The Role of the HDDI™ Collection Builder in Hierarchical Distributed Dynamic Indexing

Russell H. Bader, Miranda R. Callahan, Douglas A. Grim,
John T. Krause, and William M. Pottenger

Russell H. Bader, Miranda R. Callahan, Douglas A. Grim and John T. Krause
National Center for Supercomputing Applications
Champaign, IL 61820

William M. Pottenger, Ph.D.
Lehigh University
Bethlehem, PA 18015
billp@eecs.lehigh.edu

Keywords: data mining, information retrieval, machine learning, computational knowledge management, parsing, parts of speech tagging, noun phrase extraction, HDDI™

Abstract

The global growth in popularity of the World Wide Web has been enabled in part by the availability of browser-based search tools which in turn have led to increased demand for advances in the field of textual data mining.

Simultaneously, fully automatic content-based techniques of textual data management have been under development at a number of institutions. The time is thus ripe for the development of scalable knowledge management systems capable of handling extremely large and diverse textual collections distributed across multiple repositories.

This paper introduces the HDDIÔ Collection Builder, our custom developed solution for automatic feature extraction from such œollections. The HDDIÔ Collection Builder provides input features for several algorithms in distributed textual data mining.

1 Introduction

The explosive growth of digital repositories of information has been enabled by recent developments in communication and information technologies. Despite significant accomplishments in internetworking, however, scalable mining techniques for distributed data lag behind the rapid growth of digital collections.

In the 21st century, a significant amount of scientifically valuable information will be available via such computer communications networks. The appearance of focused digital libraries on the World Wide Web demonstrates the willingness of scientists and engineers to distribute detailed information beyond that traditionally available in the published literature (e.g., [2]). If this volume of literature, unprecedented in its scope, is to be useful to the scientific community, it is critical that new information infrastructure be developed which enables effective management of the huge volume of distributed data emerging in digital form.

Traditional methods of textual indexing combine multiple subject areas into a single, monolithic index. There are already enough documents on the Web that such mining technology often fails to perform effective search. The difficulty lies in the fact that since so many documents and subjects are being combined together, retrieving all the documents that match a particular word phrase often returns too many documents for effective search. This problem has been known for some time [3]. In order to properly address this problem, a paradigm shift is needed in the approach to indexing.

First and foremost, it is clear that digital collections are now and will continue to be distributed. Our first premise is thus that indexes, or models, in a distributed data mining system must also be distributed[1].

Second, it must be realized that the textual data contained in these distributed digital repositories is hierarchical in nature. Traditionally, knowledge hierarchies, or ontologies, have been created with human expertise[2]. Such an approach does not scale to the tremendous amount of emerging digital data for two reasons: first, as knowledge increases, new topics

[1] Note that we use the terms *model* and *index* interchangeably throughout this paper.

[2] One popular form is the thesaurus (e.g., the National Library of Medicine's MeSH thesaurus).

are emerging at a greater rate; second, the speed of emergence and the sheer volume of data preclude manual approaches to model building. Our second premise is thus that models of distributed textual data must be automatically generable while at the same time properly reflecting the hierarchical nature of knowledge.

Third, due to the rapid increase in communications bandwidth and computing and online storage capabilities mentioned above, digital collections are frequently updated. This reflects a key characteristic of 21st century collections: namely, that they are dynamic in nature. Our third premise is thus that any new information infrastructure must include dynamic model building capabilities.

In the final analysis, these three perspectives must be integrated into a cohesive whole. The goal of our research is thus to architect a computational knowledge management prototype based on HDDI™: Hierarchical Distributed Dynamic Indexing.

As part of our research we are discovering novel approaches to addressing the various issues of managing distributed digital information in the context of the aforementioned paradigm shift. This paper introduces the HDDI™ Collection Builder, our approach to automatic concept extraction from digital distributed collections of documents. The HDDI™ Collection Builder provides a basis and underlying data for the new indexing/textual data mining algorithms of HDDI™.

2 HDDIÔ Collection Builder

In the following sections we review the three functional parts of the HDDI™ Collection Builder: parsing, parts of speech tagging, and concept extraction. We also present the results of validation experiments that were performed on the Collection Builder.

2.1 Terminology

Before we introduce the functional parts of the system, we must introduce some terminology that will be used throughout this paper:

- Items: Item refers to the basic unit of data content that is used in textual data mining. We generally use item to refer to a single document; however as explained in [1], other units of information, such as subsections of a document or sentences, could be used as well. For simplicity, as in [1] and [18], we will assume that item refers to a document for the rest of this paper.
- Collections: A collection refers to a group of items that will be indexed in the HDDI™ system.
- Concepts: We use concept to refer to a maximal length English-language noun phrase that is extracted from a Collection's items. Concepts are used as keywords for the purposes of the HDDI™ computational knowledge management system.[3]

2.2 Parsing

The first step in the concept extraction process requires us to parse the original collection. Since the collection can originate from almost any source, we need the parser to work correctly on many different input formats. We also need the parser to extract and label specific fields from the collection. In order to accomplish these tasks we set out to create an extensible, reusable object-oriented parser.

Because we wanted the parser to be capable of handling data in many different input formats, we decided that the parser should have two inputs. The parser obviously requires the original collection (giving the text that is to be parsed) as its first input. The second input is a specifications (Specs) object, specific to a given data format and the needs of the user. This Specs object contains information the parser needs to extract the necessary content from the collection. The Specs object gives the sequence of characters which signify the end of a document, strings that identify the fields within the original document that are to be extracted, and strings that the parser outputs to label the content's field of origin. The use of this Specs object gives the parser its ability to parse almost anything that can be delimited; in practice, however, we generally use either XML or HTML collections. Given a Specs object which defines "DOC" tags as document delimiters, and "Title" and "Abstract" tags as marking fields with valuable content, Figure 1 shows an example of an XML item that could be used as input to the parser:

```
<DOC>
<Title>
Sample Article
</Title>
<Date>
6/14/00 21:40:00
</Date>
<Author>
R. H. Bader
</Author>
```

[3] See [1] for a review of Information Retrieval topics. Also, see [4], [5], and [6] for information about English-language grammar and noun phrases.

```
<Affiliation>
National Center for Supercomputing
Applications, Champaign, IL
</Affiliation>
<Abstract>
This is a sample abstract.
</Abstract>
<Body>
This is a sample article body.
</Body>
</DOC>
```

Figure 1: Sample XML input File.

Once the parser receives the Specs object and input in the form shown in Figure 1, it reads in one item from the collection. The parser, using information from the Specs object to break the input into tokens, loops through these tokens extracting the desired fields while ignoring the unspecified fields. When the parser reaches the end of an item, the process continues and is repeated for all other items in the collection. Figure 2 is an example of the output from the parser, using Figure 1 as the input XML file.

```
[<Title:14>, Sample Article]
[<Abstract:202>, This is a sample
abstract.]
```

Figure 2: Sample parsed output File. Only Title and Abstract fields were specified in Specs object. Field names are followed by the numeric offset which specifies the location of the extracted text in the original text.

2.3 Parts of Speech Tagging

The Parts of Speech Tagger is a rule-based system for tagging English parts of speech. This system is based on the SemanTag system ([10]) developed by Gregg Cooke, who based his work on Dr. Eric Brill's tagging system (see [7], [8], and [9] for detailed information about Brill's tagging system). It uses three levels of rule sets to determine the part of speech of each word, and tags words with their English part of speech tag, as specified in the Brown tagset (see [11]).

The tagger starts by initializing its rule tables. This involves reading about 94,000 lines of rules (of which all but about 350 are lexicon rules). The lexicon rules consist of a mapping of a given word to its parts of speech. After the lexicon rules are read, the lexical rules are read. These deal with roots of a word, and other internal properties that modify a word's part of speech usage. An example of a lexical rule would be "If a word ends in "ly", it is probably an adverb". These rules are put into a list, so they can be applied in order to each word from the input text. Due to the fact that there are only 134 lexical rules, the cost incurred by applying these rules is relatively low. The final set of rules consists of 220 contextual rules that deal with semantic information that can be derived from the position of a word in a sentence. For example, if a word is initially identified as an adverb, and the next word is a noun, a contextual rule would insure that the word is retagged as an adjective.

When the text input is read, a pre-pass separates punctuation by inserting spaces before each punctuation character. Next, the input is broken into a series of "CorpusWord" objects, each of which contains a word, a space, or punctuation, along with the tag that the word has been assigned. This process results in approximately twice as many objects being created as there are words in the input text. Once the input text is properly formatted, it is passed to the lexicon stage of the tagger.

In the lexicon stage, each word (except for those which have been marked as nonstandard) is looked up in the lexicon hash table, and the first entry in its parts of speech list is assigned to the word. If no tag exists for the word, the assumption is made that it is a noun, or a proper noun (if it is capitalized). The tagged text is then passed to the lexical rules stage of the tagger.

In the lexical rules stage each lexical rule is applied to each word. Any rule matches that are found change the part of speech tag of the word, overriding any earlier tagging decision that may have been made. After this step is completed, there are many fewer nouns as tags have been changed based upon word suffixes and other properties. Finally, the contextual rules are applied to each word, thus incorporating knowledge that can be gained from inter-word relationships into the tagging process.

It is interesting to note that the rules used, while adequate for Brill's purposes of tagging, fall somewhat short for use in tagging text from the World Wide Web, some of which is less formal or polished than text from a standard published source. Brill's rule sets still performed very well, but special exceptions needed to be coded for some very non-standard English.

2.4 Concept Extraction

A key part of textual data mining is feature, or concept extraction. For this purpose, we have designed and implemented a sophisticated English language noun phrase extractor. Our goal was to create a noun phrase extractor that would extract maximal length English language noun phrases. Our premise is that such noun phrases represent the most

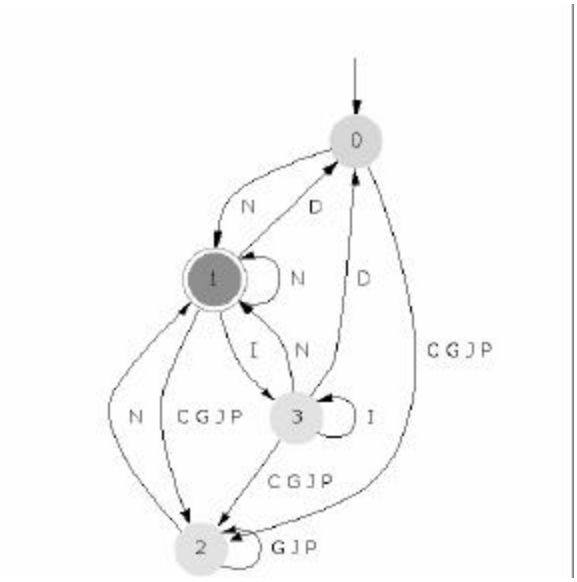
specific content in an item and should therefore be used as keyword features for indexing purposes by the HDDI™ textual data mining system. Initially, a recognizer was developed based upon work from [12], and a lexical analyzer created using JLex ([13]), a java version of the well-known Unix tool lex.

The first recognizer developed accepted the language generated by the following regular expression: [(d) a* n+]. This regular expression represents the maximal length noun phrase that can be formed by an optional determiner ("(d)") followed by zero or more adjectives ("a*") followed by one or more nouns ("n+").

It is important to note that this regular expression is not sophisticated enough to handle and extract complex noun phrases such as those containing prepositions, gerunds, or participles. This resulted in poor recall of maximal length noun phrases. To address this issue, we developed a second, more comprehensive regular expression to handle more specific types of noun phrases. Our second regular expression is shown in Figure 3. It handles more complicated noun phrases including those with participles, gerunds, and prepositional phrases. (See [4], [5], and [6], for definitions of the types of noun phrases.)

Concurrently with the extraction of noun phrases, other information that is used later in the HDDI™ model building stage is extracted and preserved. For example, a frequency of occurrence is calculated for each concept in each item as well as the character offset of each concept in the original item. Also, the field in which the concept occurred (e.g., title) is preserved. In addition, we have the option of enabling stemming of the concepts. The stemming algorithm is basically a modified version of Porter's stemming algorithm (see [15] and [16] for details on Porter's original algorithm), which was designed to work on single-word keywords. We simply apply Porter's stemming algorithm to each word of a concept, thus creating an algorithm that works on multi-word phrases (concepts). If this option is enabled, concepts that have similar meanings (but slightly different grammatical structure) can be reduced to the same concept. For example, the concepts "black dog" and "black dogs" reduce to "black dog" if stemming is enabled.

The following is an example of the functionality of the phraser. If the phraser received the input "She built an apparatus for the transformation of picture information." as the original text of an item, and "She//PP built//VBD an//DT apparatus//NN for//IN the//DT transformation//NN of//IN picture//NN information//NN .//." as the marked up text (see Figure 4 for an explanation of the parts of speech tags), the noun phrase "apparatus for the transformation of picture information" would be extracted with a character offset of 12. This concept would be given a frequency of occurrence of 1 since it occurs only once in this simple one sentence example. Note that this phrase was extracted using the second, more sophisticated regular expression for recognition of noun phrases; had the original regular expression been used, "an apparatus", "the transformation", and "picture information" would have been identified as the noun phrases.




```
C?(G|J|P)*N+(I*D?C?(G|J|P)*N+)*

C - cardinal number
G - verb: gerund or present
participle
P - verb: past participle
J - adjective
N - noun
I – preposition
D – determiner
? - option: zero or one occurrence
| - union
* - Kleene closure
+ - Kleene plus: 1 or more
occurrence
```

Figure 3: Second regular expression for English language noun phrases. The regular expression is presented both graphically and textually, followed by a key. In the graphic, state 0 is the start state, and state 1 is the final state.

```
DT - determiner
IN - preposition or subordinating
conjunction
```

```
NN - noun - singular or mass
PP - personal pronoun
VBD - verb - past tense
. - literal period
```

Figure 4: Selected Brown parts of speech tags and their definitions.

2.5 Validation Experiments

The following metrics were used to evaluate the effectiveness of the HDDI™ Collection Builder:

Precision = concepts identified correctly by phraser ÷ all phrases identified as concepts by phraser

Recall = concepts identified correctly by phraser ÷ total number of concepts identified by human expert

The following collections served as our test data:

- Grainger journal abstracts: A collection of 10 items chosen at random from the University of Illinois Grainger Engineering Library's Digital Library Initiative testbed (DLI [14]). The DLI testbed is a database of over 60,0000 engineering, computer science, and other technical full-text journal articles. For this collection, only the abstracts were extracted.

- USPTO and IBM patents: Both the United States Patent Office (USPTO) and IBM maintain web databases of various patent information. Both databases contain the full text of US patents from the early 1970's through the present, and contain over 2 million items each. A collection was formed from 5 USPTO and 5 IBM patent applications, for a total of 10 items, that were randomly selected from queries on various computer science topics, such as computer graphics, pipelining, processors, etc. As with the DLI test set, only the abstracts were extracted.

- Airline safety data: 10 airline safety reports were chosen at random from a database of airline safety data obtained from Boeing. For this collection, the full-text summary of each airline safety incident was extracted by the parser.

Concepts were extracted from each of these three collections using the HDDI™ Collection Builder and compared with a human expert analysis of the same items. The results are summarized below for each collection.

2.5.1 Grainger Journal Abstracts

Our approach for analyzing the data from these validation experiments was to perform a statistical analysis to determine the 95% confidence interval for the mean of the entire collection from a relatively small sample of the collection. We justify the use of a small sample by the fact that the precision/recall data for these items closely fits a normal distribution (see figures 6 and 8 for a comparison between the resulting precision/recall and a normal distribution). This enabled us to limit the tedium of manually calculating precision and recall. Figures 5 and 7 depict the 95% confidence interval for the means of the precision and recall obtained for this data set. As you can see, the Collection Builder performed quite well on the Grainger data. The precision is, however, slightly higher and has a smaller range for the 95% confidence interval than the recall. Due to the large standard deviation of the recall, the 95% confidence interval is large, ranging from 77.4% to 91.3%. If we were to rerun the experiment with a larger sample of the collection, we would expect that the confidence interval would decrease.

Grainger Precision

Population Standard Deviation	**2.967344**
Sample Mean	**96.93065**
Sample Size	**10**
Confidence Level	**95%**
Standard Error of the Mean	0.938356564
Z Value	-1.95996108
Interval Half Width	1.839142347
Interval Lower Limit	**95.09150765**
Interval Upper Limit	**98.76979235**

Figure 5: Grainger Precision 95% confidence interval

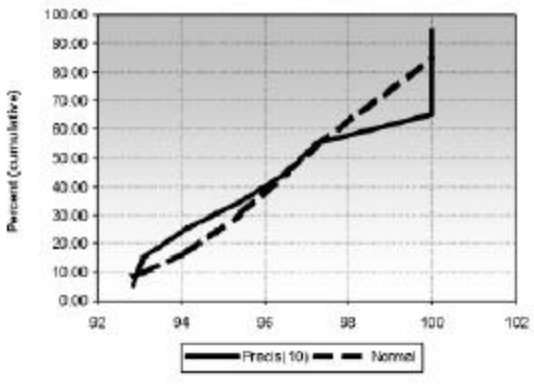


Figure 6: Grainger Precision vs. Normal Distribution

Grainger Recall Longest Match	
Population Standard Deviation	**11.10100**
Sample Mean	**84.3596**
Sample Size	**10**
Confidence Level	**95%**
Standard Error of the Mean	3.53576795
Z Value	-1.95996108
Interval Half Width	6.929967578
Interval Lower Limit	**77.42963242**
Interval Upper Limit	**91.28956758**

Figure 7: Grainger Recall 95% confidence interval

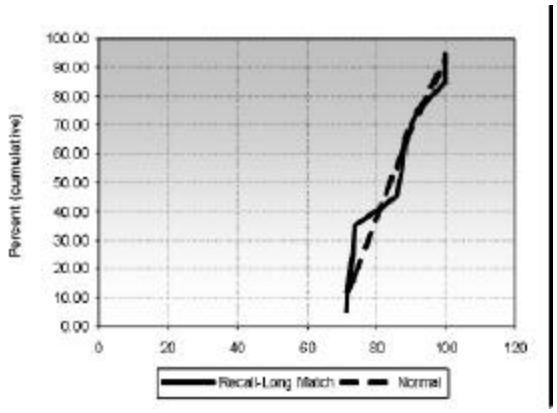


Figure 8: Grainger Recall vs. Normal Distribution

2.5.2 USPTO and IBM Patent Applications

Our approach for the patent data set was the same as that described above for the Grainger data set. The results as shown in Figures 9, 10, 11, and 12 were almost identical to those of the Grainger data set. Again, the precision was better than and had a smaller range for the 95% confidence interval than the recall, suggesting that a larger sample size would be useful to focus the recall confidence interval.

Patent Precision	
Population Standard Deviation	**3.180216**
Sample Mean	**97.20185**
Sample Size	**10**
Confidence Level	**95%**
Standard Error of the Mean	1.005672601
Z Value	-1.95996108
Interval Half Width	1.97107916
Interval Lower Limit	**95.23077084**
Interval Upper Limit	**99.17292916**

Figure 9: Patent Precision 95% confidence interval

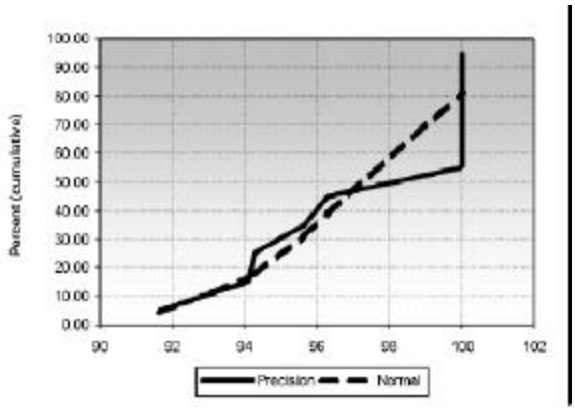


Figure 10: Patent Precision vs. Normal Distribution

Patent Recall:	
Population Standard Deviation	**12.09527**
Sample Mean	**83.08702**
Sample Size	**10**
Confidence Level	**95%**
Standard Error of the Mean	3.824860211
Z Value	-1.95996108
Interval Half Width	7.49657716
Interval Lower Limit	**75.59044284**
Interval Upper Limit	**90.58359716**

Figure 11: Patent Recall 95% confidence interval

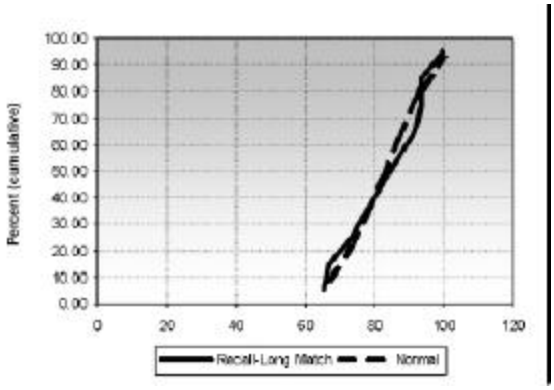


Figure 12: Patent Recall vs. Normal Distribution

2.5.3 Boeing Airline Safety Data

This collection proved to be somewhat of a challenge for the HDDI™ Collection Builder due to the unique nature of the airline safety incident reports that make up the collection. These reports are recorded in uppercase text, and almost all of the words are abbreviations. Both of these issues caused some interesting problems for the parts of speech tagger. With minor modifications, such as forcing all the text into lowercase, we were able to improve upon our initial results. However, as can be seen in

Figures 15 and 16, the recall for this collection was not nearly as good as for the other two collections. It is noteworthy, however, that the precision for the Boeing data set (Figures 13 and 14) is only slightly less than that of the Grainger and patent data sets.

The unique, highly abbreviated and grammatically incorrect text of the airline safety reports is most certainly the cause for the drastic decline in recall. We noticed that the difficulty stems from poor performance of the parts of speech tagging phase of the HDDI™ Collection Builder. Many words were tagged incorrectly, especially those that were abbreviations. The noun phraser actually still functioned correctly, however, its output was poor due to the poor quality of the tagged text that was received as input. If this type of data is to be used frequently with HDDI™ technology, it may be necessary to create a set of customized lexical/contextual rules for this data set and/or make other custom modifications to the parts of speech tagger. Another possibility would be to expand all of the known abbreviations before parts of speech tags are assigned. It is our belief that this would significantly improve the recall for this data set without the need for major modifications to the parts of speech tagger.

Boeing Precision:	
Confidence Interval Estimate for the Mean	
Population Standard Deviation	**4.362648**
Sample Mean	**93.71132**
Sample Size	**10**
Confidence Level	**95%**
Standard Error of the Mean	1.379590431
Z Value	-1.95996108
Interval Half Width	2.703943554
Interval Lower Limit	**91.00737645**
Interval Upper Limit	**96.41526355**

Figure 13: Boeing Precision 95% confidence interval

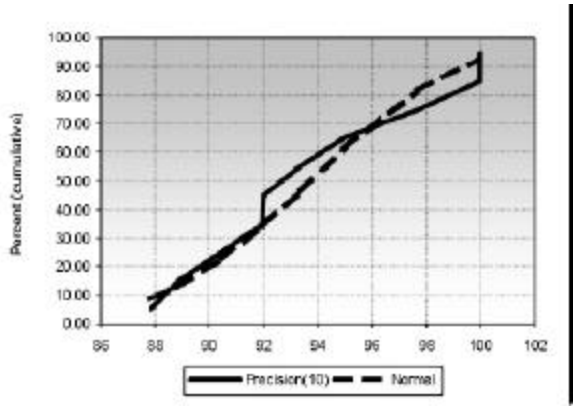


Figure 14: Boeing Precision vs. Normal Distribution

Boeing Recall:	
Population Standard Deviation	**9.94197**
Sample Mean	**68.07908**
Sample Size	**10**
Confidence Level	**95%**
Standard Error of the Mean	3.143926963
Z Value	-1.95996108
Interval Half Width	6.161974493
Interval Lower Limit	**61.91710551**
Interval Upper Limit	**74.24105449**

Figure 15: Boeing Recall 95% confidence interval

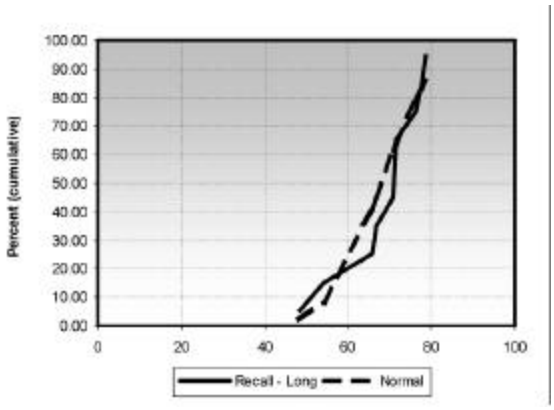


Figure 16: Boeing Recall vs. Normal Distribution

3 Summary and Conclusion

We have created the HDDI™ Collection Builder to perform automatic concept extraction from distributed digital collections of textual data. We have validated the performance of this system by calculating the precision and recall for several test collections. The initial results are promising, but further work is needed to test the Collection Builder in real-life applications. Ongoing work at Lehigh University involves the development of several applications based on HDDI™ technologies and the HDDI™ Collection Builder which will allow us to further validate and improve our approach to automatic concept extraction.

Acknowledgments

We gratefully acknowledge the assistance and contributions of the staff in the Hierarchical Distributed Dynamic Indexing Group at Lehigh University and NCSA[4]. We would also like to thank Gertjan van Noord for developing his FSA6 software ([17]), which helped us make Figure 3.

[4] Co-author William M. Pottenger, Ph.D., gratefully acknowledges the assistance of his Lord and Savior, Jesus Christ, in his life and work.

References

[1] F. D. Bouskila, "The Role of Semantic Locality in Hierarchical Distributed Dynamic Indexing and Information Retrieval", MS thesis, University of Illinois at Urbana-Champaign, Department of Electrical and Computer Engineering, 1999. (Thesis advisor was William M. Pottenger, Ph.D.)

[2] Los Alamos National Laboratory, "arXiv.org e-Print archive". http://xxx.lanl.gov , 1999.

[3] National Research Council CSTB, Computing The Future. Washington DC: National Academy Press, 1992.

[4] D. Keis, Modern English Grammar, a HyperTextbook for English 126, Department of English, College of Du-Page. papyr.com/hypertextbooks/engl126/phnoun.htm

[5] M. A. K. Halliday, Introduction to Functional Grammar, 2nd edition, London: Edward Arnold, 1994.

[6] R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik, A Comprehensive Grammar of the English Language, London: Longman, 1985.

[7] E. Brill, "A simple rule-based part of speech tagger", Proceedings of the Third Conference on Applied Natural Language Processing, ACL, 1992.

[8] E. Brill, "A corpus-based approach to Language learning", PhD. Dissertation, Department of Computer and Information Science, University of Pennsylvania, 1993.

[9] E. Brill, "Some advances in rule-based part of speech tagging", Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), 1994.

[10] G. Cooke, SemanTag, gcooke@rt66.com, http://www.rt66.com/gcooke/.

[11] W. N. Francis, H. Kucera, Brown Corpus Manual, Department of Linguistics, Brown University, 1979 revision. http://www.hit.uib.no/icame/brown/bcm.html

[12] L. Karttunen, "Directed Replacement", Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, ACL-96, Santa Cruz, California, 1996.

[13] E. Berk, JLex: A lexical analyzer generator for Java(TM), Version 1.2.5, Department of Computer Science, Princeton University, July 1999. http://www.cs.princeton.edu/~appel/modern/java/JLex/current/manual.html

[14] "UIUC Digital Library Initiative", http://dli.grainger.uiuc.edu/.

[15] M. F. Porter, "An algorithm for suffix stripping", in S. Jones, K. and P. Willet, Readings in In-formation Retrieval, San Francisco, CA: Morgan Kaufmann, 1997.

[16] M. Porter, The Porter Stemming Algorithm Official Homepage. http://www.muscat.com/~martin/stem.html

[17] G. van Noord, FSA6. http://odur.let.rug.nl/~vannoord/fsa/fsa.html

[18] Fabien Bouskila and William M. Pottenger, "The Role of Semantic Locality in Hierarchical Distributed Dynamic Indexing", Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI 2000), Las Vegas, Nevada, June, 2000.