Creation of a Style Independent Intelligent Autonomous Citation Indexer to Support Academic Research

Eric G. Berkowitz, Mohamed Reda Elkhadiri

Department of Computer Science
Roosevelt University
1400 North Roosevelt Boulevard
Schaumburg, IL 60173

Abstract

This paper describes the current state of RUgle, a system for classifying and indexing papers made available on the World Wide Web, in a domain-independent and universal manner. By building RUgle with the most relaxed restrictions possible on the formatting of the documents it can process, we hope to create a system that can combine the best features of currently available closed library searches that are designed to facilitate academic research with the inclusive nature of general purpose search engines that continually crawl the web and add documents to their indexed database.

Introduction

RUgle is a system composed of three major components, a Web scanner, a document analyzer and cross-referencing system. Using the system it is possible to perform searches of academic and other research papers extracted from the World Wide Web without needing to sort through the seemingly endless number of loosely related or valueless Web documents returned by traditional searches.

Background

Current search engines for the World Wide Web fall into one of two categories. The first is the general purpose search engine examples of which are Google.com, Yahoo.com and AltaVista.com. The engines attempt to index the entire expanse of the World Wide Web using the words and phrases in the document as tokens with which to build and index. Using these search engines, one can search the largest collection of documents from the Web. Unfortunately, this abundance of documents leads to the retrieval of many documents that may be of little worth. The lack of focus on quality also creates collection that include a large number of papers but may not include many scientific papers (Lawrence, Bollacker and Giles 1999). Additionally, the use of the entire collection of words, in the document as tokens for the index can cause a search to retrieve many irrelevant documents, particularly when the words in the search term are quite common (Bradshaw Sheinkman and Hammond 2000). For example, if one is seeking information on the content of a will one might use the word "will" as a search term. Since "will" has other meanings as a verb and a proper noun, the common search engines will return a plethora of irrelevant documents. One might then be tempted to use the search term "will and testament." This phrase might however lead to the exclusion of many relevant documents since the word "testament" does not always accompany the word "will," even in the correct context. Another problem is that if one finds a relevant document in the list returned by the search engine, having this document does not always lead to additional relevant documents. Even if the document in question has hyperlinks to other documents, the reason one might link from his document to others on the Web are many and varied and not always based on any form of relevance. Since the links in a document on the Web are placed there for the specific purpose of facilitating navigation around the WWW they incorporate the document authors' ideas, both conscious and subconscious regarding other documents a reader should view. This brings up three major shortcomings of navigating the Web via links in existing documents.

1. The link's sole purpose is to facilitate the navigation from one document to another and therefore incorporate the ideas of the designer, both conscious and subconscious, regarding other documents a user should view.
2. Designers of documents usually possess a very poor knowledge of what is available outside their own collection of documents or their own site and therefore, given the purposeful nature of links on the WWW, those links that a designer provides to other sites are often few and inadequate.
3. Related documents are often not linked at all since the designer of one document may not be aware of the existence of other relevant documents, or may simply not care to provide links to the other documents, or may even want to impede (or at least not facilitate) navigation to those other documents.

Thus attempting to navigate the WWW from within a single site fits the old saying of not being able to see the forest for the trees; one often cannot move away from one location even if a bird's-eye view would allow one to see that useful information is very nearby.

Library collections such as CiteSeer attempt to resolve many of the issues listed above. Document are only included in the collection to be searched if they meet some criteria of relevance. The content of the document is not used as a single collection of equally weighted tokens and

one can currently only search for terms in the authors names, paper title, or body. Thus one is more likely to receive a list of valuable, relevant documents from a search. Still these mechanisms suffer from shortcomings of their own. They rely on closed collections of documents. Addition of a document to the collection usually requires manual data entry or manual data correction making the process quite slow (Lawrence and Giles 1999(3)). For example, the DBLP engine relies on manual data entry. In order to facilitate the indexing mechanisms used, the documents are usually restricted to a single domain or a collection of domains. For example, CiteSeer uses a list of URLs known to contain papers and actively attempts to download papers from these sites with secondary reliance on active searching. It is, however, restricted to a limited set of domains and still remains quite dependent on manual data entry. Authors are encouraged to log on to the site and to manual correct the entries for their papers. Such search engines also usually require documents to be submitted or at least the submission of a top level URL to be searched. Manual filtering of the documents or top level URLs is required to guarantee the domain restricted nature of the collections. They do not find documents on their own. Thus, while they may aid in finding only quality, relevant documents when searching for information in an included domain, they exclude a wealth of other quality, relevant documents that have not been added to their collection.

RUgle is the first product of our research into producing a system that combines the best features of both types of searches while minimizing the effect of their shortcomings.

Design

RUgle's currently operates using five computer systems running Solaris, Linux, and Windows as shown in Figure 1. The Solaris server, called Epsilon, is the maindatabase server and contains the cross reference database with information extracted from the papers along with tracking information used by the crawler. A Linux computer, named Xeon, serves as a backup database server. Another Linux computer, called RUgle, runs the code for the crawler and document analysis interacting directly with the database server and the WWW. The windows machine, called Winbox runs code required for analysis of documents in Microsoft proprietary formats and operates under control of the document analyzer running on RUgle. The second Solaris server, named Lambda, runs the user interface code described later in this paper. From a logical perspective, RUgle's main engine is a single loop that works as follows:

- A single unprocessed citation is retrieved from the database.
- The citation is sent to the main crawler that attempts to download the referenced document using traditional search engines.
- The downloaded document is analyzed and:
    - the title and authors' names are entered into the database.
    - the bibliography entries are entered into the database and stored for later processing by the system.
- The next bibliography entry is retrieved and the loop repeats.

This logical flow is shown in Figure 2.

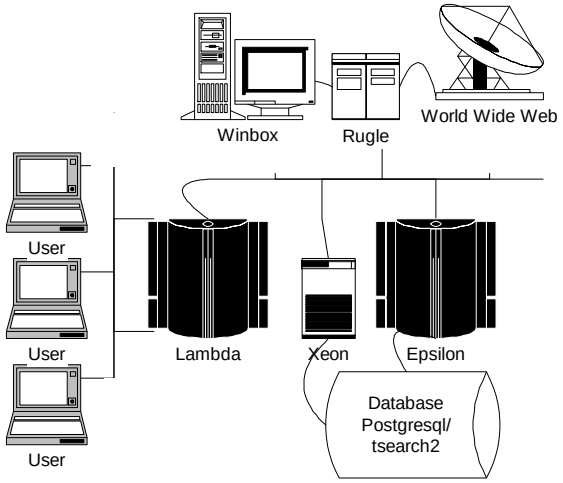

Figure 1

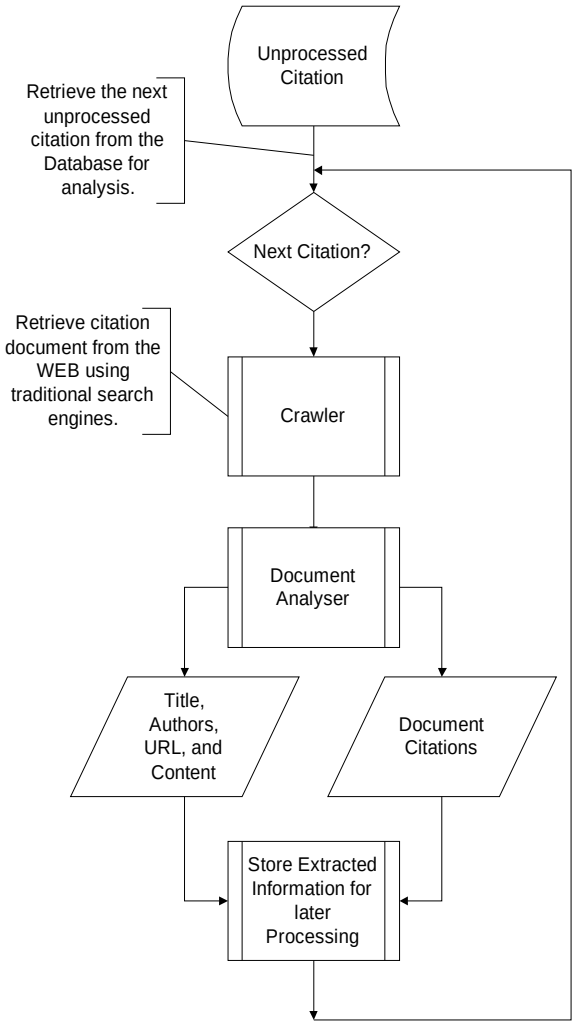

Figure 2

Active Acquisition of Documents on the Web

In order not to be restricted to a small collection of submitted documents, RUgle must actively scour the Web for documents to add to its collection. Thus RUgle employs a crawler or Web spider. RUgle's crawler is however very different from traditional crawlers in the manner in which it moves from document to document on the Web. It does not use hyper-text links to do so. As described above, these links do not always represent any objective form of relationship between documents. Instead, RUgle analyzes the content of the document currently being processed. RUgle extracts from the document the name of the author, the title and the bibliography. RUgle is designed with the assumption that a document of value will contain citations to other works. After a document is analyzed, RUgle iterates over the bibliography and extracts the components of each entry, particularly the authors of the cited work and its title. It then uses information provided by traditional search engines to try to locate the cited work on the Web. Thus, rather than relying on hyperlinks or a set of common tokens, works are considered related because one cites the other. Since a bibliography lists cited works, it is far less susceptible to purposeful manipulation than hyperlinks placed to facilitate Web navigation. Papers that are found are then downloaded from the Web. Their title, author and bibliographies are extracted and then queued for searching in a breadth first manner. The issues encountered in developing a crawler of this kind are in actually extracting the critical information from a document. It is for this reason that the traditional library collections require a full set of BibTex entries to be submitted with each paper to be indexed that incorporate information on the paper itself and the complete bibliography.

Formatting Languages

In order to process the documents and find the elements RUgle needs to index them and continue scanning the Web, RUgle needs to be able to analyze the document content. Given the almost ubiquitous nature of Adobe's Portable Document Format (PDF) on the Web we began by processing this type of formatting. Rather than parsing the formatting tags in the PDF, it was decided to use available tools designed to render a PDF document as ASCII text while maintaining as close an approximation as possible to the documents original layout. RUgle's first step in processing a document is to convert it to text. After converting the document to text RUgle begins to analyze its content.

While the PDF format is very common on the Web, we found it to be far more common in the sciences, slightly less common in the social sciences, and even less so in the humanities. If RUgle were not to be skewed against indexing papers in the humanities and toward the hard sciences it needed to be able to process documents as they are posted on the Web by researchers in those fields. Our investigation revealed that it would be crucial for RUgle to process documents in Microsoft Word (doc) format for it to have a more universal indexing ability. Doc format is far more complex than PDF, and, unlike PDF, it is not an open standard. Thus there are few tools to process documents in this format and most of the available ones have shortcomings that preclude them from processing even large subsets of documents in this format, particularly documents with embedded graphics. Even if such a tool were available, it would require constructing a whole new interface to the parsing routines and complicate system enhancements. It was decided therefore to convert doc format documents to PDF as a preprocessing stage and then let the PDF processing engine take over. The most reliable tool for the task proved to be Microsoft Word itself. RUgle searches the Web for documents in both PDF format and doc format. Documents in doc format are automatically passed through Microsoft Word and rerendered as PDF documents for further processing.

Columns

The first hurdle to processing documents originally designed for publication in journals is that many of them tend to be formatted as two columns. Linear processing of such documents yields collections of sentence fragments and renders the document useless. This because, while visually the document is in two columns and a human reader easily identifies columns on a page and treats them as separate pages when reading, the columns have no manifestation in the actual stored document which is stored as a collection of pages with linear text from left to right across lines of the full page. RUgle must determine if a document is indeed formatted in columns, and, if it is, recreate the document in one column with linear text.

RUgle does this by scanning a page and searching for word gaps (spaces) that appear in the same place in the text in all lines of the page. RUgle is built on the assumption that the probability of such a gap being a random occurrence is very low. If RUgle finds such a gap it treats it as a column break and rerenders the document wrapping the text from the left margin to the column break for the left column and the end of the column break to the right margin for the right column. This analysis is particularly difficult on pages that include bibliography entries since such entries often incorporate various amounts of spacing and indentation in the bibliography itself.

Analyzing Document Content

Since RUgle is designed not to require documents to be submitted but to locate them independently and also not to require manual data entry, RUgle needs to be able to find the data it needs to process the document on its own. Were we to restrict RUgle to a limited domain, one would think this task would be somewhat simple since, for example, papers on computer science are usually formatted using the style dictated by IEEE or the ACM. This domain specific information proved however, to be of limited values since

RUgle is intended to be completely generalized and index documents in all fields.

Our first efforts were to build parsers for the most common document formatting styles in use including APA, PAR, Chicago style, AMA, ACM, IEEE, MLA, and AAAI. Several such parsers were written and incorporated into early analysis subsystems for RUgle.

Quite surprisingly, we discovered, that even for may documents claiming to be formatted according to the dictates of a particular style, they do not actually follow all the rules of that style (Lawrence and Giles 1999(2)). While aesthetically, the papers tend to adhere closely enough to the style dictates that they look good when grouped together with other documents that also claim to follow the given style in a single publication, they actually deviate from the fine details to a sufficient degree to significantly complicate computer analysis. Distinctions that are lost to the human eye are quite apparent to any scanning and parsing algorithm. Thus, though we had created a large variety of parsers we were having difficulty analyzing even the types of documents for which they were specifically written. We learned that to function as intended, RUgle needs to not only process documents that adhere to a wide variety of formatting styles, such as APA, Chicago style, AMA, ACM, IEEE, MLA, and AAAI., but also documents that only loosely follow one of these styles, and documents that do not follow any specific style at all. Since writing an entire collection of parsers, even if a doable task, proved to be of limited value, we attempted to determine what could be assumed about the way people format documents.

We discovered that the majority of bibliographies fall into one of two categories: those that place the date at the end of the citation (an end date entry or EDE) and those that place the date in the middle of the citation (a mid-date entry or MDE). In the latter case, the date, possibly along with some collection of punctuation, usually separates the authors' names, that precede it, from the title of the publication that follows it. In the former case it is usually punctuation alone, and sometimes little more that a single space, that separates the authors' names from the title and they usually are in that order in the citation.

Thus we built a style independent, three-tiered system for processing bibliographies. The lowest tier is made of two parsers, one that processes EDEs and one that processes MDEs. Above that is a tier that attempts to unify stylistic elements of each type of entry so that the parser can successfully analyze it. At this level RUgle attempts to determine what form of punctuation the author used, such as interspersing commas or periods between bibliography entries and enclosing the date in parentheses. Once this information is determined, the entry is modified to conform more closely to a single style understood by the parser. At the top tier, RUgle attempts to determine whether the entries should be sent to the MDE unifier and parser or the EDE unifier and parser.

The other issue is to find the title and author of the document being analyzed itself. For this we integrated into RUgle the algorithms originally developed for an automated literature review system we had been developing. The algorithm determines the title by assuming the approximate location of the title in the document and then removing all the text that is not the title thereby leaving only the title. RUgle employs a two tiered system for extracting the title. The first tier attempts to determine where the title and authors' names should appear in the document. The second tier attempts to discard all other information from that section of the document and to determine from the remaining text, the title of the document and the names of its authors.

User Interface

Conventional search engines maintain complete full text indexes (FTIs) of the documents in their database. This allows a user to search for a document using any word or phrase that appears in the document. The document is treated as a single string of tokens with equal weight. The weight associated with any given token in the document is increased based on the number of times it appears in the document. Thus words in parenthetical phrases are given the same weight in indexing as words from a title, section heading or other presumably more important sections of a document that are intended by the author to summarize and typify the content of the document. RUgle maintains a full text index of only the title and authors' names. While it is the intention to maintain an FTI of only the title, the current algorithm confuses the title and author to a sufficient degree that it was decided to index both entries as a single unit. The result is that a search for a term such as "executive compensation" results in a list of documents that:

1. meet RUgle's criteria for quality. They have a stated title, author and a bibliography.
2. contain the words in the search term in the title of the document.

That is, by the criteria used to design RUgle, a quality document that the author believed addresses the terms in the search. The results page of a RUgle search is shown in Figure 3.

When a user select a title from the list on the results page, the user is presented with the following information:

1. The title and author of the paper.
2. A list of all of the citations in the bibliography of the paper.
3. A list of all cited papers that also exist in RUgle's database.
4. An image of the first page of the selected paper.

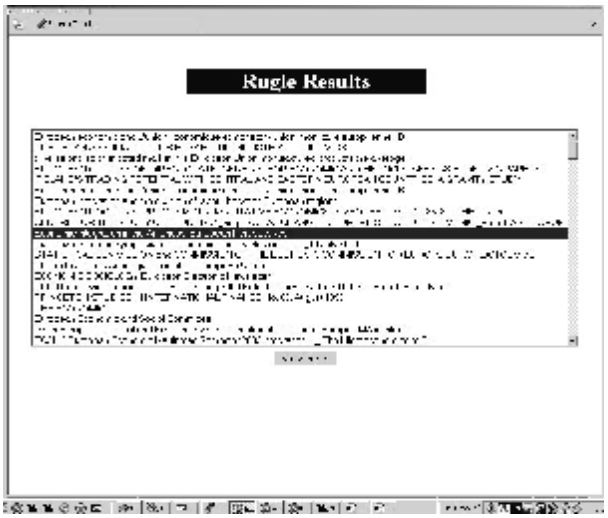

Figure 3

A user can decide to:

- return to the list of titles.
- download the paper.
- select a new paper from list of cited papers that are in RUgle's database and view it. The document display page is shown in Figure 4.

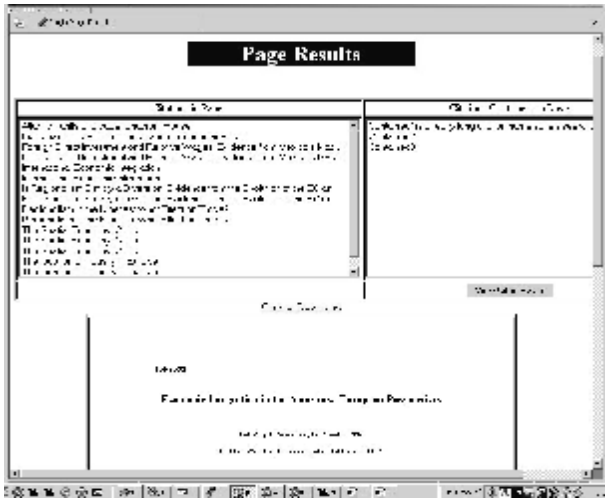

Figure 4

Analysis of Current Performance

RUgles currently has downloaded approximately 700,000 documents from the WWW. To determine how well the current algorithms are performing their tasks, we selected a random sample of 200 papers and reviewed the title and author names that RUgle had determined for them. The results are shown in Table 1.

Criterion	Percentage of Papers
Exact Title	43.27%
Title + Extra Text	11.54%
Title + Author as Title	5.77%
Partial Title	12.5%
Total Titles: 73.08%	
Exact Author	25.96%
Author + Extra Text	22.11%
Partial Author	2.88%
Total Authors: 50.95%	

Table 1

The performance for the bibliography extraction is somewhat lower. From 418174 papers scanned for a bibliography, entries were only extracted from 133882 of them. Much of this can be explained by RUgle's inability to filter out documents whose content makes them appear academic but which in reality are not and do not contain bibliographical data such as professors' resumes, course syllabi, course notes, industry white papers, etc.

Conclusion

RUgle is quickly becoming viable a system for generalized academic research. Its mechanisms, however, ignore much of the visual information hidden in a document as described in Berkowitz and Mastenbrook 2002. Utilizing this information would enhance the quality of the analysis and the indexing and it is our intention to incorporate the Purpose Encoding Document Abstraction language into future versions of RUgle. RUgle's generalized scope and its ability to index papers from all fields of research are helping to make it a useful research tool in its own right. The system's current ability to locate and then analyze bibliographies is lacking and research is being done on how to improve this aspect of the system. We are also working to improve the ergonomics of the user interface. One problem we are currently facing is the magnitude of the database which currently stores over seven million bibliography entries even with RUgle's limited ability to extract this information. Queries on the database are beginning to take more time than desired, slowing down both the updates and the end-user interface. We are researching methods to improve the overall design of the database along with methods for distributing and replicating the data to accelerate read-only queries from the user interface.

References

Eric Berkowitz and Brian Mastenbrook. “A Visual Formatting Purpose Representation Language to Enhance Automated Document Classification, Retrieval, and Indexing” In Proceedings of the 16th Annual Midwest Computer Conference, Schaumburg, IL, April 2002.

S. Lawrence, and C.L. Giles, "CiteSeer: An Automatic Citation IndexingSystem," IEEE Communications, Volume 37, Number 1, 1999, pp. 116-122.

S. Lawrence, and C.L. Giles, "Digital Libraries and Autonomous Citation Indexing," IEEE Computer, Volume 32, Number 6, 1999, pp. 67-71.

S. Lawrence, C.L. Giles, and Kurt D. Bollacker, "Autonomous Citation Matching," Proceedings of the Third International Conference on Autonomous Agents, ACM Press, New York, 1999.

S. Lawrence, C.L. Giles, and Kurt D. Bollacker, "Indexing and Retrieval of Scientific Literature," Proceedings of the Eighth International Conference on Information and Knowledge Management, 1999, pp. 139-146.

S. Bradshaw, and K. Hammond, "Automatically Indexing Documents: Content vs. Reference," In Proceedings of the Sixth International Conference on Intelligent User Interfaces, San Francisco, CA, January 14-17, 2002.

S. Bradshaw, A. Scheinkman, and K. Hammond, "Guiding People to Information: Providing an Interface to a Digital Library Using Reference as a Basis for Indexing," In Proceedings of the Fourth International Conference on Intelligent User Interfaces, New Orleans, LA, January 9-12, 2000