Han et al. proposed automatic document metadata extraction using support vector machines, a machine learning technique . In addition, Parscit is an open source package which enables users to extract reference strings from a document and then parse them. It is currently being used in CiteSeerX, a citation-oriented digital library for computing. It is based on some heuristics, e.g., using regular expressions like '/[R|r][eferences]/' or '/[B|b][ibliography]/'.

### Reference Metadata Schema

Recently, some efforts have been made to provide metadata schema for scholarly works (e.g., journal articles, conference papers, technical reports) including ETDs. Hence, we review: 1) Dublin Core (DC) Metadata Element Set, 2) Qualified DC Terms, 3) MODS, 4) ETD MS, and 5) TDL MODS.

*General Metadata Schema*

**Dublin Core Metadata Element Set**[2]: The Dublin Core Metadata Element Set is a vocabulary for describing an arbitrary digital object. It is often called 'Simple DC' and consists of fifteen primitive properties. Dublin Core is a practical and popular metadata schema which most repositories are adopting. In Simple DC, the `dc:relation` property can be used to describe references.

**Qualified DC Terms**[3]: Qualified DC terms include a variety of properties describing bibliographic information. In contrast to simple DC, these terms are in the */terms/* namespace. There are two terms related to *references*: `dcterms:references` and `dcterms::isReferencedBy`. The former describes a related resource (literally, 'References') that is referred to by the described resource, whereas the latter describes a related resource (citations) that references the described resource.

**Metadata Object Description Schema (MODS)**: MODS is a metadata standard for describing an arbitrary object, adopted by Library of Congress. It consists of twenty *top level elements*. `relatedItem` can be used for describing a reference. For more details, see the official web site[4].

*Metadata Schema Dedicated to ETDs*

**ETD MS (Metadata Standard)**: ETD MS is a standardized metadata schema for interoperability, dedicated to describing ETDs. Most of the properties in ETD MS are found in the Dublin Core Element Sets.

**TDL MODS**: The Texas Digital Library (TDL) developed another metadata schema for describing ETDs. TDL MODS is adopting MODS.

## EXTENSION OF ETD MS TO INCLUDE REFERENCE INFORMATION

To extend ETD MS to include reference information, we considered existing solutions. As shown in Table 1, Simple DC, DC Terms, and MODS provide elements for describing references. We use `dc:relation` and `dcterms:references` for our study. In addition, *Guidelines for Encoding Bibliographic Citation Information in Dublin Core* (Apps, 2005) describe helpful recommendations to include a reference into a document.

| DC | DC Terms | MODS | Extended ETD-MS | TDL ETD MODS |
|---|---|---|---|---|
| `dc.relation.references` | `dcterms:references` | `mods:relatedItem` | **`dc:relation`** **`dcterms:references`** | N/A |

Table 1. Properties for Describing Reference in Existing Metadata Schemas

### Implementation

Metadata for references can be implemented with current description languages such as HTML/XHTML, XML, and RDF.

**HTML/XHTML**: DC Metadata Initiative provides some guidelines for expressing DC metadata with (X)HTML (Johnston & Powell, 2008). When metadata for references is implemented in (X)HTML, it can be represented using *link* and *meta* tags. The former has either URL or references relative to a base URI as an attribute; the latter has the content of metadata as an attribute. An example showing how the reference metadata is implemented with HTML is described in Table 2. They have either a human readable (e.g., a plain text) or a machine readable form (e.g., OpenURL ContextObject (OCLC, 2009)).

**XML**: DC metadata can be implemented with XML. In contrast to (X)HTML, XML representation expresses reference metadata using the value of metadata property/elements/tags. The *Guideline for Dublin Core Implementation with XML* (Powell, 2003) provides some recommendations regarding implementing with XML. An example implemented with XML is compared to that of HTML in Table 2. This XML representation including metadata for references can be used in OAI-PMH, a protocol for interoperable metadata harvesting, to harvest/provide metadata for 'references' from/for other system.

**RDF**: DC metadata also can be implemented with RDF. The *Guideline for Dublin Core Implementation with RDF* (Nilsson, Powell, Johnston, & Naeve, 2008) describes how constructs and vocabularies used in DC metadata are organized and expressed with *DC Abstract Model (DCAM)* (Powell, Nilsson, Naeve, Johnston, & Baker, 2007), a RDF conceptual model, which builds on RDF undertaken by W3C. DCAM, in turn, defines the nature of component used and expresses how for the components to be combined to create information structures. Examples of the RDF representation are an *application profile*

[2] Dublin Core Metadata Elements Set, Version 1.1, http://dublincore.org/documents/dces
[3] DCMI Metadata Terms, http://dublincore.org/documents/dcmi-terms/
[4] Metadata Object Description Schema (MODS), http://www.loc.gov/standards/mods/mods-outline.html

for DC metadata or a *package* including resources and links to the resources (e.g., OAI-ORE[5]).

### Application Profile

An application profile is a set of metadata elements, properties, vocabularies, terms, and guidelines defined for a specific application. *Dublin Core Application Profile (DCAP)* provides guidelines for use of DC metadata in a specific context (Coyle, 2009). Especially, *Scholarly Work Application Profile (SWAP)*, a Dublin Core application profile for scholarly works, has been defined by a research group (Allinson, Johnston, & Powell, 2007). We describe an application profile based on SWAP to meet our functional requirements to support browsing, searching, and presentation services as well as providing metadata as well as contents of references for other scholarly repositories. Open Archive Initiative-Object Reuse and Exchange (OAI-ORE) is a standard for describing the exchange of aggregations of Web resources (Lagoze et al., 2008).

### Examples

Here is an example of a VT ETD.[6] Bibliographic reference information in the Extended ETD-MS can be included as a *plain text citation* and *OpenURL ContextObject* encoded in both XML and (X)HTML. Open URL ContextObject provides a standardized format for describing Uniform Resource Locators (URLs) so that Internet users easily can find a copy of a resource that they are allowed to access (OCLC, 2009). Table 2 and 3 show an example of ETD MS with a reference and an example of extended ETD MS in XML and (X)HTML formats, respectively.

| Property | Syntax Encoding Scheme URI | Value String |
|---|---|---|
| dc:title | | Low Frequency Finite Element Modeling of Passive Noise Attenuation in Ear Defenders |
| dc:creator | | Aamir Anwar |
| dc:contributor | | Mechanical Engineering, Virginia Tech |
| dc: publisher | | Virginia Tech |
| **dcterms: references** | | **L.E. Kinsler, A.R. Frey, A.B. Coppens, J.V. Sanders, Fundamentals of Acoustics, 4 th ed., John Wiley & Sons Inc. New York, 2000.** |
| **dcterms: references** | **Info:ofi/fmt: kev:mtx:ctx** | **&ctx_ver=Z39.88-2004& rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook<br>&rfr_id=info%3Asid%2Focoins.info%3Agenerator&rft.genre=book&rft.btitle=Fundamentals+of+Acoustics<br>&rft.title=Fundamentals+of+Acoustics&rft.aulast=Kinsler<br>&rft.aufirst=L.+&rft.auinit=L.E.K.&rft.aucorp=Frey+A.R.<br>&rft.au=L.++L.E.K.+Kinsler&rft.au=Coppens+A.B.<br>&rft.au=Sanders+J.V.+&rft.date=2000&rft.pub=John+Wiley+%26+Sons+Inc.<br>&rft.place=New+York&rft.edition=4+th+ed.** |

**Table 2. An Example of ETD MS with References**

| | Reference to a Book Encoded in XML | Reference to a Book Encoded in (X)HTML |
|---|---|---|
| Schema declaration | <?xml versino="1.0" encoding="UTF-8"?><br><thesis xmlns = http://www.ndltd.org/standards/metadata/etdms/1.0/<br>xmlns:dcterms = http://purl.org/dc/terms/ xsi:schemaLocation =<br>"http://www.ndltd.org/startds/metdata/etdms/1.0/<br>http://www.ndltd.org/standards/metadata/etdms/1.0/etdms.xsd"> | <link rel="schema.etdms" href =<br>"http://www.ndltd.org/standards/metadata/etdms/1.0/" /><br><link rel="schema.dcterms" href="http://purl.org/dc/terms/" /><br><link rel="schema.KEV" href="info:ofi/fmt:kev:mtx:" /> |
| Title, | <title>Low Frequency Finite Element Modeling of Passive Noise Attenuation in Ear Defenders</title> | <meta name="etdms.Title" content="Low Frequency Finite Element Modeling of Passive Noise Attenuation in Ear Defenders"/> |
| Author, etc. | <!– Below is ETD-MS v.1.0 metadata --><br>... | <!– Below is traditional ETD-MS metadata --><br>... |
| **A single reference** | **<!– The reference is described --><br><dcterms:references id="1">L.E. Kinsler, A.R. Frey, A.B. Coppens, J.V. Sanders, Fundamentals of Acoustics, 4 th ed., John Wiley & Sons Inc. New York, 2000. </dcterms:references><br><dcterms:references id="1" scheme="KEV.ctx" > ctx_ver=Z39.88-2004<br>&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook<br>&rfr_id=info%3Asid%2Focoins.info%3Agenerator&rft.genre=book&rft.btitle=Fundamentals+of+Acoustics<br>&rft.title=Fundamentals+of+Acoustics&rft.aulast=Kinsler<br>&rft.aufirst=L.+&rft.auinit=L.E.K.&rft.aucorp=Frey+A.R.<br>&rft.au=L.++L.E.K.+Kinsler&rft.au=Coppens+A.B.<br>&rft.au=Sanders+J.V.+&rft.date=2000&rft.pub=John+Wiley+%26+Sons+Inc.<br>&rft.place=New+York&rft.edition=4+th+ed. </dcterms:references>** | **<!– The first reference is described --><br><meta name="dcterms.references" id="1" content="L.E. Kinsler, A.R. Frey, A.B. Coppens, J.V. Sanders, Fundamentals of Acoustics, 4 th ed., John Wiley & Sons Inc. New York, 2000."/><br><meta name="dcterms.references" scheme="KEV.ctx" id="1"<br>content="ctx_ver=Z39.88-2004<br>&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook<br>&rfr_id=info%3Asid%2Focoins.info%3Agenerator<br>&rft.genre=book&rft.btitle=Fundamentals+of+Acoustics<br>&rft.title=Fundamentals+of+Acoustics&rft.aulast=Kinsler<br>&rft.aufirst=L.+&rft.auinit=L.E.K.&rft.aucorp=Frey+A.R.<br>&rft.au=L.++L.E.K.+Kinsler&rft.au=Coppens+A.B.<br>&rft.au=Sanders+J.V.+&rft.date=2000&rft.pub=John+Wiley+%26+Sons+Inc.<br>&rft.place=New+York&rft.edition=4+th+ed."/>** |
| Rest of references | <!– The rest of references are described--><br>... </thesis> | <!– The rest of references are described--> |

**Table 3. Example of Extended ETD MS in XML and (X)HTML**

[Note: line breaks within 'content' are for presentation only. The ContextObject should be a single unbroken line.]

[5] Open Archives Initiative Object Reuse and Exchange (OAI-ORE), http://www.openarchives.org/ore/1.0/primer
[6] Title page for ETD etd-12012004-170927, http://scholar.lib.vt.edu/theses/available/etd-12012004-170927/

Table 4 exemplifies a usage of reference metadata in SWAP and OAI-ORE. Notice that bold parts describe references.

**Example of SWAP**

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix eprints: <http://purl.org/eprint/terms/> .
@prefix etdms: <http://www.ndltd.org/etdms/terms/> .
DescriptionSet{
  Description {
    Resource URI (<http://parsifal.dlib.vt.edu:3001/browse/etd-02092005-171659>
      Statement {
           Property URI { dc:type }
           Value URI ( <http://purl.org/eprint/entityType/ScholarlyWork> )
      }
      Statement {
           Property URI { dc:title }
       Literal Value String("Low Frequency Finite Element Modeling of Passive Noise Attenuation in Ear Defenders")
      }
      # Basic Metadata (e.g., authors,  keywords, department,  existing in ETD MS
           ...
      Statement (
           Property URI ( dcterms:references )
           Value String ( "L.E. Kinsler, A.R. Frey, A.B. Coppens, J.V. Sanders,
                     Fundamentals of Acoustics, 4 th ed., John Wiley & Sons Inc. New York, 2000." )
           Value String("&ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook
                    &rfr_id=info%3Asid%2Focoins.info%3Agenerator&rft.genre=book&rft.btitle=Fundamentals+of+Acoustics
                    &rft.title=Fundamentals+of+Acoustics&rft.aulast=Kinsler&rft.aufirst=L.+&rft.auinit=L.E.K.
                &rft.aucorp=Frey+A.R.&rft.au=L.++L.E.K.+Kinsler&rft.au=Coppens+A.B.&rft.au=Sanders+J.V.+&rft.date=2000
                  &rft.pub=John+Wiley+%26+Sons+Inc.&rft.place=New+York&rft.edition=4+th+ed.")
        Syntax Encoding Scheme URI ( kev:ctx )
      )
           ...
    Statement {
     Property URI ( eprint:isExpressedAs)
     ValueURI(<http://scholar.lib.vt.edu/theses/available/etd-02092005-171659/unrestricted/Masters_Thesis_Aamir.pdf>)
    }
  }
  Description {
    Resource URI(<http://scholar.lib.vt.edu/theses/available/etd-02092005-171659/unrestricted/MastersThesisAamir.pdf>)
                    ...
```

**Example of OAI-ORE**

```
<?xml version='1.0' encoding='unicode' ?>
<rdf:RDF xmlns:ore="http://www.openarchives.org/ore/terms/"
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:dcterms="http://purl.org/dc/terms/"
          xmlns:foaf="http://xmlns.com/foaf/0.1/"
          xmlns:dc="http://purl.org/dc/elements/1.1/">
          <rdf:Description rdf:about="http://parsifal.dlib.vt.edu:3001/rem/ref/etd-02092005-171659">
                    <ore:describes rdf:resource="http://parsifal.dlib.vt.edu:3001/rem/ref/etd-02092005-171659" />
                    <dcterms:creator rdf:parseType="Resource">
                              <foaf:name>Sung Hee Park</foaf:name>
                              <foaf:page rdf:resource="http://scholar.lib.vt.edu/" />
                    </dcterms:creator>
                    <dcterms:created rdf:dataType="http://www.w3.org/2001/XMLSchema#dateTime"> 2005-02-09T17:16:59
               </dcterms:created>
                    <dc:rights>This Resource Map is available under the Creative Commons Attribution-Noncommercial
                     2.5 Generic license</dc:rights>
                    <dcterms:rights rdf:resource="http://creativecommons.org/licenses/by-nc/2.5/" />
          </rdf:Description>
          <rdf:Description rdf:about="http://parsifal.dlib.vt.edu:3001/browse/etd-02092005-171659">
                    <ore:isDescribedBy rdf:resource="http://parsifal.dlib.vt.edu:3001/browse/etd-02092005-171659" />
                    <dc:title>ETD with References</dc:title>
                    <dcterms:creator rdf:parseType="Resource">
                              <foaf:name>Anwar, Aamir</foaf:name>
                              <foaf:mbox rdf:resource="aanwar@vt.edu" />
                    </dcterms:creator>
                    <ore:aggregates rdf:resource="Human Start Page Link" />
                    <ore:aggregates rdf:resource="PDF Link" />
                    <dcterms:references rdf:resource="Reference_1" />
                    ...
                    <dcterms:references rdf:resource="Reference_n" />
                    <rdf:type rdf:resource="Link to Type of Aggregation" />
                    <ore:aggregates rdf:resource="Reference_1" />
                    ...
          </rdf:Description>
          ...
          <rdf:Description rdf:about="http://addison.vt.edu/record=b2077343">
                    <dc:title>Fundamentals of acoustics</dc:title>
                    <dc:language>en</dc:language>
          </rdf:Description>
          ...
</rdf:RDF>
```

**Table 4. Example of Scholarly Works Application Profile (top) and Example of OAI-ORE (bottom)**

## AUTOMATIC REFERENCE SECTION EXTRACTION

Automatic reference section extraction is a module of the ETD-db system for exposing references to the public. Figure 2 illustrates its system architecture. The automatic reference section extraction consists of the following: pdf2txt, feature extraction, learning (training), and reference section extraction (see Figure 3).

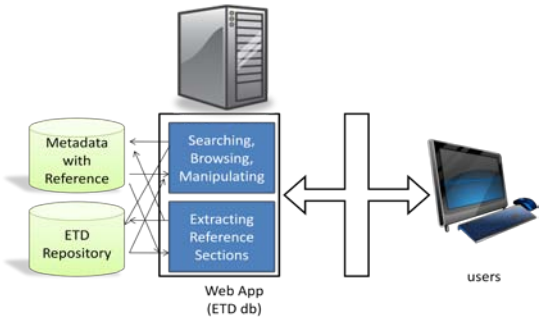



**Figure 2. System Architecture**

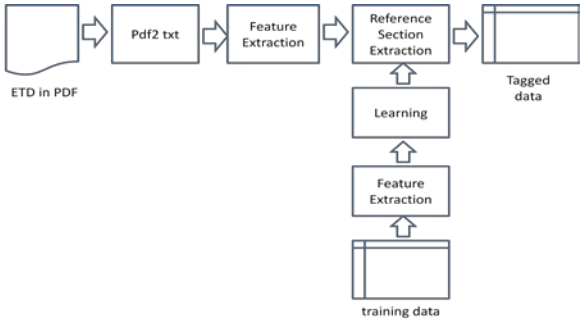



**Figure 3. Dataflow Diagram of reference section extraction**

### PDF2Txt

The Apache PDFBox API is an open source, Java-based library that supplies components for working on and manipulating PDF files. In the context of our project, the operations required dealing with stripping content from a PDF document and writing it to a text file. The version used in this project was 1.4.0.

### Feature Extraction

Features that we use can be categorized into three types: Word local features, line features, and contextual features. Table 5 describes each feature with simple examples.

28 different string patterns (e.g., types of punctuation, capitalization, etc.) are used as word features. The token vector is given a bit string for each pattern it does or does not have (e.g., 0,1,0,0,1,0,0,0,…..,0,0,1,0). Each feature vector is based on a line of text with a token vector for the tokens it has and a null vector for all of the tokens it lacks. For training data, each line is labeled as 'REF' if it is a *reference*, as 'NON-REF' otherwise.

| Feature Name | Descriptions | Examples |
|---|---|---|
| Word local features | 28 different string patterns | Types of punctuation, capitalization, etc. |
| Line feature | Patterns in a line | Number of word in the line, percentage of capitalized words |
| Contextual feature | Patterns of a neighborhood | Class ('REF' or 'NON-REF') of neighbor lines before and after the current line |

**Table 5. Feature sets**

### Training

After extracting features, a machine learning based classifier is trained to determine reference sections from texts. It is important to create training sets for WEKA (Waikato Environment for Knowledge Analysis). First, some lines from a reference section and from the body of the text are extracted to a separate file. For training, features also should to be extracted. The feature vectors are then created to measure the similarity between each line on the vector space. In particular, there exists a class tag (i.e., 'REF' or 'NON-REF') indicating if that line is a reference or not, as an attribute of each vector.

### Reference section extraction (Classification)

In this way, the program can "learn" over time various patterns that references follow. Once a classifier object is trained, any feature vector can be tested against previous classifications to see if it "matches" any previous patterns. To implement this, WEKA, open source machine learning software, is used to deal specifically with data mining operations. The operations provided by WEKA are crucial to the machine-learning techniques employed by our software. The version used in this project was 3.7.3.

Figure 4 illustrates VT ETD-db with Reference Metadata. 'References' after an 'Abstract' are included to show the users

which references the ETD refers to.

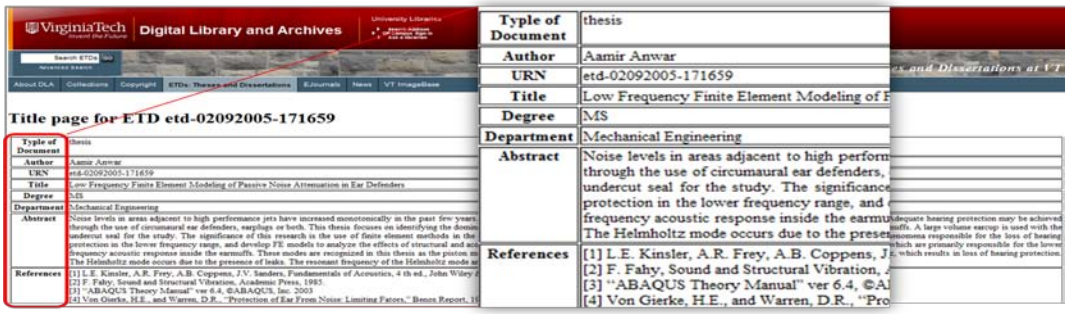



**Figure 4. VT ETD-db with Reference Metadata**

## EVALUATION

We evaluated our machine learning approach to reference section extraction. We used six documents randomly selected from the VT-ETD db system, and marked their reference sections and non-reference sections, manually. Table 6 shows the statistics of documents used in this evaluation. Incidentally, all reference sections were found at the end of documents.

| Items | Document1 | Document2 | Document3 | Document4 | Document5 | Document6 |
|---|---|---|---|---|---|---|
| **# of lines** | 4,818 | 4,899 | 2,237 | 6,178 | 2,369 | 2,254 |
| **# of reference lines (location)** | 324 (end) | 291 (end) | 63 (end) | 214 (end) | 145 (end) | 73 (end) |
| **Percentage of reference lines** | 6.7% | 5.9% | 2.8% | 3.5% | 6.1% | 3.2% |
| **# of features** | 5,185 | 5,493 | 3,208 | 6,061 | 3,393 | 4,097 |

**Table 6. Data, Used in Evaluation, Randomly Sampled.**

We evaluated two tokenization methods: Support Vector Machines (i.e., with a normal tokenizer and a simple tokenizer) and one existing method, ParsCit. Our normal tokenizer considers delimiters (space, tab, carriage return, period (.), comma (,), semicolon (;), colon (:), single quotation mark ('), double quotation mark ("), parentheses, and question mark (?)). Our simple tokenizer drops period, comma, semicolon, colon, double quotation mark, and parentheses, as compared with the normal tokenizer. ParsCit is based on heuristics using regular expressions. Table 7 shows precision, recall, and F1-score of these three tokenizer methods of interest, run against six test datasets.

| | Doc1 | | | Doc2 | | | Doc3 | | | Doc4 | | | Doc5 | | | Doc6 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Pre.** | **Rec.** | **F1** | **Pre.** | **Rec.** | **F1** | **Pre.** | **Rec.** | **F1** | **Pre.** | **Rec.** | **F1** | **Pre.** | **Rec.** | **F1** | **Pre.** | **Rec.** | **F1** |
| **Normal +SVM** | 0.99 | 0.80 | 0.88 | 0.97 | 0.75 | 0.85 | 0.92 | 0.54 | 0.68 | 1.00 | 0.83 | 0.91 | 0.86 | 0.65 | 0.74 | 0.83 | 0.35 | 0.49 |
| **Simple +SVM** | 1.00 | 0.74 | 0.85 | 0.97 | 0.66 | 0.78 | 0.92 | 0.38 | 0.54 | 0.99 | 0.67 | 0.80 | 0.89 | 0.51 | 0.65 | 0.95 | 0.26 | 0.41 |
| **ParsCit** | 1.00 | 1.00 | **1.00** | 1.00 | 1.00 | **1.00** | 1.00 | 1.00 | **1.00** | 1.00 | 1.00 | **1.00** | 0.67 | 1.00 | **0.80** | 1.00 | 1.00 | **1.00** |

**Table 7. Result of reference section extraction (Recall, precision)**

When we informally did experiments on chapter reference section extraction through picking some documents with chapter references, ParsCit failed saying "Citation text cannot be found: ignoring". The ETD selected used "Literature Cited" as the reference section header in each chapter. ParsCit does probably not include "Literature Cited" as a starting word of a reference section. Even though we did an experiment with chapter reference sections starting with 'References', ParsCit extracted only the references in the last chapter; it also had some extraction errors which failed to find the end of reference section. When we did an experiment considering contextual features, against document 6 (which showed the worst performance), the performance was improved, resulting in: precision, 0.973; recall, 0.986; F1, 0.979.

## DISCUSSION

One of the problems that we encountered was in creating feature vectors. We wanted to include both patterns in each token, and the tokens themselves, to help identify which lines were references. Initially we created a string with a 0 or 1 for all of the 28 patterns (separated by commas), for each token. The feature vector would be a combination of the pattern strings for each token it had, and null pattern strings (all zeros) for all of the tokens it did not have. This process created incredibly large

.CSV files. The long feature vectors would take up too much memory and crash the Java virtual machine.

To solve the problem with incorrect formatting, we had to go into our reference extraction program and generate a file in a format compatible to WEKA (an .arff file) which requires attribute cells (e.g., '@attribute A_CAPS [0,1]', where 'A' is a word; 'CAPS' is a pattern) before feature vectors (e.g.,'1 0 0 0 0 1 …') to place. This is an extremely long and time consuming step, as an attribute cell must be added for every pattern (e.g., CAPS, NUMERIC, ONLYLETTERS) times the number of words (e.g., 'A', 'Abraham', …) in the dictionary (i.e., @attribute A_CAPS [0,1], @attibute A_NUMERIC [0,1] …, @attribute A_LETTER_SONLY [0,1], @attribute Abraham_CAPS [0,1], and so on).

To solve the problem of incredibly long feature vectors, we reduced the size of the string created for each token. We treated each of the zeros and ones in the pattern string as a bit in an integer. This reduced the pattern string to a single number. This procedure is also to keep each string pattern combination unique. In the end, these integer values have been transformed into bit strings by a filter *NumericToBinary* in the WEKA tool.

## CONCLUSION & FUTURE WORK

In this paper, our main contributions are to enable users to easily access reference information without opening the digital content file, usually, stored in PDF format. Moreover, integrating the automatic reference metadata, the digital library will be able to provide bibliographical information in a specific way, i.e., COinS (Context Object in Span) corresponding to each ETD and its references for a crawler (e.g., Zotero, a Firefox plug in) with more plentiful metadata (ETD-MS). Especially, we focused on references/bibliography information in a pilot project carried out by a group in a Virginia Tech Computer Science course (CS4624). We stored a new reference metadata item into an ETD metadata DB, in which the current ETD-db system does not provide reference metadata. Machine learning techniques not only show great potential for reference extraction, but also can be applied to extracting specific data from references.

## ACKNOWLEDGMENTS

We thank Kenneth Lee, Sean Springa, and Adam Larson, senior students of the Department of Computer Science at Virginia Tech, for conducting a class project in the Spring 2011 CS4624 course. We also give thanks to Collin Brittle and Kimberli Weeks, members of the staff of the Digital Library and Archive Department at Virginia Tech, for constructive comments on metadata schema implementation for references.

## REFERENCES

Allinson, J., Johnston, P., & Powell, A. (2007). A Dublin Core Application Profile for Scholarly Works. *ARIADNE*, (50). Retrieved from http://www.ariadne.ac.uk/issue50/allinson-et-al/

Apps, A. (2005). Guidelines for Encoding Bibliographic Citation Information in Dublin Core. from http://dublincore.org/documents/dc-citation-guidelines/index.shtml

Coyle, K. (2009). Guidelines for Dublin Core Application Profiles. from http://dublincore.org/documents/profile-guidelines/index.shtml

Grishman, R., & Sundheim, B. (1996). *Message Understanding Conference -6: A Brief History*. Paper presented at the 16th International Conference on Computational Linguistics (COLING).

Johnston, P., & Powell, A. (2008). Expressing Dublin Core metadata using HTML/XHTML meta and link elements. from http://dublincore.org/documents/dc-html/

Lagoze, C., Sompel, H. V. d., Johnston, P., Nelson, M., Sanderson, R., & Warner, S. (2008). Open Archives Initiative Object Reuse and Exchange. from http://www.openarchives.org/ore/1.0/toc.html

Nilsson, M., Powell, K., Johnston, P., & Naeve, A. (2008). Expressing Dublin Core metadata using the Resource Description Framework (RDF). from http://dublincore.org/documents/dc-rdf/

OCLC. (2009). OpenURL. from http://www.oclc.org/research/activities/openurl/default.htm

Powell, A. (2003). Guidelines for implementing Dublin Core in XML. from http://dublincore.org/documents/dc-xml-guidelines/

Powell, A., Nilsson, M., Naeve, A., Johnston, P., & Baker, T. (2007). DCMI Abstract Model.

Sager, N. (1981). *Natural Language Informatin Processing: A Computer Grammar of English and its Applications*. Reading, Massachusetts: Addison Wesley.