5-2007

Automatic pure anchor-based taxonomy generation from the world wide web.

Joseph Paul Elliott
University of Louisville

AUTOMATIC PURE ANCHOR-BASED TAXONOMY GENERATION FROM THE WORLD WIDE WEB

By
Joseph Paul Elliott
B.S., University of Louisville, 2003

A Thesis
Submitted to the Faculty of the
Graduate School of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Master of Engineering

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2007

AUTOMATIC PURE ANCHOR-BASED TAXONOMY GENERATION FROM THE WORLD WIDE WEB

By
Joseph Paul Elliott
B.S., University of Louisville, 2003

A Thesis Approved on

May 2, 2007

By the following Thesis Committee:

Dr. Antonio Badia, Thesis Director

Dr. Olfa Nasraoui

Dr. Suraj Alexander

ABSTRACT

AUTOMATIC PURE ANCHOR BASED TAXONOMY GENERATION FROM THE WORLD WIDE WEB

Joseph Paul Elliott

May 3, 2007

This thesis proposes a new method of automatic taxonomy generation using the link structure of Webpages. Taxonomy is a hierarchy of concepts where each child concept is said to be encompassed by its parent concept. Techniques have previously been developed to extract taxonomies from a traditional text corpus, but this thesis relies exclusively on the links between documents in the corpus, as opposed to the text of the corpus itself.

A series of algorithms were designed and implemented to realize the objectives of this thesis. These programs perform comparably to other techniques using the text in the documents and have shown that there is information available in the link structure of Webpages when creating concept taxonomies.

ACKNOWLEDGEMENTS

I would like to thank Dr. Antonio Badia for all of his guidance over the course of the year. Without his help this thesis would not have been possible. His knowledge and perspective helped make this thesis what it is. I would also like to thank Dr. Olfa Nasraoui for providing valuable suggestions and recommendations on how to improve this thesis.

Finally, I would like to thank my girlfriend, Anne, for putting up with me and making me countless frozen dinners while working on this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Background

The World Wide Web is an ever changing, yet vast source of information. New websites are constantly appearing and old ones disappearing. Two pages may be connected via direct link today and not tomorrow. Also, web pages are created using a variety of technologies (HTML, CSS, JavaScript, Flash, etc.) and presented in as many human languages as one can name. However, in spite of the difficulty of extracting data from the Web, researchers continue to try, due to the abundant amount of available information on any conceivable subject.

One such set of attempts is the Semantic Web [13], an effort directed toward a more complete markup language that the creators of Webpages can use to identify pieces of data on their websites. Therefore, if items on a webpage are appropriately marked, software agents can know what kind of data a text string represents, instead of just displaying it to the user. The Semantic Web uses the Resource Description Framework (RDF), a markup language designed to describe objects, and to present information in an organized and consistent form. It also uses the Web Ontology Language (OWL) to describe objects, properties of those objects, and their relationships. Using both RDF and OWL web designers can describe the data in their Webpages and how it relates to both itself and other pieces of data in a form more easily manageable by software. This approach to extract data effectively and accurately from the internet puts the burden on the designer to create their pages in a manner that can be easily parsed and understood and ties them to rigid standards in the way they encode their content.

Researchers have also attempted to automatically extract information from the World Wide Web by applying traditional Information Extraction techniques to the Web's documents, Webpages [1,3,4,5,8]. For instance, in [3], PANKOW is developed to search for lexico-syntactic patterns on the internet using Google in order to discover concept relationships. Lexico-syntactic patterns were originally developed to be used on a traditional text corpus, but are now applied to the Web. Also, [4] discusses how to represent terms as vectors extracted from webpages and calculate statistical similarities between the vectors to determine concept relationships. Similar approaches, developed for Information Extraction from a standard text corpus, are being tried on the Web.

Due to the difficulty of creating taxonomies (from either a traditional text corpus or the World Wide Web), generally an ontology engineer, a human judge with topic specific knowledge, is required to validate, trim and add to the automatically generated ontology. This is known as the Knowledge Acquisition Bottleneck. Because of this bottleneck, heuristics that create more accurate ontologies are very valuable as they reduce, and may one day eliminate, the time required for the ontology engineer to review the created ontology.

This thesis outlines a new method of taxonomy construction built from data extracted from the link structure of the internet. The taxonomy created displays perceived relationships between terms, and is displayed as a rooted tree. That is to say, it is a graph with no cycles and has a designated root node. Each node represents a term and the edges from one node to the next represent a parent-child relationship. The concept represented by the term in the child node should be more specific and subsumed by the concept represented by the parent node. The algorithms in this thesis also identify topic and instance nodes/terms. A *topic node* is more generic and is able to be expanded into more specific concepts. An *instance node* is specific and is a single occurrence of a particular topic node. An example taxonomy is show in Figure 1-1.

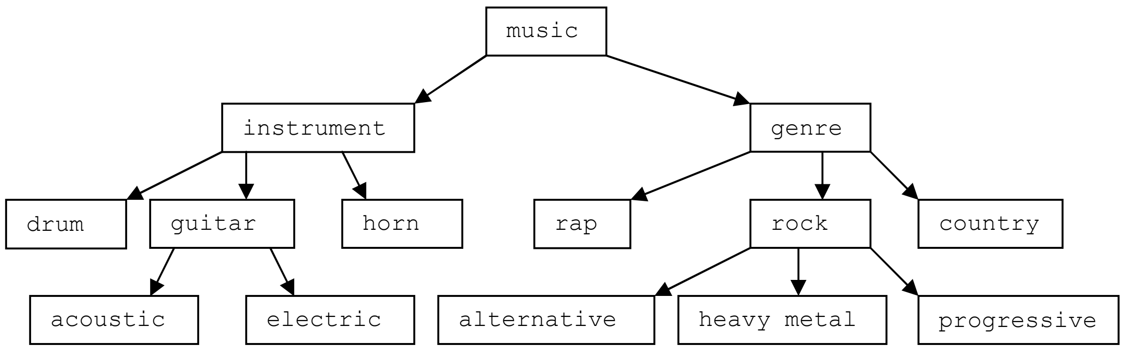


Figure 1-1 Example term taxonomy

Currently there are no “perfect” taxonomy creation algorithms. Every algorithm has its strengths and drawbacks and use different pieces of information for generation. The heuristics proposed in this thesis are unique in that they use specifically the link structure of Webpages and the anchor text in links to automatically generate taxonomies. The software developed in this thesis was able to create viable taxonomies from three different topic domains, SPORTS, COMPUTER HARDWARE, and NEWS. Each of these taxonomies were judged by humans and found to perform favorably based on developed statistical metrics that evaluate the consistency of a taxonomy. Additionally, while most existing techniques use the *entire document text*, this thesis only uses a fraction of the document that is limited to the *anchor text*. Therefore this thesis shows that there is valuable and currently ignored information inherent in the link structure of webpages for automatic taxonomy generation.

1.2. Organization of this Thesis

Chapter Two presents a detailed literature review of existing automatic taxonomy generation techniques. Chapter Three discusses the algorithms used to generate taxonomies from links in Webpages. Chapter Four presents the software developed for this thesis to implement the ideas in chapter two. Chapter Five presents the experiments designed to test the performance of our approach, then shows the results obtained, and discusses ways to improve the algorithms discussed in this thesis. Finally, Chapter 6 discusses the conclusions reached from performing the research in this thesis.

2. LITERATURE REVIEW

The Web is a very large source of potentially valuable, unorganized information. Because it is such an impressive source of information, a substantial effort has gone into extracting information and knowledge from, and organizing its contents.

2.1. Bottleneck Minimization/Process Definition

One set of efforts has been directed at creating concept hierarchies to act as a backbone for organizing other sets of information. These techniques generally suffer from a knowledge acquisition bottleneck, the process of a human knowledge engineer pruning a tree created from one of a variety of algorithms. Tools have been developed to lower the time and resources spent pruning these hierarchies [1,5,9].

[1] details a tool developed to compare a variety of clustering techniques in order to more easily determine which methods are the most effective in ontology building. The developed workbench provides methods for choosing what grammatical relations indicate a relationship between concepts and what pruning threshold to use. It uses a standard vector based distance measure to determine if two concepts are related.

The Mo'K workbench is an excellent tool for comparing how grammatical relationships and pruning parameters affect ontology building. It facilitates a deeper insight and understanding these parameters contribute to the final ontology.

The Mo'K workbench is similar to the *SiteGraph* tool developed in this thesis. They are both graphical utilities that allow the user to modify factors controlling the taxonomy and see how it affects the relationships therein.

[5] uses K-Means clustering and Latent Semantic Indexing to extract topics from a corpus. These techniques were combined into a software suite that proposes topics and relationships to an ontology engineer in a concise, manageable way who makes the final decisions about the topic relationships.

The authors outline an interface that the engineer could use to facilitate the process of pruning an existing ontology created using a variety of other approaches. It provides an impressive and complete interface to allow quick and easy ontology editing.

The Ontology Editor developed in this paper is similar to the *SiteGraph* application developed in this thesis. However, while they both provide graphical representations of the created taxonomies, Ontology Editor allows the user to delete relationships in order to refine the taxonomy. It is a tool for directly editing taxonomies while *SiteGraph* is a tool for viewing and manipulating taxonomies created by the ideas in this thesis.

[9] outlines a process for semi-automatic ontology construction. It identifies four steps for ontology learning used by their Ontology maintenance application, OntoEdit. The Import/Reuse step discusses the methods for which already defined ontologies can be merged with an existing one. Extraction deals with the actual ontology creation from a text corpus. Various standard ontology extraction techniques are discussed such as Hierarchical Concept Clustering and Lexical Entry Extraction. Hierarchical Concept Clustering uses the similarity of items to create a hierarchy by grouping those items

which are most similar. Similarity can be calculated using measures such as adjacency or syntactic relationships. Lexical Entry Extraction is the process of extracting N-grams from a document set based on statistical frequencies of co-occurrence. The third step, pruning an ontology, includes the balancing act of removing those concepts and relationships that are most likely invalid from the ontology. The final step is ontology refinement. During the refinement step the ontology pieces of the ontology are extracted and fine tuned for use with a specific application. The authors provide a good overview of various techniques used in Information Extraction and Ontology Building. They also suggest a solid, robust framework for creating an ontology building system and addresses the challenges and benefits to automatic ontology creation. However, it does not significantly add to the study of ontology building and only serves as summary or introductory work. This paper discusses very broad ideas regarding an entire life cycle of ontology maintenance. However, the ideas in this thesis are a very specific method that would fall under the single step, Extraction, discussed in this paper.

2.2. *Syntactic Analysis*

Many techniques have been used to create the taxonomies to be evaluated by a knowledge engineer including traditional knowledge extraction techniques such as collocation measures and syntactic pattern matching to the large text corpus that the internet provides [3, 4, 7]. These measures have enjoyed varying degrees of success, but still require human interaction to create viable term hierarchies.

[3] discusses a method of using Google to expand a text corpus to find instance of and subclass relationships. Previous attempts to use text patterns to find these

relationships suffered from a lack of text to search. The method takes a given Instance `<I>` and Concept `<C>`, and searches for the Hearst Patterns [14], e.g. `<C>s such as <I>` and `such <C>s as <I>`. It also uses Definite, Apposition and Copula patterns. Each of these pattern sets exploits commonly used syntactic patterns to extract term relationships. Using this method a given instance can be compared with a large number of concepts to see which concept it matches up with most successfully.

Although this method provides impressively reliable results it requires a rather narrow set of instances and concepts to be effective. Since every combination of every concept and instance and phrase must be searched this technique would require a large amount of time and internet resources to process even a small set of data. If only 10 concepts and 100 instances and all 8 mentioned patterns are used it would require 10 * 100 * 8 = 8000 Google searches on a very small set of data. Another weakness to this approach is that it requires a set of concepts and instances to start with as opposed to generating its own from the corpus. For this reason it would be a powerful, albeit expensive, pruning algorithm as opposed to an ontology builder itself.

These methods would be a great addition to other ontology building algorithms because it can reinforce or refute an existing set of relationships. For instance, if these methods were to be integrated it into the ideas in this thesis, they could be used to check all created relationships to see if they appear a significant number of times on the internet. This would be a very powerful pruning step if the resources were available.

[7] proposes a new method for extracting concepts and relationships from text documents and compares it to another approach employed by Text-To-Onto. Text-To-Onto is another automatic taxonomy generation program which uses the *tf/idf* measure to

compute similarity. *tf/idf* (term frequency–inverse document frequency) is a measure used in information retrieval to determine the importance of a word by using its frequency of appearances. CRCTOL (Concept Relation Concept Tuple based Ontology Learning) is a system that given a parsed and tagged text input and a domain lexicon creates an ontology. This system uses TIM-DRM scores to create an initial list of terms and then uses a second process to pull commonly used single terms out of multi-term phrases found using TIM-DRM. TIM-DRM is a statistical measure, which uses *tf* and *idf* and adds syntactic relationships, to create a more meaningful weighting parameter. Semantic Relations are extracted using syntactical relations of the form `<Noun, Verb, Noun>`. The two nouns have a relationship through the verb.

In a given domain this technique outperformed the Text-To-Onto in both concept and relationship extraction using the measures of recall and precision.

The techniques discussed in this paper are similar to the ideas in this thesis in that they both automatically attempt to generate taxonomies. However, CRCTOL is based on syntactic analysis of a set of documents while this thesis only uses the link structure of a set Webpages to determine concept relationships.

[6] generalized to the case of digital libraries, the syntactic hierarchy based similarity originally proposed in [19] to compare URLs for the purpose of clustering user sessions. The idea behind this similarity is that URLs are considered closer if they share in common a larger proportion of the path from the root of the hierarchy to each of the URLs.

2.3. Statistical Analysis

Clustering algorithms have also been applied to sets of Webpages in order to measure similarity and extract hierarchies [2, 8]. These clusters have also been compared to a site's link structure as a measure of the cohesiveness and organization of the site [10].

[10] attempts to use page clustering and link structure to make suggestions on how to improve a website. The authors suggest a method of slowly increasing the number of clusters to create a tree of which the root node contains all of the pages and each level groups the pages into an increasing number of clusters. The number of clusters to be used in this process is determined experimentally for each site. Then, using usage information, the most frequently visited clusters are identified. The authors then conclude that for "good" website design, clusters of pages should be connected together and those most visited clusters should have links from the main page. While this approach does use the link structure of the site to determine information about the site, it does not attempt to build taxonomies of terms from this structure as does this thesis. However, the paper uses this information, compared to usage statistics and page similarity measures to try to determine if a site is organized well.

[8] gives an overview of the current state and accepted practices in retrieving taxonomies from the web and then proposes a combination of syntactic and statistical methods for creating taxonomic relations. It states that the solution to more intelligent and functional web services is semantic descriptions of objects that will facilitate a "new level of Web Intelligence". The paper concludes that automatic ontology generation and object classification is the key to the Semantic Web and, therefore, research in this field is valuable. Unfortunately, due to the complexity of information available on the web

this is also a very difficult task. The author begins by classifying automatic ontology generation into two main categories. The first is a symbolic approach which relies on matching lexico-syntactic patterns to create term relationships. While this technique is powerful it suffers from being language dependent and not as scalable as its alternative, a statistic based approach. The statistics based clustering approach uses a similarity measure and a computation strategy to create clusters of similar words. It has the strength of being scalable, however, it suffers because it doesn't make use of language constructions which have been shown to be very powerful. The author then goes on to propose a new method combining the advantages of the statistical and symbolic approaches. Given an existing taxonomy and a new word to insert, the authors propose considering both the words semantic and statistic relationships to the words already in the hierarchy. Although, it was concluded that this method is not statistically advantageous over the previous methods, it was expected that further study of combined approaches will yield better results.

The ideas proposed in this thesis could be considered a combination of syntactic and symbolic approaches as well. However, instead of using lexico-syntactic patterns to create relationships between words, the actual link structure of the web pages themselves is used. Also, collocation statistics by appearance in links is used to determine some relationships between sibling terms.

3. THEORY AND DESIGN

3.1. Introduction

This thesis focuses on using only the link structure and anchor text in a set of domain specific sites in order to extract a hierarchy. It does not propose to create complete or perfect hierarchies, but rather to show that the largely ignored link structure of a site contains valuable information for taxonomy building. Generally website designers attempt to organize their sites for easy human access and navigation to the topics in their site. Therefore, there is information inherent in the way the pages in a site are linked and by examining the link structure, anchor text and URL itself, a taxonomy can be created. Using this information only, this thesis proposes methods to map a site and organize the pages from "most generic" to "most specific". Then the relationships in this tree are examined to create a taxonomic tree.

The methods discussed in this thesis do not attempt to create complete taxonomies from the topic domains in question. These methods do, however, show that using the link structure of sites is a valuable addition to the variety of techniques already employed in automatic taxonomy creation. Two applications were developed, *SiteGraph* and *SiteMap*, in order to create taxonomies using the ideas discussed in this thesis. These applications generated taxonomies from three different topic domains, NEWS, SPORTS, and COMPUTER HARDWARE, that were evaluated using several different measures including human judges. Theses taxonomies compared favorably with taxonomies generated using other means. Therefore, this thesis shows that the information contained in the link structure of webpages can and should be used in automatic taxonomy generation.

3.2. Notation

Terms

A term, defined as a string of characters, will be referred to with a lowercase *t* or variant thereof (*t′* for instance).

Link

A link, defined as the <a> tag in an html document, will be referred to with a lowercase *l* or variant thereof (*l′* for instance). A link has the following properties:

l.text	The text, or anchor text, is the set of terms, *t*, that falls between the opening <a> tag and the closing </a> tag. The text is filtered according to a procedure to be described later.
$[\{l_1,l_2,l_3 \ldots l_n\}]$	Given a set of links [] returns the set of terms from the text of each link. $[\{l_1, l_2, l_3 \ldots l_n\}] = l_1.text \cup l_2.text \cup l_3.text \cup \ldots \cup l_n.text$

Webpage

A Webpage, defined as a single document as shown by a normal web browser, will be referred to with a lowercase *p* or variant thereof (*p′* for instance). A page has the following properties:

p.url	The URL for this web page.
p.domain	The fully qualified domain name from the URL. For instance, the domain of the url http://www.google.com/index.html is www.google.com.
p.depthrating	The depth rating is a natural number between 0 and 1 which gives a relative indication of how general or specific p's contents are. See Section 3 for a specific description of how a depth rating is calculated.
p.linksout	The set of links contained in webpage p.

$D(p, p')$	The D function takes two pages and returns the minimum link distance between the two pages. Since this is a real metric, it has the following properties: $D(p_1, p_2) = 0$ $D(p_1, p_2) = D(p_2, p_1)$ $D(p_1, p_2) \leq D(p_1, p_3) + D(p_3, p_2)$ The distance from a page to itself is always zero. The direction of the links does not affect distance and the distance from page to another is always the minimum link distance.

WebSite

A website is defined as a set of Webpages that are a certain link distance from the "root" page and belong to the same domain. A website will be referred to with a lowercase *w* or variant thereof (*w'* for example). The set of all websites used in an algorithm will be represented by a capital *W*.

w.root	The "root" page of the website. This is not necessarily the "home" page or what might be considered the root page by a human viewer of the website. See Section 3 for more information on how the root pages are obtained.3
w.pages	The set of pages whose domain match w.root and are reachable within δ links or less. Formally, $w.pages(\delta) = \{p \mid D(p, w.root) \leq \delta \wedge p.domain = w.root.domain\}$ The δ value is used to limit the resource usage of the crawler to a reasonable amount.
pages(*W*)	The pages() function takes a set of Websites, *W*, and returns the union of their *pages* set. Formally, $pages(W) = w_1.pages(\delta) \cup w_3.pages(\delta) \cup \ldots \cup w_n.pages(\delta)$

Notation Refinement

Now that all of our concepts are defined we can expand the definition of the Link and Webpage.

A Link, *l*, also has the following characteristics:

l.to	The webpage pointed to by the link indicated by the contents of the href attribute.

l.from	The webpage containing the link.

A Webpage, *p*, also has the following characteristics:

p.website	The website that this page belongs to. Note that the website's domain matches p.domain. Formally, $p.website = w : w.root.domain = p.domain$
p.pagesto	The set of all pages that the links in p point to and that belong in the same website as p. Formally, $p.pagesto = \left\{ \begin{array}{l} p' \mid \exists l : l.from = p \wedge l.to = p' \wedge \\ p.website = p'.website \end{array} \right\}$
p.linksin	The set of all links pointing to p by other pages in the same website as p. Formally, $p.linksin = \{ l \mid l.to = p \wedge l.from \in p.website.pages \}$
p.pagesfrom	The set of all pages that contain links that point to p and belong to the same website as p. $p.pagesfrom = \{ p' \mid \exists l : l \in p.linksin \wedge l.from = p' \}$
p.terms	The set of terms used in anchor text for anchors that link to p. Formally, $p.terms = \{ t \mid t \in [p.linksin] \}$
count(*p*,*t*)	Given a page p, and a term t, this is the total number of times t appears in [p.linksin]. This considers the text in anchors only and not the rest of the page content.

Relationships

The terms from the anchor text and link structure of the Website are used to create a relationship with the following properties.

A relationship is a set of tuples with three attributes:

(*parent-term*, *child-term*, *count*)

parent-term	The parent term or more general term in the taxonomy.
child-term	The child term or more specific term in the taxonomy. There is only one tuple for each child-term and parent-term combination. If a tuple appears then the following is true about the parent-term and child-term: $\exists p \exists p' : p \in W \wedge p' \in W \wedge p' \in p.pagesto$ $\wedge\ r.parentterm \in p.terms$ $\wedge\ r.childterm \in p'.terms$

count	A natural number from 0 to infinity indicating the number of times the parent-term and child-term appear in the above relationship. The contribution of the pages is normalized by the number of terms in the page.
parent_term_support(*r*, *R*)	Given a tuple r and a relationship R where r Є R, the parent_term_support is a number between 0 and 1 that represents the percentage of all tuples r′ with the same parent-term as r, r.parent-term. Formally, $\text{parent\_term\_support}(r, R) = \frac{r.count}{\sum_{r' \in R \mid r'.parentterm = r.parentterm} r'.count}$ In other words, this is the tuple's count divided by the sum of the counts of the tuples where the parent term matches r.parent-term.
add (*r*, *R*)	Given a tuple r and a relationship R, if there already exists a tuple in R with a matching parent-term and child-term then r.count is added to the existing tuple. Otherwise r is added to the relationship R. Formally, `add(r, R):` `if` $\exists r' : r' \in R$ $\wedge r'.parentterm = r.parentterm$ $\wedge r'.childterm = r.childterm$ `r'.count += r.count` `else` `R = R U {r}`

3.3. Creating the Relationships

All websites that contain a significant amount of content require a link structure to allow navigation from the highest level documents, the site's root or home page, to the lowest level, topic specific documents. For example, a site with information about the C programming language may have a home page, topic pages about keywords, function libraries, etc. and then specific pages detailing one keyword, one function etc. Organizing the page from most generic to the most specific sets up the tree from which the term hierarchy is eventually derived. This functionality is encapsulated by the *SiteMap* application discussed in Section 4.1.

The process for developing the taxonomy is shown in Figure 3-1.

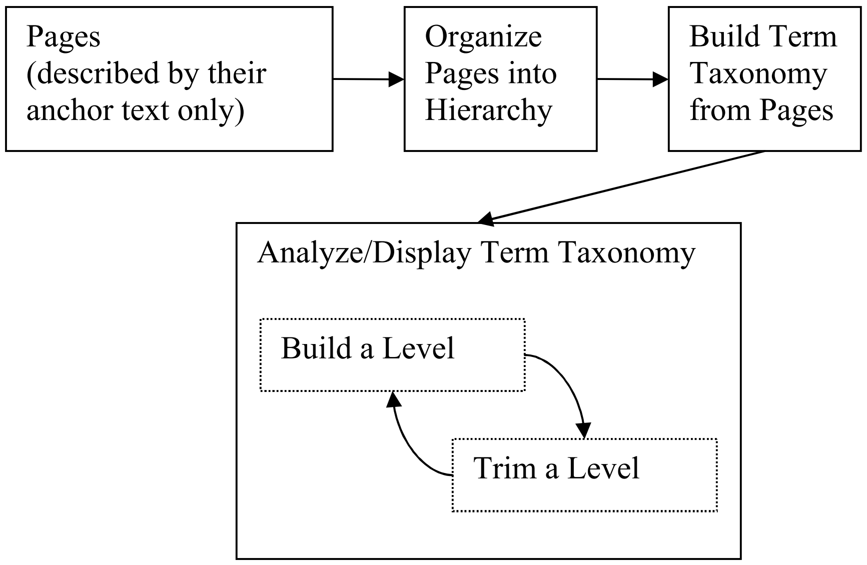

Pages
(described by their anchor text only)
Organize Pages into Hierarchy
Build Term Taxonomy from Pages
Analyze/Display Term Taxonomy
Build a Level
Trim a Level

Figure 3-1 Overview of taxonomy creation

3.3.1. Crawling the Pages

First a set of websites that define a certain domain are chosen and downloaded. The set of pages retrieved by the crawler from each WebSite is formally defined as:

$$w.pages(\delta) = \{p \mid D(p, w.root) \leq \delta \wedge p.domain = w.root.domain\}$$

Recall that the function D() is defined in Section 3.2 as being the shortest link distance between two pages. The value δ is set experimentally.

A crawler is used to navigate each site and store information regarding the links, pages, and anchor text. For each webpage, all of the incoming and outgoing links with their anchor text is recorded. In order to limit the amount of a website the crawler explores and to limit the amount of time it ran, several rules are employed, as explained below.

Rule 1: The crawler limits itself only to the domain of the root webpage. In an attempt to keep the crawler obtaining pages in the domain we are interested in creating a hierarchy for, the simplest solution is to keep it inside of the website it is traversing.

Rule 2: In order to keep the crawler from spending too much time on a single website, only pages found in the first δ levels of the link structure from the root are used. Because the sites used tended to be very large, crawling a single one would take too long to allow for efficient use of time. Also, even though only δ levels are used, most sites still return thousands of Webpages.

Rule 3: To keep the crawler focused on pages that contain domain specific content and away from pages that are specific to site structure or maintenance, links with the following text in the URL or anchor are not followed:

```
forum
site
newsletter
search
archive
blog
rss
```

After we have mapped each website, the crawled Webpages are organized from most general to most specific in a hierarchy. The webpage hierarchy will in turn be used to create the term hierarchy as described below.

3.3.2. Organizing the Pages

Next, we will reorganize the graph of Webpages created by the site's link structure into a tree. This will require an algorithm that will attempt to organize the pages by content (from most general at the root to most specific at the leaves). After all of the pages are downloaded and stored, the first step is to calculate a depth rating for each page which gives an indication of how deep in the tree it should appear. The depth rating for each page is calculated as follows:

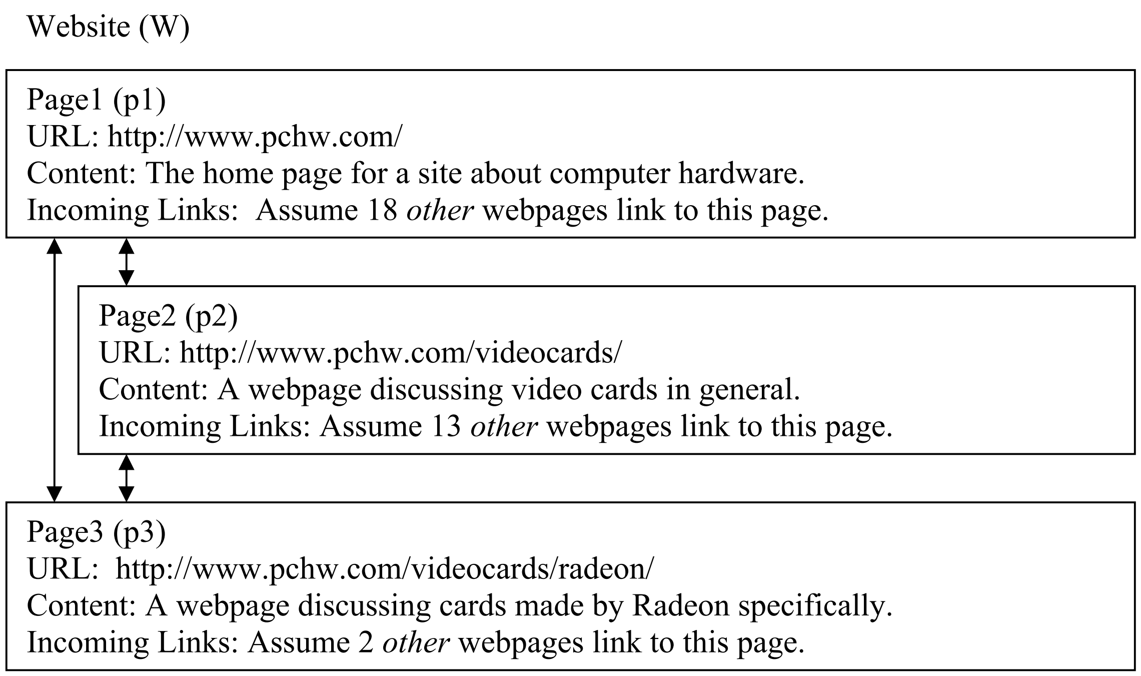
$$p.depthrating = \alpha * \left(\frac{p.url.length}{\max_{p' \in p.website.pages} p'.url.length} \right) + (1 - \alpha) * \left(1 - \left(\frac{|p.linksin|}{\max_{p' \in p.website.pages} |p'.linksin|} \right) \right)$$

where length is a function that returns the number of characters in the URL. In other words, the above formula returns the sum of the length of the page's URL divided by the longest URL in the website and the page's number of incoming links divided by the number of incoming links of the page with the most incoming links. These two factors that make up the depth rating are also weighted using the parameter α whose value is determined experimentally.

Then, after the depth rating for each page is calculated, the pages are organized (using the depth rating) in a tree structure where each node represents one Webpage. We start with the node that represents the root of the Website. All of the Webpages that the root Webpage links to are added as children of the root node in the tree. Then the tree is traversed and every Webpage node in the tree that doesn't yet have children has the Webpages it links to added as children in the tree. This process is repeated until all of the Webpages in the site end up in the tree.

We note, however, that a webpage can only exist *once* in the tree. Therefore, when a webpage is reached that already exists in the tree, the algorithm uses the depth rating to see if it should move the webpage node or leave it where it is. The page is moved only if two conditions are satisfied. First, the page has to be moving farther down the tree, towards the leaves, away from the root. Second, the depth rating of the webpage to be moved has to be significantly larger than the depth rating of the webpage node it would be moved under.

For example, consider a Website about Computer Hardware which contains three Webpages as shown in Figure 3-2. The example below is contrived and the URLs involved are not known to exist and should not be followed:



Page1 (p1)
URL: http://www.pchw.com/
Content: The home page for a site about computer hardware.
Incoming Links: Assume 18 *other* webpages link to this page.

Page2 (p2)
URL: http://www.pchw.com/videocards/
Content: A webpage discussing video cards in general.
Incoming Links: Assume 13 *other* webpages link to this page.

Page3 (p3)
URL: http://www.pchw.com/videocards/radeon/
Content: A webpage discussing cards made by Radeon specifically.
Incoming Links: Assume 2 *other* webpages link to this page.

Figure 3-2 Three pages in an example website

```
w.root = p1
w.pages = {p1, p2, p3, ..., pn}

p1.url = http://www.pchw.com/
p1.url.length = 20
p1.pagesfrom = {p2, p3, ..., pn}
|p1.linksin| = 20

p2.url = http://www.pchw.com/videocards/
p2.url.length = 31
p2.pagesfrom = {p1, p3, ..., pn }
|p2.linksin| = 15

p3.url = http://www.pchw.com/videocards/radeon/
p3.url.length = 38
p3.pagesfrom = {p1, p2, ..., pn }
|p3.linksin| = 4
```

First, we need to calculate the depth rating of each page.

```
max(|w.pages.linksin|) = 20
max(w.pages.url.length) = 38

p1.depthrating = .6 * (20 / 38) + .4 * (1 - (20 / 20)) = .6 * .526 + .4 * 0   = .316
p2.depthrating = .6 * (31 / 38) + .4 * (1 - (15 / 20)) = .6 * .816 + .4 * .25 = .589
p3.depthrating = .6 * (38 / 38) + .4 * (1 - (4 / 20))  = .6 * 1    + .4 * .8  = .920
```

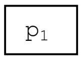
Next we need the standard deviation of the depth ratings:

```
DepthStdDev = .303
```

Now we will execute the logic in the tree building algorithm designed to build a tree from the most general webpage to the most specific webpage:

Step 1:

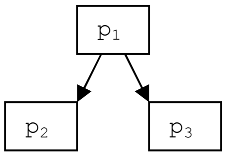
```
Q = {$p_1$}
T  =
```



```
       for each Webpage pChild in $p_1$.pagesto
       $p_2$ does not exist in the tree
              therefore it is made a child of $p_1$ and placed in the queue
       $p_3$ does not exist in the tree
              therefore it is made a child of $p_1$ and placed in the queue
```

Step 2:

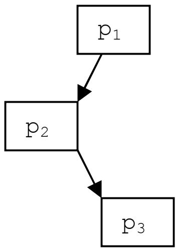
```
Q = {$p_2$, $p_3$}
T =
```



```
for each Webpage pChild in $p_2$.pagesto
$p_1$ does exist in the tree
              $p_2$.depthrating + DepthStdDev < $p_1$.depthrating
              .589 + .303 < .316  (FALSE)
$p_3$ does exist in the tree
              $p_2$.depthrating + DepthStdDev < $p_3$.depthrating
              .589 + .303 < .920  (TRUE)
              therefore it is removed from $p_1$ and made a child of $p_2$
```

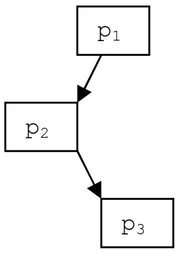
Step 3:

```
Q = {$p_3$}
T =
```



```
for each Webpage pChild in $p_3$.pagesto
$p_1$ does exist in the tree
              $p_3$.depthrating + DepthStdDev < $p_1$.depthrating
              .920 + .303 < .316  (FALSE)
$p_2$ does exist in the tree
              $p_3$.depthrating + DepthStdDev < $p_2$.depthrating
              .920 + .589 < .589  (FALSE)
```

Completed Tree:



Due to the length of the url and the number of incoming links, the webpage about Radeon brand video cards is moved down under the webpage about video cards in general. Although this example is simplistic, it demonstrates how the depth rating is calculated and used to organize the Webpages of websites from most general to most specific.

If, however, p_3 had a larger than expected number of incoming links, it would be feasible that p_3 would not be moved down below p_2. For example, if *p3* had 40 incoming links:

```
max(|w.pages.linksin|) = 40
max(w.pages.url.length) = 38

p1.depthrating = .6 * (20 / 38) + .4 * (1 - (20 / 40)) = .6 * .526 + .4 * .5   = .516
p2.depthrating = .6 * (31 / 38) + .4 * (1 - (15 / 40)) = .6 * .816 + .4 * .625 = .739
p3.depthrating = .6 * (38 / 38) + .4 * (1 - (40 / 40)) = .6 * 1    + .4 * 0    = .600

DepthStdDev = .113
```

In this case, even though p_3 should be under p_2, the situation would be reversed and p_2 would be under p_3 because the *DepthStdDev* is less than the difference in their *depthratings*. However, in this case, it requires that p_3 have nearly three times the number of incoming links which would be unlikely if the content in p_3 truly was more specific.

3.3.3. Calculating Term Relationships

The final step in creating the term relationships is to process the tree created in the above step and, using the Webpage `terms` set (defined in Section 3.2), create a set of relationships. The terms are the actual strings of characters representing a word found in an anchor link referring to this webpage.

The tree is traversed and for each parent-child connection in the tree, a set of tuples is generated to be added to the relationship. A tuple is created for every combination of terms in the parent webpage's *`terms`* set and child webpages *`terms`* set. These sets contain every term in any anchor text that is pointing to each page. Therefore, if a parent page has the terms `{computer, pc}` and has two child webpages with the term sets `{motherboard, memory}` and `{sound, card}` then tuples are created with the following parent and child terms:

```
computer-motherboard
computer-memory
computer-sound
computer-card
pc-motherboard
pc-memory
pc-sound
pc-card
```

As the terms are added, they are normalized so that the contribution of each child webpage was weighed equally. This simply means that the count of each term was divided by the total number of terms that refer to this webpage. For instance, if a treenode has two child treenodes with pages that contain the list of terms and counts in Table 3-1,

Child Webpage 1		**Child Webpage 2**	
child.term	**child.count**	**child.term**	**child.count**
motherboard	24	sound	3
memory	10	card	1

Table 3-1 Example child counts before normalization

then the first webpage would contribute significantly more than the second webpage simply because it had more links referencing it. Therefore, we normalize the contribution by dividing the first Webpage's counts by 24 + 10 = 34 and the second Webpage's counts by 3 + 1 = 4. This yields:

Child Webpage 1		Child Webpage 2	
motherboard	0.705882	sound	0.75
memory	0.294118	card	0.25

Table 3-2 Example child counts after normalization

Therefore, both child Webpages contribute equally to the terms of the webpage of the current treenode.

When creating a relationship (set of tuples described in Section 3.2), each website contributes using the above process. The pages in the site are first organized using the depth rating, and then the tree is traversed and has the relationships added using the process described in this section. When a website generates a tuple, it is added to the relationship using the **add(*r*, *R*)** described in Section 3.2. Therefore, webpages contribute to the overall taxonomy by adding individual tuples to the relationship. If two webpages generate the same tuple, then it is strengthened by adding the counts together. In this fashion, the relationship is created from all of the contributing websites to be analyzed in the following sections.

3.4. Analyzing the Relationship to Create the Term Hierarchy

Now that the relationships have been created, the next step is to create the term hierarchy. A variety of parameters that directly influence its size and shape have to be adjusted to provide meaningful term hierarchies. These parameters control tree depth, trimming, allowed terms, the root term, and a large number of other factors. All of these parameters and this functionality are encapsulated in the *SiteGraph* application discussed in Section 4.2.

3.4.1. Terminology

term_anchor_support(t): The *term anchor support* is simply the proportion of all pages that a given term appears in (anchor text only). Given a term *t*:

$$\frac{\left|\{p \mid p \in \mathrm{pages}(W) \wedge t \in [p.linksin]\}\right|}{\left|\mathrm{pages}(W)\right|}$$

Note that a capital *W* is used indicating the set of all pages of all websites crawled to create the relationship *R*.

Instance Terms: *Instance terms* (sometimes referred to as instance children when a specific child/parent relationship is being discussed) are terms that use only a relationship percentage value compared to a cutoff value to determine if they are valid. *An instance term intends to represent a specific instance of a broader subject.* For example, in the SPORTS domain a player's name such as "BRYANT" could be considered an instance term.

Topic Terms: *Topic terms* (sometimes referred to as topic children when a specific child/parent relationship is being discussed) are terms that use both a *relationship percentage* value and the *term_anchor_support* as measures to determine validity. *A topic term intends to represent a subject that can be broken down into smaller instances.* For instance, in the SPORTS domain a sport such as "BASKETBALL" could be considered a topic term.

Modifier Terms: Referring to terms as "*modifiers*" simply indicates that they co-occur more frequently in the same link's anchor text than a given threshold. They are searched for and displayed in results simply to give the user more insight into the relationships between terms. For example, in the COMPUTER HARDWARE domain the term "SOUND" could be considered a modifier of the term "CARD".

topic_rank: The *topic rank* of a term *t*, given a parent *p*, is the *term_anchor_support* plus the *parent_term_support* (see Section 3.2) where *t* is the child-term and *p* is the parent-term. It is used in generating topic terms. Formally,

$$\texttt{topic\_rank}(p, t) = \texttt{term\_anchor\_support}(t) + \texttt{parent\_tem\_support}(R, r)$$

where *R* is the relationship created from the webpages and *r* is the tuple containing t as the child-term and *p* as the parent-term.

3.4.2. Input Parameters

root-term - This is the term chosen to be the root of the hierarchy.

max-levels - The maximum number of levels keeps the algorithm from continuing infinitely. Generally, the other trimming parameters will keep the hierarchy from reaching the maximum number of levels. However, this value is included as a fail safe.

instance-cutoff - The instance-cutoff is used when searching for child instance terms for a parent term. They are generated by comparing the *instance-cutoff* value with the *parent_term_support* value. For instance, if the term "CARD" appears as the parent-term in the tuples in *R*, listed in Table 3-3.

parent-term	child-term	count	parent_term_support
CARD	VIDEO	0.52	0.07
CARD	AUDIO	2.88	0.41
CARD	RADEON	1.77	0.25
CARD	GEFORCE	0.31	0.04
CARD	PCI	1.5	0.21

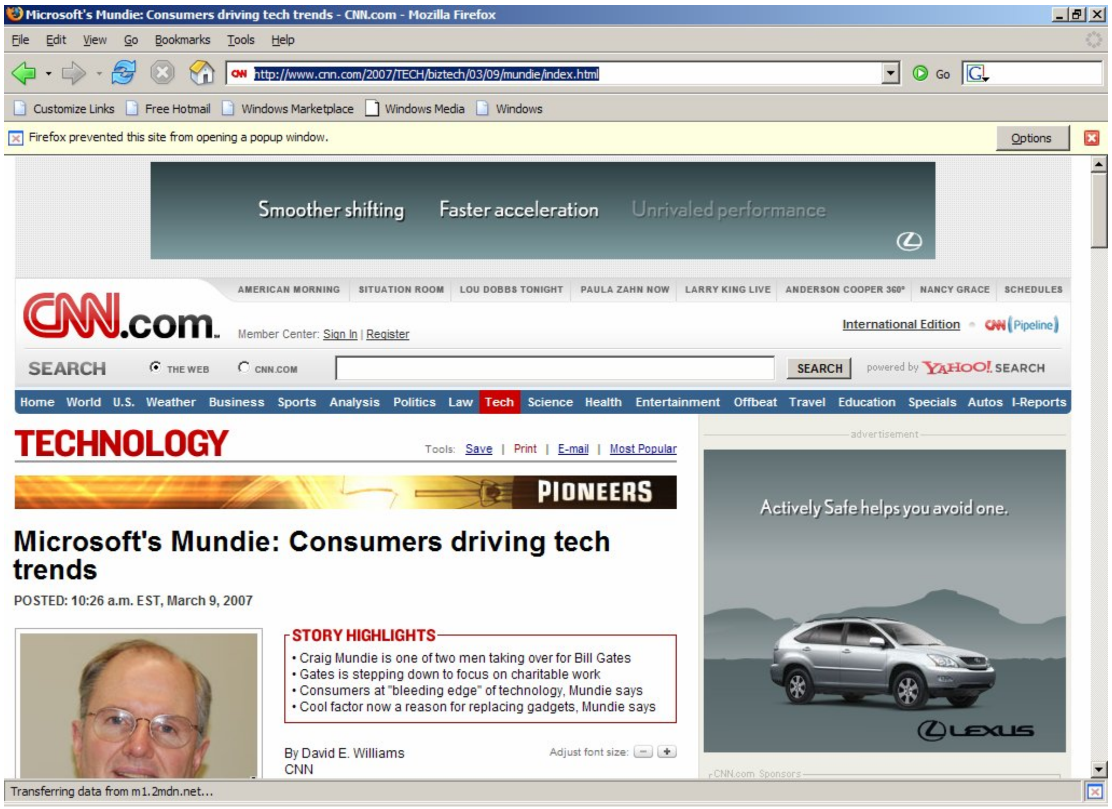
Table 3-3 Example children demonstrating the use of the instance-cutoff parameter

and the *topic-cutoff* is 0.2 then PCI, AUDIO, and RADEON would be retrieved as child instance terms.

top-topic-percent - This parameter controls what percent of all potential topics actually become topics. It defaults to ten percent. See below for more information.

topic-cutoff - The topic cutoff is used when searching for child topic terms for a parent term. They are generated by comparing the *topic-cutoff* value with the *parent_term_support* value. The values are compared exactly like the *instance-cutoff* value, except the relationships are also ordered by *topic_rank* and only the *top-topic-percent* of terms are returned. Because only the *top-topic-percent* of candidate topics are retained the *topic-cutoff* tends to be an order of magnitude less than the *instance-cutoff*.

Sorting the pages in this manner causes those terms appearing in the largest percentage of pages to be favored for topics. This method was used (and works) because the links that tend to contain "topic" terms are those that appear as navigational links on every page in a site. For instance, when navigating http://www.cnn.com (as of March, 2007) there is a blue bar appearing at the top of every page allowing instant navigation to the main topics of the site: World, U.S., Business, Sports, etc. Refer to figure 3-3.



Firefox prevented this site from opening a popup window.

Microsoft's Mundie: Consumers driving tech trends

POSTED: 10:26 a.m. EST, March 9, 2007

STORY HIGHLIGHTS
- Craig Mundie is one of two men taking over for Bill Gates
- Gates is stepping down to focus on charitable work
- Consumers at "bleeding edge" of technology, Mundie says
- Cool factor now a reason for replacing gadgets, Mundie says

By David E. Williams
CNN

Figure 3-3 Example webpage from CNN.com (March, 2007) hosted in the Firefox web browser

Because this design structure is used so frequently, the *term_anchor_support* of a term is a very powerful indicator of whether or not this term is a good topic. Returning to our example, Table 3-4 shows the child terms with the *term_anchor_support* added and sorted by a *topic_rank*:

parent-term	child-term	count	parent_term_ support	term_anchor_ support	topic_rank
CARD	AUDIO	2.88	0.41	0.48	0.89
CARD	VIDEO	0.52	0.07	0.52	0.59
CARD	PCI	1.5	0.21	0.23	0.44
CARD	RADEON	1.77	0.25	0.12	0.37
CARD	GEFORCE	0.31	0.04	0.09	0.13

Table 3-4 Example children sorted by topic_rank

Using this methodology AUDIO and VIDEO are the two most likely candidates for topic children of the parent term "CARD" based on the topic_rank.

topic-trim-cutoff - The algorithm first attempts to generate topic children for a given term. If, however, during the trimming phase a greater percentage of topics are removed than the *topic-trim-cutoff***,** then all of the topic children are removed and are replaced with instance children using the above methodologies.

modifier-cutoff - This parameter determines what percentage of a term's total appearances have to co-occur with another term for it to be considered a modifier term. In this case, co-occurrence is defined as appearing in the same anchor text. Creating modifier terms is part of the trimming phase and is designed to reveal more information about the relationships between sibling terms.

Using the above input parameters, the taxonomy is created by building one level at a time and then trimming that level until either *max-levels* is reached or all of the leaf nodes are instance nodes (See Section 3.4.4 to find how instance nodes are created).

3.4.3. Building a Level

We start by adding the *root-term* as the root node of the tree and a Topic node. Then we build each level by traversing the tree and creating child topic nodes using the *topic-trim-cutoff*. After each level is built it is trimmed using the below process. The trimming process determines if a Topic node makes a significant contribution to the taxonomy. If it does not then the Topic children are removed and replaced with Instance children. Therefore, because the building process does not build off of instance children, the taxonomy creation process will eventually end when there are no more topic leaf nodes.

3.4.4. Trimming a Level

After a level is built, the algorithm uses a series of trimming rules that remove and move TreeNodes. There are three basic trimming rules that are applied to each level.

Topic Trimming

The first kind of trimming removes topic children, and, if certain conditions are met, it removes all topic children of a node and replaces them with instance children. The hierarchy does not build off of instance children so if a node's child topic terms are replaced with child instance terms, then that will be the last level of the tree on this branch.

First, the trimming logic checks how many unique topic children a node brings into the tree. If a node fails to bring any new topics to the entire tree, it has its child topic nodes removed. Next, if the topic node provides at least one new topic to the tree, it then removes all topics that already appear in a higher level. If it removes a larger percentage than the *topic-trim-cutoff* parameter, or only has one topic child left then it will have all of its topic children removed. Then, all of those nodes who had their Topic children removed, have them replaced with Instance children. Instance children are generated as described above.

Modifier Trimming

The next kind of trimming, identifies "modifier" nodes among sibling terms and decides if one should be moved over as a modifier of another. Each node on the newly created level is checked against each of its sibling terms to see if its co-occurrence in anchor text with every other term divided by its total appearances is greater than the *modifier-cutoff* parameter. This is performed for both instance and topic nodes. If this is true then the term is moved over as a "modifier" of the other term.

	Appearances
power	200
supply	100
co-occurrence:	80

Table 3-5 Co-occurence example

In the example, shown in Table 3-4, supposing the *modifier-cutoff* is set to .5, then *supply* would be considered a modifier term of *power* because 80% of the times *supply* appeared in anchor text it appeared with *power*. However, *power* would not be considered a modifier of *supply* because only 40% of its appearances in links co-occurred with *supply*. This is equal to the conditional probability of the terms. If,

$$p(A \mid B) > \mathit{ModifierCutoff}$$

is true then B is considered a modifier of A.

CrossTopic Trimming

Sometimes a child term will appear across so many topic terms, that expanding it further would cause huge redundancies in the tree. Therefore terms that appear across a sufficient number of topic nodes will be considered “cross topic”. In order to find cross topics nodes, a given term’s appearances on a given level is compared to the average appearances by the other nodes. If a term is two standard deviations away from the average term appearance at that level it is to be considered a cross topic node because only those terms that were true outliers should be moved up as a cross topic node. During experimentation it was found that a single standard deviation was not restrictive enough and too many nodes were considered cross topic.

For instance, in the tree cross topics are listed in Table 3-6 marked with orange:

Level					
1	**2**	**3**			
				VIDEO	1
	CARD	VIDEO		SOUND	2
		SOUND		USB	3
		USB		ABIT	1
		PCI		NIKON	1
				MEMORY	1
HARDWARE	MOTHERBOARD	ABIT		PCI	1
		USB		DIGITAL	1
		SOUND			

		MEMORY			
	CAMERA	NIKON		Average	1.375
		USB		Std Dev:	0.744024
		DIGITAL		Threshold:	2.863048

Table 3-6 Crosstopic trimming example.

If we are trimming the third level, then USB would be considered a cross topic term. This is because it appears three times in the level which is greater than the average appearance at that level plus two times the standard deviation.

4. IMPLEMENTATION

A series of programs were developed to implement the ideas described in this thesis regarding taxonomy building and information extraction. These programs were developed in C# using Visual Studio .NET 2005 with SQL Server as a back end to store the recovered data. Several externally developed tools were also used such as the Badger Information Extraction Software (http://www-nlp.cs.umass.edu/software/badger.html) developed by the Center for Intelligent Information Retrieval at the University of Massachusetts.

The first program, *SiteMap*, downloads and maps a list of websites and inserts them into an SQL Server database. It is responsible for accurately maintaining the links between all pages in a site, and tracking what words are used in the link text to link from page to page. The second program, *SiteGraph*, reads the data from the database and displays it in a manageable form to the user. It also allows the user to easily modify a variety of constants and threshold values in order to understand what values lead to the best results.

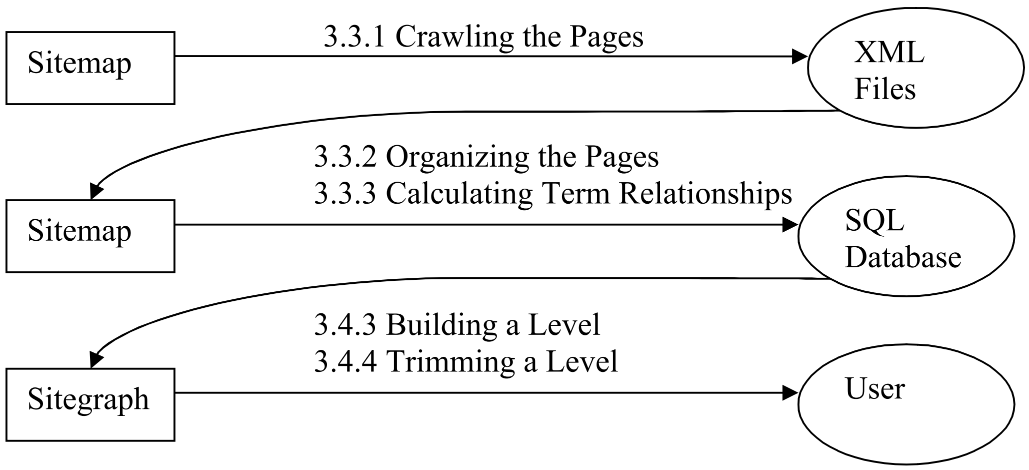
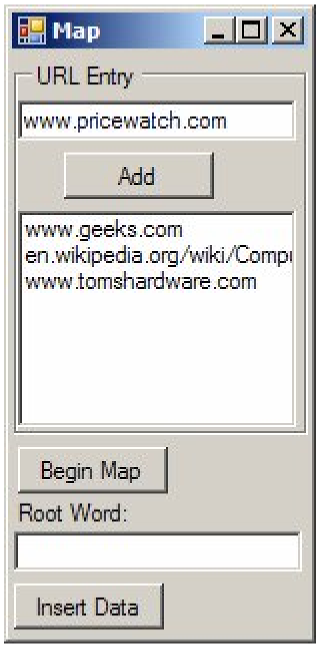


Figure 4-1 Overview of Implementation

4.1. SiteMap

The *SiteMap* application is responsible for mapping a list of sites, arranging the pages in a hierarchy of each individual site from most general to most specific, extracting link text from each page and inserting all relationships, pages and links into the database. This application was developed using C# in Visual Studio 2005.



www.pricewatch.com

www.geeks.com
en.wikipedia.org/wiki/Compı
www.tomshardware.com

Figure 4-2 Screenshot of *Sitemap* as used to map websites

The interface allows the user to add a list of Webpages by entering each site into the URL Entry text box and clicking add. This populates the listbox with each entered URL. When the user clicks "Begin Map" the program will then map each website in order and save the data in XML files. The websites used were determined by searching Google using the topics we intended to create taxonomies for and using the first five results. For a given website the mapping process can take anywhere from half an hour to two or three hours depending on the number of links found.

When the user clicks "Insert Data" the program will open each indicated xml file (created in the Mapping Phase), create a hierarchy of pages, and insert the relationships into the database using the contents of the Root Word text box as the root of each website. It generally takes about half an hour to an hour per site to insert all of its relationships into the database.

4.1.1. Mapping and Parsing

This phase of the program happens when the user clicks the Begin Map button. Each list entry is the root (`w.root`) of a website (`w`). For each Url in the listbox *SiteMap* follows all of the mapping specifications explained in Section 3.3.1. This includes the depth restrictions, staying in the same domain as the w.root page and not following links that contain specific stopwords.

The anchor text of each link in each webpage is subjected to a cleaning process to extract only the nouns as terms to be added to the XML file using the following steps:

1) Replace common escape sequences with the characters they represent:

```
&quot => "
&amp  => and
&lt   => <
&gt   => >
&nbsp => <space>
```

2) Remove any html tags that may fall in the anchor text <*>.

3) Pass the text through Badger Information Extraction Software (http://www-nlp.cs.umass.edu/software/badger.html) and only use the terms identified as nouns by the software.

4) Remove any terms that match the following terms. First remove pronouns:

```
i,me,mine,you,you,your,he,him,his,she,her,hers,it,its,we,us,our,ours,you,,you,yours,they,
them,their,theirs
```

Also, remove artifacts left by badger: `%poss%,@@`. `%poss%` is created by Badger to indicate a term is possessive. `@@` is created by Badger to indicate a word indicating a time span or position in time. i.e. yesterday, March 13th, 4:00 PM, etc.

5) Remove common endings:

```
ies => y, s (unless the term ends in a double s), ing
```

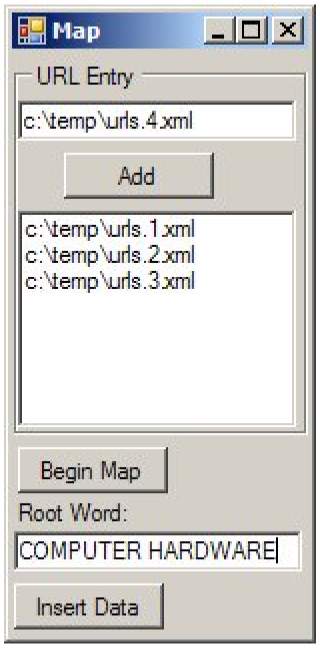
Originally the nouns were then passed through the Porter Stemming algorithm to remove common endings and combine singular with plural forms. However, it was determined that the Porter Stemming algorithm was too aggressive and combined terms that did not make sense in the context of taxonomy building. For instance, it combines the terms "DIGITAL" and "DIGIT" whose English meaning is very different. Therefore, a simple, custom algorithm was written to only change plural forms to singular and remove the "ing" ending. The set of terms that is retrieved from Badger is now run through this simpler algorithm. This leaves a list of stemmed nouns to be stored as the set of terms for a given link. This set of terms is the same set that is referred to in Section 3.2 and is retrieved using the [] syntax: [*p.linksin*].

Each website is stored as its own XML file with its Webpages, urls, links, and processed anchor text. There is only one website per file. The XML contains, for each mapped webpage, its URL and an id field. The webpage node has a `<linksto>` node which contains a list of all pages it links to and the number of times it links to that page. The `<linksfrom>` node contains a list of all pages that this page is linked from and all of the terms used in each link. The `<terms>` node contains all of the terms used to refer to this page (obtained from anchor text in the `<linksfrom>` nodes) and how many times they were used to refer to this page. The `<TotalCount>` node is the total number of terms used to refer to this page. This XML files is then parsed by the same program in a second run when the user clicks “Insert Data”. See below for the DTD for the xml file and refer to the appendix for a sample of the file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Count (#PCDATA)>
<!ELEMENT Term (Text, Count)>
<!ELEMENT Terms (TotalCount, Term*)>
<!ELEMENT Text (#PCDATA)>
<!ELEMENT TotalCount (#PCDATA)>
<!ELEMENT count (#PCDATA)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT link (pointsto, count, terms)>
<!ELEMENT linksfrom (link+)>
<!ELEMENT linksto (link*)>
<!ELEMENT page (url, id, linksto, linksfrom, Terms)>
<!ELEMENT pointsto (#PCDATA)>
<!ELEMENT root (page+)>
<!ELEMENT term (#PCDATA)>
<!ELEMENT terms (term*)>
<!ELEMENT url (#PCDATA)>
```

4.1.2. Inserting Into the Database

After the user has mapped a set of websites and has saved the XML files using *SiteMap*, they would use the program again to import the XML files into the database. This time the user would enter the XML file names into the listbox and press the “Insert Data” button. The user also indicates the root subject that they are importing:



c:\temp\urls.4.xml
c:\temp\urls.1.xml
c:\temp\urls.2.xml
c:\temp\urls.3.xml
Root Word: COMPUTER HARDWARE

Figure 4-3 Screenshot of *Sitemap* used to insert data into the database

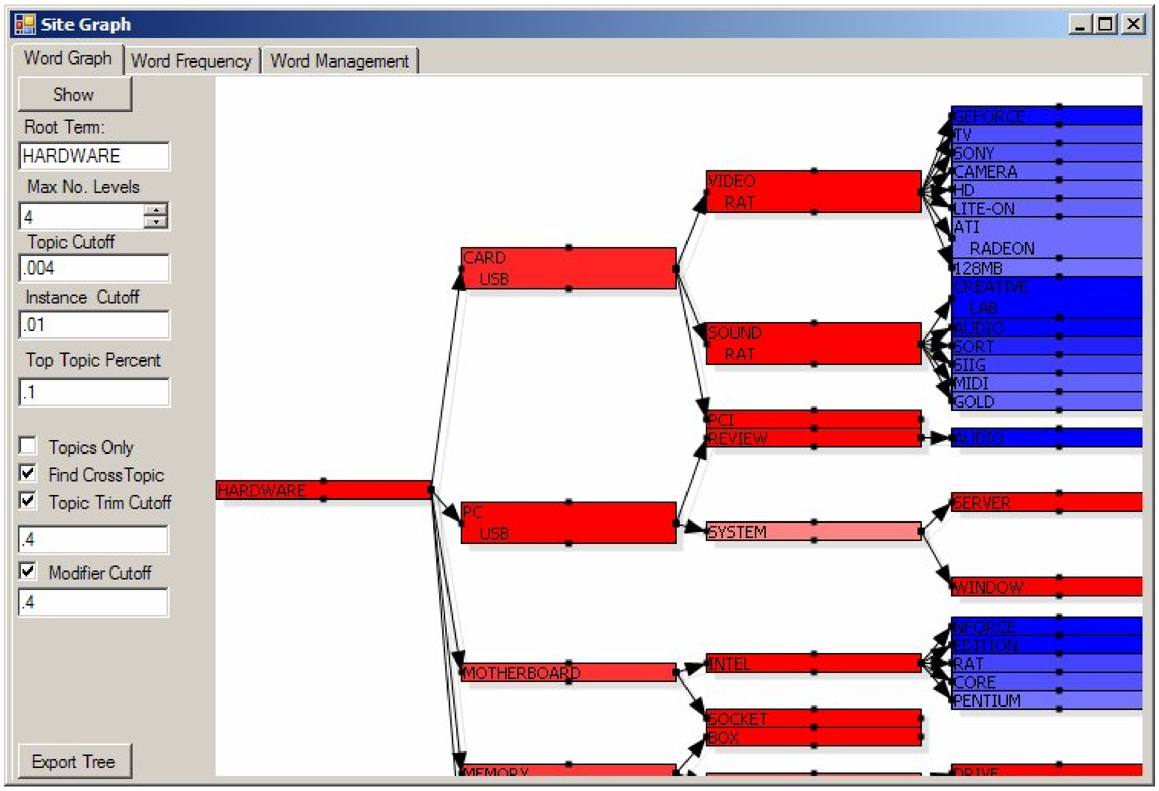
When the user clicks “Insert Data” *SiteMap* imports each XML element one at a time and builds the tree of Webpages from most generic to most specific as detailed in Section 3. Each XML file contains information from only one website.

After the tree of Webpages has been built in memory, *SiteMap* recursively traverses the tree breadth-first and inserts each relationship (*r* in Section 3.2) into the SQL Database for later analysis. *SiteMap* also inserts all links between all Webpages and all processed anchor text into the database in order to help in calculating the term_anchor_support as mentioned previously.

4.2.   SiteGraph

Now that the data has been inserted in the database the *SiteGraph* application provides an interface to examine the data and extract a hierarchy of terms from it based on the link structure of the tree created by *SiteMap* and inserted into the database.

4.2.1. Word Graph



Root Term: HARDWARE
Max No. Levels 4
Topic Cutoff .004
Instance Cutoff .01
Top Topic Percent .1
☐ Topics Only
☑ Find CrossTopic
☑ Topic Trim Cutoff .4
☑ Modifier Cutoff .4

Figure 4-4 Screenshot of the Word Graph tab of *SiteGraph*

The Word Graph tab makes use of the Open Source Netron Light control that continues to be developed under the name Netron Reloaded[1]. This tab displays the tree built from the database using the algorithm described in Section 3. It allows the user to adjust all of the variables that control the tree structure which helps them determine the settings that optimize the results. The tree diagram displays nodes retrieved as topic terms in red and nodes retrieved as instance terms in blue. The darker the shade the larger a percentage the child word appears in the parent word. It generally takes about a minute to generate a taxonomy from the relationships in the database depending on the options selected.

[1] http://sourceforge.net/projects/netron-reloaded

Root Term: This is the term that the user wishes to use as the root node of the tree. Although other terms can be used, the root term specified when the tree is inserted into the database by *SiteMap* is generally used as it contains the richest and deepest tree. This is the same as the *root-term* parameter in Section 3.4.

Max No. Levels: This is the maximum height of the tree. The value is considered a safeguard to keep the program from spending too much time generating the tree and is usually chosen to be larger than the expected number of levels. Most trees generate around three levels of topic terms so a common value for this parameter is four. This is the same as the *max-levels* parameter in Section 3.4.

Topic Cutoff: This is the same as the *topic-cutoff* parameter in Section 3.4. It controls how restrictive the algorithm is when finding Topic Terms.

Instance Cutoff: This is the same as the *instance-cutoff* parameter in Section 3.4. It controls how restrictive the algorithm is when finding Instance Terms.

Top Topic Percent: This is the same as the *top-topic-percent* parameter in Section 3.4. It controls what percentage of potential topics actually become topics.

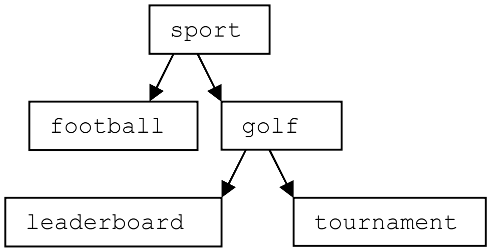
Topics Only: This check box keeps the program from retrieving the instance nodes. This is used to reduce the time to build the tree but still see the main structure. It is always left checked for generating complete taxonomies.

Find Cross Topic: This check box controls whether or not "Cross-Topic" nodes are identified. It is always left checked for generating complete taxonomies.

Topic Trim Cutoff: This check box and the nearby text box control whether or not the instance trimming rules are applied to each node. If they are, then value indicated in the text box indicates the percentage required to be kept in order for the child nodes to be considered topic nodes as described in Section 3.4. It is always left checked for generating complete taxonomies. The value in the textbox is used as the *topic-trim-cutoff* value in Section 3.4.

Modifier Cutoff: This check box and the nearby text box control whether or not all of the child nodes are checked against their siblings to see if their collocation is significant enough to be a modifier. It is always left checked for generating complete taxonomies. The value in the textbox is used as the *modifier-cutoff* value in Section 3.4.

Export Tree: The export tree button exports the displayed tree into a comma delimited .csv file. If the tree is:



then the comma delimited file would be:

```
sport,football,
,golf,leaderboard
,,tournament
```

and it would appear in a spreadsheet editor as:

sport	football	
	golf	leaderboard
		tournament

Context Menu

If the user right clicks on any node in the tree they are given a context menu which gives them tools to further understand and make sense of the underlying data:

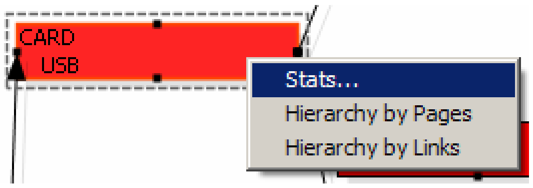
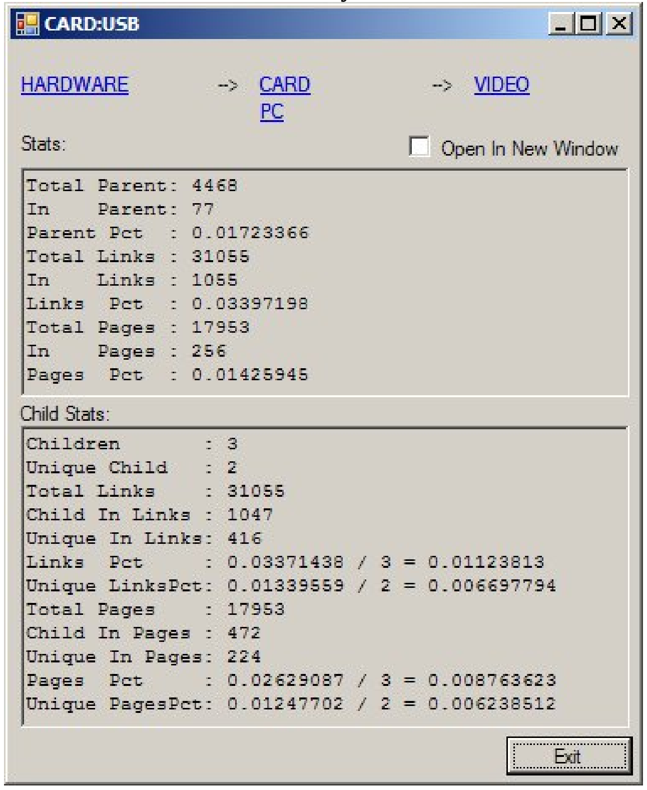


Figure 4-5 Screenshot of a taxonomy node context menu in *SiteGraph*



```
Total Parent: 4468
In    Parent: 77
Parent Pct  : 0.01723366
Total Links : 31055
In    Links : 1055
Links  Pct  : 0.03397198
Total Pages : 17953
In    Pages : 256
Pages  Pct  : 0.01425945
```

```
Children        : 3
Unique Child    : 2
Total Links     : 31055
Child In Links  : 1047
Unique In Links: 416
Links  Pct      : 0.03371438 / 3 = 0.01123813
Unique LinksPct: 0.01339559 / 2 = 0.006697794
Total Pages     : 17953
Child In Pages  : 472
Unique In Pages: 224
Pages  Pct      : 0.02629087 / 3 = 0.008763623
Unique PagesPct: 0.01247702 / 2 = 0.006238512
```

Figure 4-6 Screenshot of the Stats dialog in *SiteGraph*

Stats Dialog

The stats dialog shows a variety of stats regarding the node and the node's children. It also provides links allowing the user to navigate between the node's children, parent, and siblings.

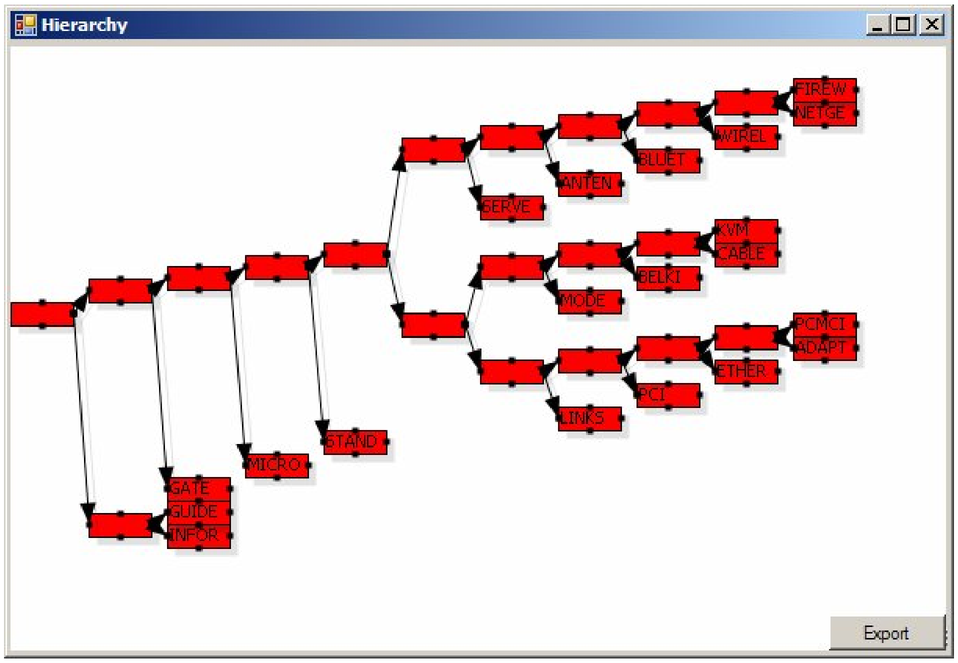


Figure 4-7 Screenshot of the Hierarchy dialog in *SiteGraph*

Hierarchy Dialog

The hierarchy dialog uses a hierarchical clustering algorithm detailed in [8] on the children of the selected node and displays the results. The algorithm begins by assigning each term to its own cluster. It then repeatedly combines the two "nearest" clusters until there is only one cluster remaining. The algorithm uses the group-average distance between two clusters, i.e. the average cosine distance between all combinations of terms from each cluster to determine the total distance between clusters. The distance between two terms is defined as the cosine distance between vector representations of each term. The vectors used are either constructed from the collocation with all other terms in pages or in links (chosen by the user). Two terms collocate in a link if they both appear in the anchor text of the same link. Two terms collocate in a page if they both appear in the anchor text of links that appear on the same page. For instance, if a term t extracted from the anchor text of a link collocates with t_0 24 times, t_1 17 times, t_2 0 times, etc. then the first three values in its vector representation would be [24,17,0,....]. The cosine distance between two vectors is defined as:

$$\cos(X, Y) = \frac{\sum_{x \in X, y \in Y} xy}{\sqrt{\sum_{x \in X} x^2 \sum_{y \in Y} y^2}}$$

In this manner the terms are clustered and displayed to the user.

4.2.2. Word Frequency

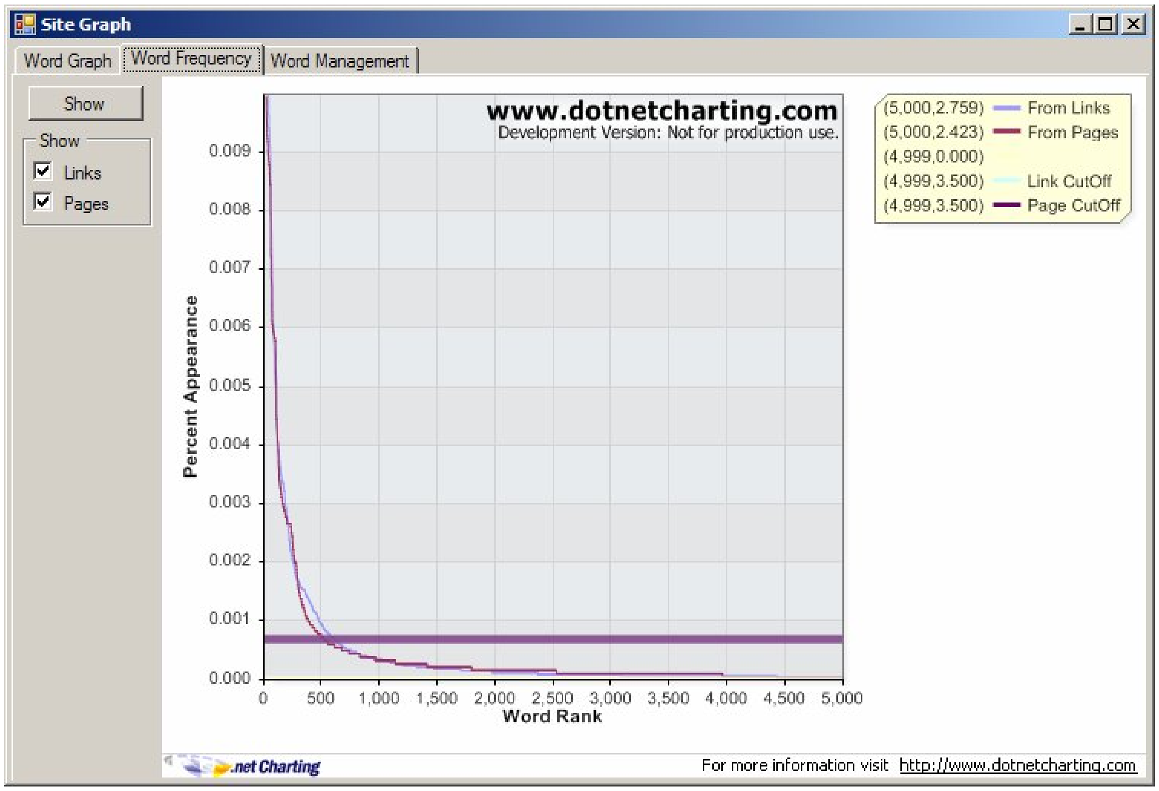


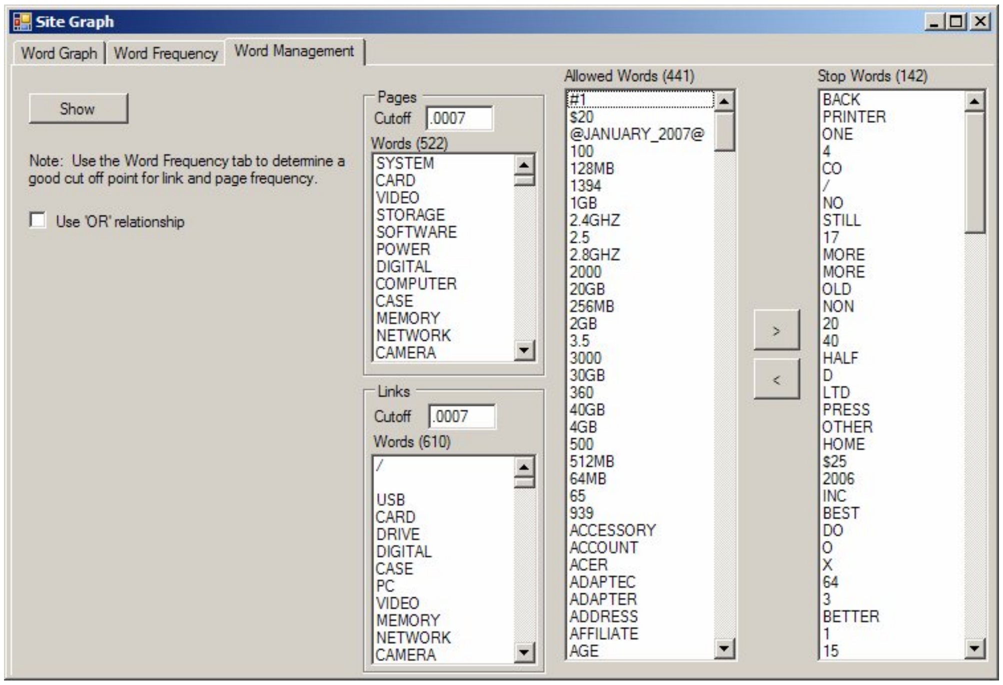
Figure 4-8 Screenshot of the Word Frequency tab in *SiteGraph*

The Word Frequency tab uses a charting object developed by .net Charting[1]. This tab displays the distribution of term appearances in links and pages (a term is considered to appear in a page if it appears in the anchor text in a link in that page) and helps the user choose the cutoff values for links and pages. This tab is used in conjunction with the

[1] http://www.dotnetcharting.com

Word Management Tab which shows the actual words that are allowed for different thresholds.

4.2.3. Word Management



Note: Use the Word Frequency tab to determine a good cut off point for link and page frequency.

☐ Use 'OR' relationship

Pages
Cutoff .0007
Words (522)
SYSTEM
CARD
VIDEO
STORAGE
SOFTWARE
POWER
DIGITAL
COMPUTER
CASE
MEMORY
NETWORK
CAMERA

Links
Cutoff .0007
Words (610)
/

USB
CARD
DRIVE
DIGITAL
CASE
PC
VIDEO
MEMORY
NETWORK
CAMERA

Allowed Words (441)
#1
$20
@JANUARY_2007@
100
128MB
1394
1GB
2.4GHZ
2.5
2.8GHZ
2000
20GB
256MB
2GB
3.5
3000
30GB
360
40GB
4GB
500
512MB
64MB
65
939
ACCESSORY
ACCOUNT
ACER
ADAPTEC
ADAPTER
ADDRESS
AFFILIATE
AGE

Stop Words (142)
BACK
PRINTER
ONE
4
CO
/
NO
STILL
17
MORE
MORE
OLD
NON
20
40
HALF
D
LTD
PRESS
OTHER
HOME
$25
2006
INC
BEST
DO
O
X
64
3
BETTER
1
15

Figure 4-9 Screenshot of the Word Management tab in *SiteGraph*

The word management tab allows the user to see the current stop words and allowed words. It also lets them choose the threshold values for percent appearance in pages and links.

If the "Use 'OR' relationship" checkbox is checked, the union of the terms allowed by pages and terms is used. However, if it is unchecked, the intersection is used.

4.3. Miscellaneous tools

4.3.1. stopwords.vbs

Stopwords.vbs is a vbscript that was written to import the list of stopwords into the database. It opens a text file, stopwords.txt, and inserts all of the words into the stopwords table in the database.

Stopwords.txt was initially comprised of the words found at this site http://www.pagex.com/webtools/stopwords.cfm. Additional words were added such as numbers and various symbols.

4.3.2. wordnet.vbs

Wordnet.vbs is a vbscript written to determine what percentage of the terms that appeared in the *SiteMap* database also appeared in Wordnet. It is currently unused. Originally Wordnet was going to be used to evaluate the generated taxonomies. However, due to a large number of proper terms (brand names, product names), Wordnet was not recognizing a significant percentage of the terms and could not be used to evaluate the taxonomies. In the Computer Hardware domain it recognizes 3702 out of 8515 total terms: 43.8%.

4.3.3. results.vbs

Results.vbs is a vbscript written to read the .csv files output by the *SiteGraph* application and generate the conditonal entropy and subsumption results discussed in Section 5.

4.4. Implementation Issues

A variety of issues and difficulties came up during the implementation process and many lessons were learned during the process. For instance, originally, only the relationships created while building the pyramid were inserted into the database using the words and relationship tables. This caused issues when the data such as collocation in pages and links became important and so the process was rewritten to add the full link structure in using the link and pages tables. This allowed a considerable amount of new data to make sense of the tree and relationships between the words. The process should have been written from the ground up to put the lowest level of data (the words, links and urls) into the database and the word relationships should have been built from that. It would have allowed much better flexibility in the relationship building algorithms from the very beginning.

Secondly, an open source crawler should have been used. A custom written crawler was used which had to be updated several times to support more and more tags as they were discovered. Portions of some websites were probably omitted because certain kinds of links are not supported by the currently used crawler. The current crawler only supports the `<a>` tag and the `alt=` attribute for image links. An open source crawler would have been significantly easier, faster, and more complete than the custom written crawler.

Also, a text parser, Badger (http://www-nlp.cs.umass.edu/software/badger.html), was used to extract nouns from the anchor text as that is the only part of speech that was to be included in the hierarchy tree. It was hoped that adjectives, verbs, and other parts of speech would be excluded. Unfortunately, Badger struggled with most anchor text because most anchor text is not complete sentences. Badger does an admirable job of parsing full sentences, however, with phrases, it tends to consider the entire thing as a noun phrase and mark every term as a noun. For instance, this phrase: "Overclocking 9 Value-Priced DDR2-800 Kits" has every word marked as a noun. Future implementations would use a different text parser.

Finally, a C# implementation of the Porter Stemming algorithm was originally used to stem words and combine concepts. However, the stemming tended to be too strict and it was decided that a custom written parser that simply stripped plurals and "ing" endings would appropriately combine terms. The Porter Stemming was combining terms such as “DIGITAL” and “DIGIT”, concepts that we wanted to remain separate. Therefore a much simpler algorithm was implemented that suited the needs of this project more appropriately.

Many issues were encountered and lessons were learned during the implementation process. Most importantly:

1) Always store the lowest level of data before processing
2) Use already written, proven code where available
3) Understand the strengths and limitations of all of the third party tools used.

5. EXPERIMENTAL RESULTS

5.1. Generation

In order to evaluate the validity of the ideas proposed in this thesis, three topic domains were chosen at random: Computer Hardware, Sports, and News. In order to be impartial, the set of websites (*W*) chosen to represent each topic domain are the first five websites returned by Google when searching for each phrase "Computer Hardware", "Sports" and "News" (entered without quotes into Google). The first five Webpages for each domain are:

Computer Hardware:

http://www.pricewatch.com
http://www.geeks.com
http://www.newegg.com
http://www.tomshardware.com
http://www.devhardware.com

Sports:

http://espn.go.com
http://sportsillustrated.cnn.com
http://sports.yahoo.com
http://msn.foxsports.com
http://www.sportsline.com

News:

http://www.cnn.com
http://news.google.com
http://www.foxnews.com
http://news.yahoo.com
http://www.msnbc.com

Each of these URLs was used as a *w.root* along with $\delta = 3$ and $\alpha = 0.6$ (these values were determined by trial and error) to create each website in the set *W*. These websites were the top five returned by Google in the fall of 2006.

5.1.1. Human Judging

The above algorithms were used to generate one taxonomy for each set of websites (each topic phrase) using the following settings. The *topic-cutoff* and *topic-trim-cutoff* are not included because they were determined by comparing input combinations versus output measures in Section **Error! Reference source not found.**. The other settings were used for the taxonomies submitted for human judging and they were determined previously by trial and error.

Computer Hardware:	(crawled 2/4/2007)
Tree Parameters	
root-term:	HARDWARE
max-levels:	4
instance-cutoff:	.01
modifier-cutoff:	.4
top-topic-percent:	10%
Term Set Parameters	
Pages:	.0007
Links:	.0007
Intersection (And relationship)	

Sports:	(crawled 2/4/2007)
Tree Parameters	
root-term:	SPORT
max-levels:	4
instance-cutoff:	.01
modifier-cutoff:	.4
top-topic-percent:	10%
Term Set Parameters	
Pages:	.0008
Links:	.0004
Intersection (And relationship)	

News:	(crawled 1/30/2007)
Tree Parameters	
root-term:	NEW
max-levels:	4
instance-cutoff:	.01
modifier-cutoff:	.4
top-topic-percent:	10%
Term Set Parameters	
Pages:	.001

Links: 1
Union (Or relationship)

The complete taxonomy result sets submitted for human judging are available in the appendix.

5.2. Evaluation

In order to evaluate the taxonomies created from the sites listed above, three methods were chosen. First, the conditional probability of the child and parent term appearing in the same page was used. In our case, since we only used the anchor text of links, this measure is the conditional probability of the child and parent term appearing in the anchor text of two links appearing in the same page. According to this *subsumption measure* [12, 15, 20], if *p* subsumes *c* then the following is true:

$$P(p \mid c) > .8 \wedge P(c \mid p) < P(p \mid c)$$

The second measure, *Generalization/Specialization Quality*, is calculated using the conditonal entropy of the parent and child terms [18]. Given a child term, *c*, and a parent term, *p*, the Generalization/Specialization Quality is calculated as:

$$-P(p, c) * \log(P(p \mid c))$$

This measure is designed to “quantif[y] how a hierarchy respects the relation of generalization/specialization between the objects which are in the nodes.” [18] A lower value indicates a better taxonomy.

The third measure used human judges to determine the validity of the taxonomy. Six judges were asked to rate each relationship (parent-term, child-term). This measures the precision of the created taxonomy. Each of the judges was at least moderately familiar with each of the three topic domains, NEWS, SPORTS, and COMPUTER

HARDWARE. The judges were knowledgeable enough to understand the basic concepts in each domain and how they are related to each other.

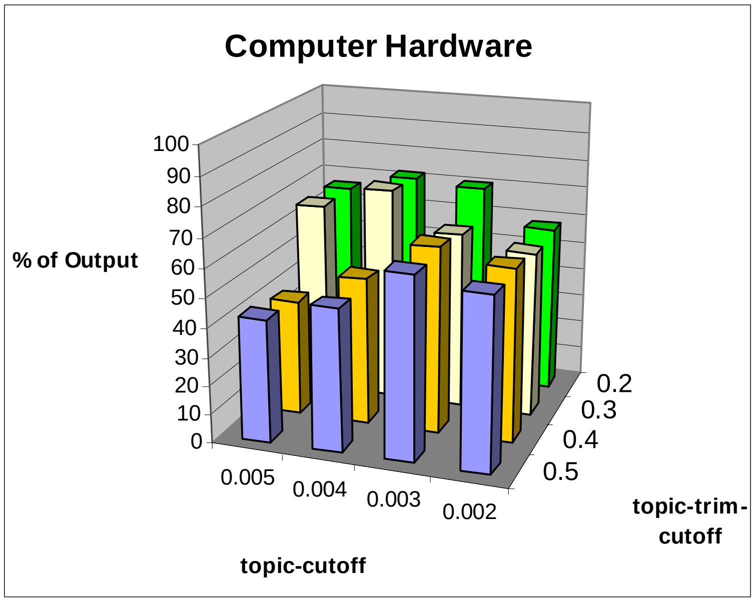
5.2.1. Effect of Cutoff Parameters

Based on preliminary trial and error experiments, we found that the two input parameters that most affect the content of the tree were the *topic-trim-cutoff* and the *topic-cutoff*. The *topic-cutoff* affects the number of topic nodes allowed into the tree and the *topic-trim-cutoff* affects how liberal the algorithm is when trimming Topic nodes. Both of these parameters have a very large effect on the length and breadth of the tree. Sixteen combinations of both input parameters for each domain were tried versus the percent of relationships in the output taxonomies that fulfilled both requirements of the subsumption measure. First, P(*parent*|*child*) must greater than 60% (see rationale for choosing 60% below). Second, P(*parent*|*child*) must be greater than P(*child*|*parent*). The chosen input values for topic-cutoff were: .002, .003, .004, and .005. The chosen input values for topic-trim-cutoff were: .2, .3, .4, and .5. The effect of the parameters on the average Specialization/Generalization quality described in [18] was also studied.

Due to the fact that the subsumption measure is designed for traditional taxonomy building techniques [12, 15, 20], instead of using the recommended 80% guideline we will be using 60%. Traditional techniques use either syntactic pattern matching or statistical collocation both of which require related terms to be near each other in text. Therefore, such measures are good for evaluating taxonomies already based on distance or pattern matching. Because the techniques developed in this thesis rely primarily on link structure, related terms don't necessarily have to appear in the same document.

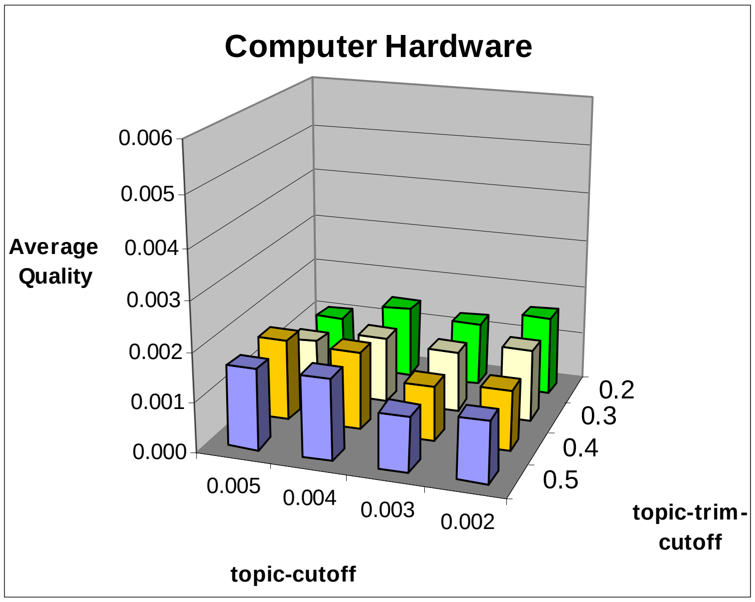
Also, the subsumption measure was designed using complete text documents which would contain significantly more text then the anchor text used by these algorithms.

Computer Hardware



	topic-cutoff				
topic-trim-cutoff		**0.005**	**0.004**	**0.003**	**0.002**
	0.5	42.42	49.02	63.25	59.35
	0.4	40.00	50.94	64.57	60.00
	0.3	66.67	**74.42**	61.18	57.14
	0.2	66.67	72.41	71.01	58.46

Figure 5-1 Percent of Output that fulfills Subsumption Measure vs. Input parameters for Computer Hardware domain

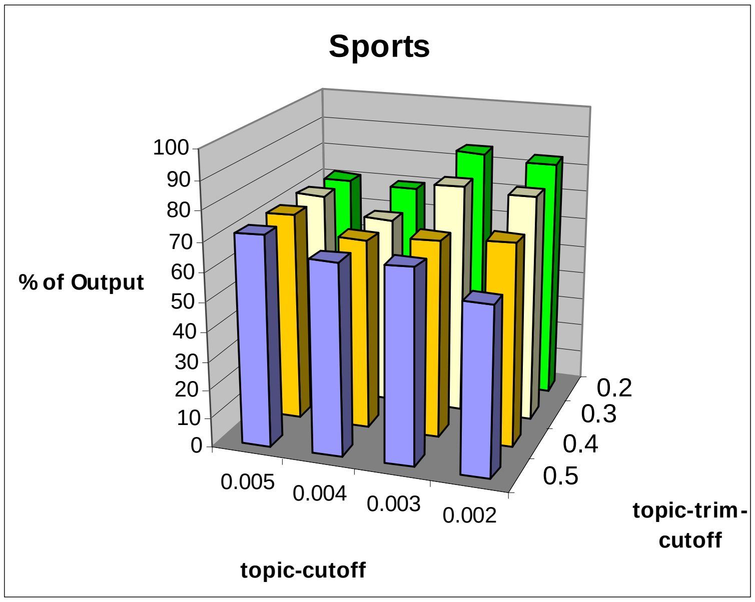


	topic-cutoff				
		0.005	**0.004**	**0.003**	**0.002**
topic-trim-cutoff	**0.5**	0.0016	0.0016	**0.0011**	0.0012
	0.4	0.0017	0.0016	**0.0011**	0.0012
	0.3	**0.0011**	0.0013	0.0012	0.0015
	0.2	**0.0011**	0.0015	0.0013	0.0016

Figure 5-2 Average Generalization/Specialization quality vs. Input parameters for Computer Hardware domain

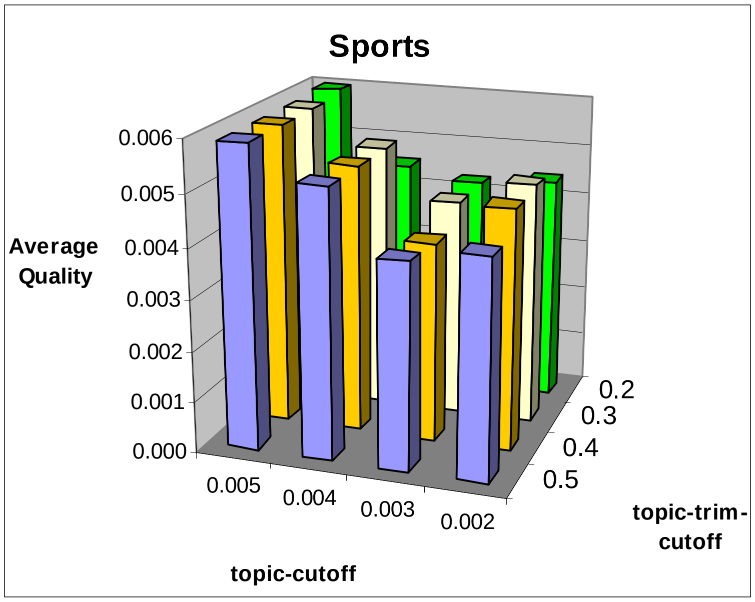
According to the subsumption measure, the Computer Hardware domain was best when the *topic-cutoff* is 0.004. Also, except for a couple exceptions, it appears as if the more restrictive values for the *topic-cutoff* and *topic-trim-cutoff* generate better results. Also, the average Generalization/Specialization Quality agrees with the Subsumption measure. According to both measures the worst taxonomies are created when the *topic-cutoff* is 0.005 and the *topic-trim-cutoff* is 0.5 or 0.4.

Sports



	topic-cutoff				
		0.005	**0.004**	**0.003**	**0.002**
topic-trim-cutoff	**0.5**	71.88	65.71	66.67	57.61
	0.4	71.88	65.71	67.86	69.57
	0.3	71.88	65.71	79.55	78.13
	0.2	71.43	70.37	**85.00**	83.33

Figure 5-3 Percent of Output that fulfills Subsumption Measure vs. Input parameters for Sports domain

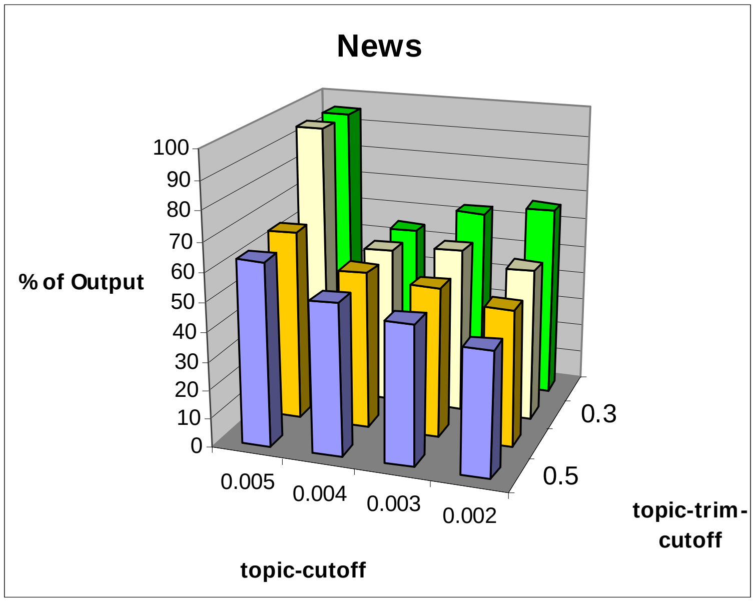


		topic-cutoff			
		0.005	**0.004**	**0.003**	**0.002**
topic-trim-cutoff	**0.5**	0.0059	0.0052	0.0040	0.0043
	0.4	0.0059	0.0052	**0.0039**	0.0047
	0.3	0.0059	0.0052	0.0043	0.0048
	0.2	0.0066	0.0045	0.0043	0.0044

Figure 5-4 Average Generalization/Specialization quality vs. Input parameters for Sports domain

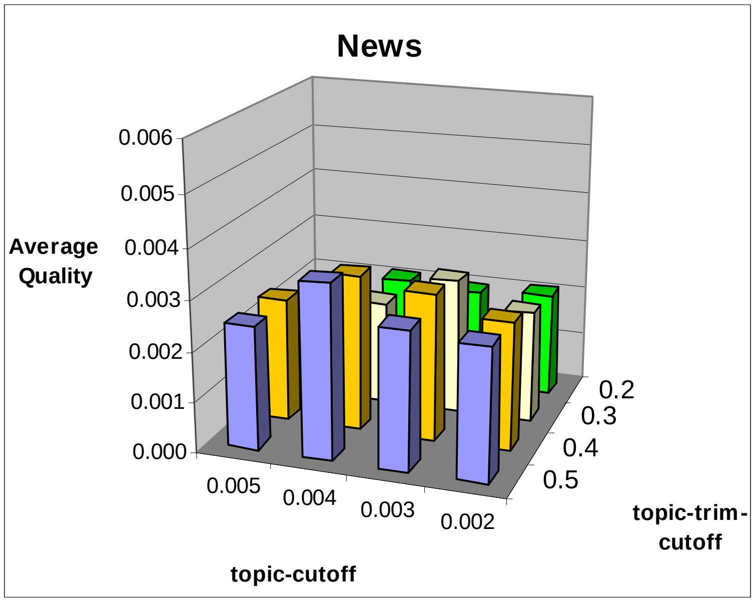
The Sports topic domain also shows that more restrictive values of the *topic-cutoff* and *topic-trim-cutoff* tend to create better results as measured by the Subsumption measure. Overall the results are the best of all three topic domains and reach as high as 85% of all output relationships according to the Subsumption measure. Also, the Subsumption measure and the Generalization/Specialization quality again agree on the best and worst taxonomies in the Sports domain.

News



	topic-cutoff				
		0.005	**0.004**	**0.003**	**0.002**
topic-trim-cutoff	**0.5**	63.16	52.29	48.39	42.76
	0.4	65.79	54.46	52.03	47.24
	0.3	**95.24**	54.70	57.14	52.71
	0.2	**95.24**	54.70	63.41	66.67

Figure 5-5 Percent of Output that fulfills Subsumption Measure vs. Input parameters for News domain



	topic-cutoff				
		0.005	**0.004**	**0.003**	**0.002**
topic-trim-cutoff	**0.5**	0.0025	0.0035	0.0028	0.0026
	0.4	0.0025	0.0031	0.0029	0.0025

	0.3	**0.0008**	0.0021	0.0027	0.0022
	0.2	**0.0008**	0.0021	0.0020	0.0021

Figure 5-6 Average Generalization/Specialization quality vs. Input parameters for News domain

The News topic domain also shows that more restrictive values of the *topic-cutoff* and *topic-trim-cutoff* tend to create better results as measured by the subsumption measure. The two output taxonomies that had the highest ratings (95%) both created less than twenty relationships in the final taxonomy. For more practically sized taxonomies, the scores in the News topic domain were less than the previous domains. The most impressive score being about 65% of output relationships fulfilling the subsumption measure. Also, the two measures again agree on which taxonomies are the best.

Over all three topic domains there appears to be a correlation between *topic-cutoff* and *topic-trim-cutoff*. Making these values more restrictive tends to create results that score better using the both the Subsumption Measure and the Generalization/Specialization quality.

5.2.2. Human Judging

Six taxonomies were generated and submitted for human judging. Each taxonomy was generated using the settings in Section 5.1.1 and a *topic-cutoff* and *topic-trim-cutoff* that generated the best taxonomies in Section **Error! Reference source not found.**. For the SPORTS and COMPUTER HARDWARE domain, two different taxonomies were indicated best by the Subsumption measure and the Generalization/Specialization quality. Therefore, both taxonomies were submitted for human judging. In the NEWS domain the same taxonomy scored the best for both measures. The second best taxonomy from the NEWS domain was also added. The chosen taxonomies for human evaluation are listed in Table 5-1.

Topic Domain	topic-cutoff	topic-trim-cutoff	# of relationships	Subsumption Measure	Generalization/ Specialization Measure
Computer Hardware	0.004	0.3	43	74.42%	0.0013
Computer Hardware	0.005	0.2	36	66.67%	0.0011
Sports	0.003	0.2	40	85.00%	0.0043
Sports	0.003	0.4	56	67.86%	0.0039
News	0.005	0.2	21	95.24%	0.0008
News	0.003	0.2	123	63.41%	0.0020

Table 5-1 Output Measures for the taxonomies submitted for human judging

5.2.3. Human Judges

Each of six human judges was presented with the following text:

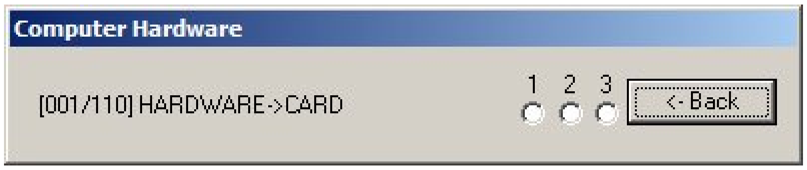
```
       A taxonomy is a classification in a hierarchical system.  A hierarchy is a system
of ranking and organizing things, where each element of the system (except for the top
element) is subordinate to a single other element.  The concept of the subordinate, child
element should be more specific than the parent element.

For instance:
    In the domain of Musical Instruments
    Parent -> Child
    DRUM   -> BONGO
    would make sense because a bongo is a type of drum.

       You will be presented with pairs of elements (a parent and child) from a taxonomy
of terms generated from three topic domains:  SPORTS, COMPUTER HARDWARE, and NEWS.  For
each pair rank them on a scale from 1 (is not meaningful) to 3 (is meaningful) keeping in
mind the domain the term pair is coming from.

       1: NOT MEANINGFUL
       2: SOMEWHAT MEANINGFUL
       3: MEANINGFUL
```

Then each judge used a program, *SiteJudge*, to evaluate the pairs of terms in the result set. The judge was presented with this screen:


Computer Hardware

[001/110] HARDWARE->CARD

Figure 5-7 Screenshot of *SiteJudge*

The parent and child relationship was displayed along with the domain and a set of radio buttons to select the rating (1 through 3). Selecting a rating would make the next relationship immediately appear. If the judge felt they had misselected or wanted to review their choices they could use the back button. To see the full set of results refer to the Appendix.

The average ratings and standard deviation of all relationships from all judges for each submitted taxonomy was:

Topic Domain	topic-cutoff	topic-trim-cutoff	Average Rating	Percent > 2	Standard Deviation
Computer Hardware	0.004	0.3	2.43	79.07%	0.45
Computer Hardware	0.005	0.2	2.30	66.67%	0.44
Sports	0.003	0.2	2.43	90.00%	0.43
Sports	0.003	0.4	2.35	82.14%	0.49
News	0.005	0.2	2.30	80.95%	0.49
News	0.003	0.2	2.24	57.72%	0.45

Table 5-2 Average and standard deviation of relationship ratings by topic and totals

As one can see the results are positive. The Sports domain did the best with 90% of relationships having an average rating across all raters greater than 2, while the News domain was overall the worst. This may be due to the fact that News is, perhaps, the broadest of all three topics. In the case of the Sports and Computer Hardware domains, the taxonomy that scored better using the Subsumption measure also was judged better by humans. Also the standard deviations are all relatively low meaning that, in general, the judges agreed on the meaning of the terms and rated them very closely.

6. CONCLUSION AND FURTHER WORK

This thesis describes a method to extract information from the link structure of a website and proves that this is a viable method for automatic taxonomy generation. Two programs, *SiteMap* and *SiteGraph*, were developed in order to demonstrate the information available in the link structure of the website.

The taxonomies created by *SiteMap* and *SiteGraph* were evaluated using three different techniques. The Subsumption measure, the Generalization/Specialization Quality and particularly the human judging validate the use of the link structure of Websites to create term taxonomies. The methods used in this thesis do not necessarily create perfect taxonomies, but they definitely prove that the link structure contains unused information that, combined with other techniques, can create more and more accurate term taxonomies with less and less human interaction. This thesis contributes new ideas about using the link structure and text from the World Wide Web, and has an impact on the fields of Information Retrieval and Data Mining. In the future, more information will become available on the web, hyperlinked with existing documents, which means that the use of the links between documents will become a more important clue into data categorization.

Also in the future, the ideas in this thesis could be used in combination with other automatic taxonomy generation algorithms. The link structure could be used in concert with clustering or syntactic analysis in order to create more accurate taxonomies. Also, the link structure could be used more directly by clustering algorithms by considering the linking terms in collocation measures.

There are still many barriers to automatically creating taxonomies that the ideas in this thesis did not overcome. Accurately parsing natural languages is difficult, but as these techniques improve, taxonomy generation will also improve greatly and vice versa. There are three main shortcomings of natural language processing that impacted the results of this thesis.

First, this thesis attempted to only categorize nouns by using a natural language processor, but it became apparent that a large number of other parts of speech ended up in the taxonomies. Part of speech processing is important in all kinds of taxonomy generation and gives important clues to how words should be placed in a hierarchy.

Second, the accurate identification of n-grams in the document set would have increased the strength of the algorithm. If you pull apart the n-gram “SOUND CARD”, you lose the meaning of the terms in the document. Processing the term “SOUND” separately from “CARD” is completely different from the combined concept.

Finally, more accurately identifying and combining the meaning of synonymous terms and phrases would have strengthened the concepts in the final taxonomies. Some concepts were referred to using many different term sets such as: “COLLEGE BBALL”, “BASKETBALL”, “NCAA BASKETBALL” or from another domain: “MOTHERBOARD”, “MOBO” or “MB”. If these concepts were identified as being similar while parsing the documents, and their meanings combined, the final taxonomies would have been significantly stronger.

As computers become more “intelligent” and require the ability to classify larger amounts of information, taxonomy generation becomes more important. The ability to process new data and place it appropriately into a hierarchy of known data is important for both humans and computers. Early research, such as the work in this thesis, on taxonomy generation and data classification can be considered an important step in the field of Artificial Intelligence and Semantic Web Mining.

REFERENCES

[1] G. Bisson, C. Nedellec, and L. Canamero. “Designing clustering methods for ontology building - The Mo'K workbench”. In Proceedings of the ECAI Ontology Learning Workshop, pages 13--19, 2000.

[2] D. Boley. “Hierarchical taxonomies using divisive partitioning”. Technical Report TR-98-012, University of Minnesota, 1998.

[3] P. Cimiano , S. Staab, “Learning by Googling”, ACM SIGKDD Explorations Newsletter, v.6 n.2, p.24-33, December 2004

[4] A. Faatz, R. Steinmetz, “Ontology enrichment with texts from the WWW”, in Proceedings of Semantic Web Mining second Workshop at ECML/PKDD-2002, Finland, 2002.

[5] B. Fortuna, D. Mladenic, M. Grobelnik, (2005a). “Semi-automatic construction of topic ontology”. Proceedings of the ECML/PKDD Workshop on Knowledge Discovery for Ontologies

[6] P. Ganesan, H. Garcia-Molina, J. Widom. Exploiting hierarchical domain structure to compute similarity. ACM Transactions on Information Systems, 21(1):64–93, January 2003.

[7] X. Jiang, A. Tan, “Mining Ontological Knowledge from Domain-Specific Text Documents”, Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM ’05), 2005

[8] A. Maedche, V. Pekar, and S. Staab. “Ontology learning part one - on discovering taxonomic relations from the web”. In Web Intelligence, pages 301--322. Springer Verlag, 2002.

[9] A. Maedche, S. Staab, “Ontology Learning for the Semantic Web”, IEEE Intelligent Systems, v.16 n.2, p.72-79, March 2001

[10] B. Poblete, R. Baeza-Yates, “A Content and Structure Website Mining Model”. Proceedings of the 15th international conference on World Wide Web, pages 957-958, Edinburgh, Scotland, 2006.

[11] P. Velardi, A. Cucchiarelli, M. Petit, “A Taxonomy Learning Method and its Application to Characterize a Scientific Web Community”. IEEE Transactions on Knowledge and Data Engineering. Vol. 19, No. 2, February 2007

[12] R. Krishnapuram, “Introduction to Knowledge Management and Text Mining”. Invited Tutorial in FUZZIEEE 2003, St. Louis

[13] http://www.w3.org/2001/sw/

[14] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. “Proceedings of the 14th International Conference on Computational Linquistics”, pages 539-545, 1992.

[15] D. Lawrie and W.B. Croft. Discovering and comparing topic hierarchies. In “Proceedings of RIAO 2000”

[16] http://sourceforge.net/projects/netron-reloaded

[17] http://www.dotnetcharting.com/

[18] H. N. Fotzo, P. Gallinari, "Learning « Generalization/Specialization » Relations between Concepts – Application for Automatically Building Thematic Document Hierarchies", IN RIAO, 2004.

[19] O. Nasraoui, H. Frigui, A. Joshi, and R. Krishnapuram, "Mining Web Access Logs Using Relational Competitive Fuzzy Clustering", In Proc. Eight International Fuzzy Systems Association World Congress - IFSA 99, Taipei, August 99. http://citeseer.ist.psu.edu/140325.html

[20] Sanderson and Croft. “Deriving concept hierarchies from text.” SIGIR '99, Berkley, CA USA

[21] http://www-nlp.cs.umass.edu/software/badger.html

APPENDIX A (CODE)

This appendix contains code detailing the most important parts of the *SiteGraph* and *SiteMap* applications. The code is not reproduced here in full due to its length.

This first code section describes the way the Webpages are organized into a tree hierarchy using the depth rating.

```
Assume:
+Q is a Queue
    Enqueue(): Adds an object to the tail of the queue.
    Dequeue(): Returns and removes the object at the head of the queue.
    length   : Returns the number of objects in the queue.
+T is a Tree
    root    : The root treenode
+w is a Website as defined in the Notation section
+A TreeNode has the following properties and functions
    page    : A Webpage as defined in the Notation section
    nodes   : The set of child nodes of this node.
    AddNode(): Takes a TreeNode and adds it to the nodes set.
    depth   : the distance from the root node.
+FindPageInTree(p) is a function that searches the tree for the Webpage p in the each
    TreeNode's Page property.  It returns a TreeNode if it finds it, otherwise null.
+MoveNodeInTree(tn, tn') is a function that moves the node tn (and all descendants)
    from its current location to be a child of tn'.
+DepthStdDev is the standard deviation of all depth ratings of Webpages in W.pages
+tChild is a TreeNode
+pChild is a Webpage
+tNew is a newly created TreeNode

fBuildTree()

T.root.page = w.root
Q.Enqueue(T.root)

//while we have nodes left to process
while Q.length > 0
    TreeNode tNode = Q.Dequeue()

    //check each webpage this node points to
    for each Webpage pChild in tNode.page.pagesto
       //check to see if the node already exists
       if (tChild = FindPageInTree(pChild)) is null
            //it hasn't been added, so by default its our child
            tNew.page = pChild
            tNode.AddNode(tNew)
            Q.Enqueue(tNew)
       else
            //it already exists in the tree, check to see if we should move it
            if tNode.page.depthrating + DepthStdDev < tChild.page.depthrating
                    if tNode.depth > tChild.depth
                            MoveNodeInTree(tChild, tNode)
                    end if
            end if
       end if
    end for
end while
```

The above algorithm adds each TreeNode with its associated Webpage into a queue, starting with the TreeNode encapsulating `w.root`, from which they are removed and processed one at a time. As it processes the TreeNode it checks each member of the TreeNode's Page's `pagesto` set. If the Webpage from the `pagesto` set has not yet been added to the tree then a new TreeNode is created and added as a child of `tNode` with the Webpage as its `page` property. If the Webpage has already been added to the tree, it evaluates two conditional expressions to determine if it should move the Webpage to be a child of the current TreeNode.

$t.Node.page.depthrating + DepthStdDev < tChild.page.depthrating$
$t.Node.depth > tChild.depth$

The first expression checks to see if the depth rating of the Webpage in question is greater than the potential parent Webpage's depth rating plus one standard deviation. The standard deviation is used to make sure that the Webpage's depth ratings are significantly different.

The second expression makes sure that we are always moving the TreeNode "down" the tree to a lower level. This check makes sure that a TreeNode that exists at a lower, "more specific" level isn't moved up or sideways.

This next code section describes the way the hierarchy of Webpages is used to create the Relationship (a set of tuples described in Section 3.2).

```
Assume:
+The TreeNode objects are the same as those in the above section (Organizing the
Pages)
+L is a list
    Contains(): Returns true if the input string exists in L
    Add(): Adds the input string to L
+T is the same tree as in the above section (Organizing the Pages)
+This recursive function is called on fCalcRel(T.root, {})
+AddRel() is a function defined externally which takes a parent term, child term, and
a value indicating the strength of the relationship.  These values are stored for
later use.

fCalculateRelationships(TreeNode tn, List L)
```

```
//for every combination of parent
foreach parent in tn.page.terms
    foreach pChild in tn.nodes.page
        //and child term
        foreach child in pChild.terms
            //if we've not already added it
            if not L.Contains(child)
              //add the tuple to the relationship
              fAddRel(parent,child,count(pChild,child) / Σt ∈ pChild.terms count(pChild, t))
            end if
        end for
    end for

   L.add(parent-term.term)
end for

foreach pChild in tn.nodes.page
    //and recurse the tree
   fCalculateRelationships(pChild, L)
end for
```

This recursive algorithm descends the tree created out of the Webpages in the website and adds relationships for all of the anchor text used to refer to the current webpage as the "parent" terms (`tn.page.terms`) and all of the anchor text used to refer to all of its child Webpages as the "child" terms (`tn.nodes.page.terms`). The `fAddRel()` function is a wrapper for the `add()` function described in Section 3.2 above. If a tuple, r, already exists such that `r.parent-term = parent.term` and `r.child-term = child.term` then

$$\texttt{r.count +=} \frac{\texttt{count}(pChild, child)}{\sum_{t \in pChild.terms} \texttt{count}(pChild, t)}$$

Otherwise, a new tuple is created and added to the relationship R with

$$\frac{\texttt{count}(pChild, child)}{\sum_{t \in pChild.terms} \texttt{count}(pChild, t)}$$

as `r.count`.

The list `L` is used to keep the algorithm from adding children to a term if any ancestor node has already handled that term. Early on during development it was noticed that very often, more generic terms used farther up in the tree would have their child relationships severely altered by more specific uses farther down inside of the tree.

Therefore, a list of already used words was created to keep descendants from contributing to an already defined term.

It should be noted that, as the terms are added, they are normalized so that the contribution of each child webpage was weighed equally:

$$\mathtt{fAddRel}\left(parent, child, \frac{\mathrm{count}(pChild, child)}{\sum_{t \in pChild.terms} \mathrm{count}(pChild, t)}\right)$$

This next code section shows the driving loop that *SiteGraph* uses to create the final taxonomies using the Relationship.

```
Assume:
+T is a Tree
    root    : The root treenode
+A TreeNode has the following properties and functions
    term    : a character string
    nodes   : The set of child nodes of this node.
    AddNode(): Takes a TreeNode and adds it to the nodes set.
    depth   : the distance from the root node.
    instance: a boolean value indicating if this is an instance term
+root-term and max-levels are input parameters as defined above.

fBuildTaxonomy()

T.root.term   = root-term
CurrentLevel  = 0
while(CurrentLevel < max-levels)
    BuildLevel(CurrentLevel, T.root)
    TrimLevel(CurrentLevel, T.root)
    CurrentLevel++
end while
```

This code section shows how the taxonomy is trimmed while it is being created.

```
Assume:
+L is a list of tree nodes
    Add() : adds the input TreeNode to the list
+AddInstanceChildren(): adds instance terms as described above using the input
    TreeNode.term as the parent term.
+RemoveChildren(): Removes all child nodes from the TreeNode.nodes set
+CountUniqueChildrenInWholeTree(): Counts the number of children of the input TreeNode
    whose .term appears nowhere else in the tree.
+CountUniqueChildrenInHigherLevels(): Counts the number of children of the input
    TreeNode whose .term appear in no nodes whose depth is <= the input
    TreeNode.depth.
+RemoveChildrenInHigherLevels(): Removes all child nodes of the input TreeNode whose
    .term appears whose depth is <= the input TreeNode.depth.
+UniqueChildren and TotalChildren are integer values
+topic-trim-cutoff is the cutoff parameter set by the user described above.
+CurrentLevel is input this trimming function

TrimLevel(CurrentLevel, T)

//get all nodes at this level
for each TreeNode tn in T where tn.depth = CurrentLevel
    nUniqueChildren = CountUniqueChildreninWholeTree(tn)
    //and remove children if we either contribute no new children or our
    //new children/all children ratio is less than topic-trim-cutoff
    if nUniqueChildren = 0
            L.Add(tn)
    else
            nUniqueChildren = CountUniqueChildreninHigherLevels(tn)
            nTotalChildren  = |tn.nodes|

            if nUniqueChildren / nTotalChildren > topic-trim-cutoff or
                            nUniqueChildren = 1                    then
                    //add to the remove list
                    L.Add(TreeNode)
            end if
    end if
end for

//Remove all children from trimmed topic nodes and add instance children
for each TreeNode tn in L
    RemoveChildren(tn)
    AddInstanceChildren(tn)
end for

//now remove children that have already appeared in the tree at a higher lvl
for each TreeNode tn at CurrentLevel
    RemoveChildrenInHigherLevels(tn)
end for
```

This code section shows how crosstopic terms are discovered which is part of the trimming process.

```
Assume:
+TermCount is an object with the following properties and methods
    count: The number of times a term has appeared
    treenode: The treenode containing the Term
+L is a list of TermCount objects
Contains(): takes a string and returns true if one of the TermCount.treenode.terms
    matches it.
Add(): takes a TreeNode and integer and adds a new TermCount object
[]: takes a string and returns the TermCount object whose treenode.term matches it.
+CalcAvgAppearance() takes a list of term count objects and returns the average of the
    count property
+CalcStdDevAppearance() takes a list of term count objects and returns the standard
    deviation of the count property
+MarkAsCrossTopic() marks the node as a crosstopic node so it's displayed
    appropriately to the user.
    +Avg and StdDev are real values

for each TreeNode tn in T where tn.Depth = CurrentLevel + 1
    if L.Contains(TreeNode.Term)
            listNodeCounts[TreeNode.Term].count++
    else
            listNodeCounts.Add(TreeNode, 1)
    end if
end for

fFindCrossTopic()

Avg    = CalcAvgAppearance(L)
StdDev = CalcStdDevAppearance(L)

for each TermCount tc in L
    //if this term appears more times than the average plus two times the
    // std dev, than this is an outlier and a modifier term.
    if tc.Count > Avg + StdDev * 2 then
            MarkAsCrossTopic(tc.treenode)
    end if
end for
```

SiteMap.sql

Due to length, only the tables, functions, and procedure stubs are included.

```
--use sitemap
--create database sitemap

use sitemap
go

create table words
(
       id int                 NOT NULL,
       word varchar(100)      NOT NULL
)
go

create table relationship
(
       parent_id      int     NOT NULL,
       child_id       int     NOT NULL,
       count          int     NOT NULL
)
go

create table pages
(
       id int                 NOT NULL,
       url varchar(500)       NOT NULL
)
go

create table link
(
       id             int     NOT NULL,
       to_id          int     NOT NULL,
       from_id        int     NOT NULL,
       word_id        int     NOT NULL
)
go

create table stopwords
(
       id      int     NOT NULL
)
go

CREATE FUNCTION fGetNextID()
RETURNS int

create procedure pInsertWord
       @p_Word varchar(100)

create function fGetID
       (@p_Word varchar(100))
RETURNS int

create procedure pInsertRelationship
       @p_WordParent varchar(100),
       @p_WordChild  varchar(100),
       @p_Count      int

CREATE function fGetParentCount
       (@p_id int)
RETURNS int

CREATE FUNCTION fGetNextPageID()
RETURNS int

CREATE FUNCTION fGetNextLinkID()
RETURNS int
```

```
create function fGetPageID
       (@p_URL varchar(500))
RETURNS int

create procedure pInsertPage
       @p_URL    varchar(500)

create function fGetLinkID
       (@p_FromURL varchar(500), @p_ToURL varchar(500))
RETURNS int

create procedure pInsertLinkTerm
       @p_FromURL varchar(500),
       @p_ToURL   varchar(500),
       @p_Term    varchar(100)

create function fGetAllowedTerms
       (@p_LinkThreshold real,
        @p_PageThreshold real,
        @p_Or           bit)
RETURNS @words table
       (
              id             int,
              word           varchar(100),
              in_links       int,
              total_links    int,
              in_pages       int,
              total_pages    int
       )

create function fGetAllowedTermsFromLinks
       (@p_LinkThreshold real)
RETURNS @words table
       (
              id             int,
              word           varchar(100),
              inlinks        int,
              totallinks     int,
              percentlinks   real
       )

create function fGetAllowedTermsFromPages
       (@p_PageThreshold real)
RETURNS @words table
       (
              id             int,
              word           varchar(100),
              inpages        int,
              totalpages     int,
              percentpages   real
       )

create function fGetChildren
       (@p_word              varchar(100),
        @p_LinkThreshold     real,
        @p_PageThreshold     real,
        @p_Threshold         real,
        @p_Or                bit)
returns @childwords table
       (
              child_id       int,
              child_word     varchar(100),
              child_count    int,
              parent_count   int,
              perc_appear    real,
              in_links       int,
              total_links    int,
              in_pages       int,
              total_pages    int
       )
```

```
create function fGetCollocationByPage
       (@p_word1             varchar(100),
        @p_word2             varchar(100))
returns @collocation table
       (
              word1total int,
              word2total int,
              collocation int
       )

create function fGetCollocationByLink
       (@p_word1             varchar(100),
        @p_word2             varchar(100))
returns @collocation table
       (
              word1total int,
              word2total int,
              collocation int
       )
```

APPENDIX B (XML FORMATS)

This appendix contains the format of two of the XML files mentioned in this thesis. The first, *SiteMap* XML, is created when *SiteMap* maps Webpages and saves the pages. The second, Term Cluster XML, is created when the user selects Export from the Cluster Dialog in the *SiteGraph* application.

SiteMap XML

```
<root>
        <page>
                <url><![CDATA[http://www.foxsports.com:80]]></url>
                <id>0</id>
                <linksto>
                        <link>
                                <pointsto>1</pointsto>
                                <count>1</count>
                        </link>
                        ... repeat for all links to
                </linksto>
                <linksfrom>
                        <link>
                                <pointsto>28</pointsto>
                                <count>2</count>
                                <terms>
                                        <term><![CDATA[BEN]]></term>
                                        <term><![CDATA[MALLER]]></term>
                                        <term><![CDATA[RUMORS]]></term>
                                        <term><![CDATA[NOTES]]></term>
                                </terms>
                        </link>
                        ... repeat for all links from
                </linksfrom>
                <Terms>
                        <TotalCount>4</TotalCount>
                        <Term>
                                <Text><![CDATA[BEN]]></Text>
                                <Count>1</Count>
                        </Term>
                        ... repeat for all terms
                </Terms>
        </page>
        ... repeat for all pages
</root>
```

Term Cluster XML

```
<split>
        <split>
                <split>
                        <word><![CDATA[MIDI]]></word>
                        <word><![CDATA[AUDIO]]></word>
                </split>
                <split>
                        <word><![CDATA[LAB]]></word>
                        <word><![CDATA[CREATIVE]]></word>
                </split>
        </split>
        <word><![CDATA[SORT]]></word>
```

```
</split>
```

APPENDIX C (RESULTS)

Human Judged Taxonomies

Computer Hardware

topic-cutoff: 0.004
topic-trim-cutoff: 0.3

HARDWARE	CARD:USB	VIDEO	GEFORCE
			TV
			SONY
			CAMERA
			HD
			LITE-ON
			ATI:RADEON
		SOUND	
		PCI	
	PC:USB	REVIEW	
		SYSTEM	SERVER
			WINDOW
			CISCO
			XP
			PRO
			LAPTOP
			APPLE
			RAM
	MOTHERBOARD	INTEL	
		SOCKET	
	MEMORY	BOX	
		512MB	
		DDR	
		OCZ	
		1GB	
		256MB	
		RAT	
		FLASH:USB	
		2GB	
		CPU	
		FAN	
		128MB	
	NETWORK:USB	WIRELESS	NETGEAR
			FIREWALL
		MICROSOFT	

topic-cutoff: 0.005
topic-trim-cutoff: 0.3

HARDWARE	CARD:USB	VIDEO:RAT:PCI	GEFORCE
			TV

			SONY
			CAMERA
			HD
			LITE-ON
			ATI:RADEON
			128MB
		SOUND:RAT:PCI	CREATIVE:LAB
			AUDIO
			SORT
			SIIG
			MIDI
			GOLD
	PC:USB	REVIEW	
		ATHLON	
		PENTIUM	
	MOTHERBOARD	INTEL	
		SOCKET	
	MEMORY	BOX	
		FLASH	DRIVE
			2GB
			DISK
			1GB
			RAT
			PORTABLE
			KINGSTON
			512MB
			256MB

Sports

topic-cutoff: 0.003
topic-trim-cutoff: 0.2

SPORT	NFL:PHOTO	REGISTER	
		STATE	
		OKLAHOMA	
		SPORTSNATION	
		QUICK:HIT	
	NHL:TEAM	RANK	
		ALL-STAR	
		ROSTER	
		STAR	
	MLB:PHOTO:TEAM	DEAL	
		BOND	
		RED	
		ODD	
		SIGN	
		HALL	
		STAT	
		TRANSACTION	

		INJURY	
		PLAYER	
		STAND	
	NBA:PHOTO:TEAM	DRAFT	
		HISTORY	
		TORONTO	
	NCAA:TEAM	STAT	
		STAND	FLORIDA
			MICHIGAN
			TEXA
		PLAYER	
		WOMEN	
	SCHEDULE	RECAP	
		BOX:SCORE	
		VIDEO	
		HUNT	

topic-cutoff: 0.003
topic-trim-cutoff: 0.4

SPORT	NFL:PLAYER	REGISTER	
	NHL:TEAM	RANK	
		ALL-STAR	
		ROSTER	
		STAR	
	MLB:PLAYER:TEAM	DEAL	
		BOND	
		RED	
		ODD	
		SIGN	
		HALL	
		STAT	
		TRANSACTION	
		INJURY	
		STAND	
		PHOTO	
	NBA:PLAYER:TEAM	DRAFT	OKLAHOMA
			STATE
			SPORTSNATION
			HISTORY
	NCAA:PLAYER:TEAM	STAT	STATISTIC
			LEADER
			FUTURE
		STAND	STATE
			FLORIDA
			MICHIGAN
			TEXA
		WOMEN	WNBA
			STATE
			FLORIDA

			TV
			SCOREBOARD
	SCHEDULE	RECAP	JONE
			FULL
			SHOT
			CHART
		BOX:SCORE	SIMMON
			WIRE
			NIGHT
			CHAMPION
			TV:ESPN
			TITLE
			PETER
			SCORECARD

News

topic-cutoff: 0.005
topic-trim-cutoff: 0.2

NEW	WORLD	IRAN:U.S.	HOUSE:WHITE:FULL:COVERAGE	SHIITE:IRAQI:ARMY:LEADER:CULT
				ABC
				MISUSED:ISRAEL:BOMB
				RESPOND:CHENEY
				HAGEL:CRITICISM
				DENOUNCE
				ATTACK
				MILLION
				MARK
				PASTRY:CHEF
				PLEAD:TERROR:SUSPECT
				GUILTY
				HIRE:GIULIANI
				N.H:GOP:CHAIR
			IRAQ:FULL	MILITARY
				SURGE:TROOP
				BUSH
				OPTION
				DEATH
				TROUBLE
				CONFLICT
				IMAGE
	HEALTH	BIRD:FLU		
		VIDEO		
		LIBRARY		
		HEART		
		DRUG		

		MOM:OLDEST		
		CLINIC		
		LABOR		
		BIG		
		CHINA		
		MENTAL		

topic-cutoff: 0.003
topic-trim-cutoff: 0.2

NEW	WORLD	ROAD		
		RETIREMENT		
		IRAN		
		FORECAST		
		CANADIAN		
		WINDOW:VISTA		
		TEMPERATURE		
		PERSON:OLDEST		
	HEALTH:VIDEO	BIRD:FLU		
		LIBRARY		
		HEART		
		DRUG		
		MOM:OLDEST		
		CLINIC		
		LABOR		
		BIG		
		CHINA		
		MENTAL		
	ENTERTAINMENT:VIDEO	TV	QUESTION	AP
				CHENEY
				COLT
				CASE
				LIVE
				VOTE
				PART
				STAND
				TRIAL
			SHOW	PHOTO
				JOHN
				TROUBLE
				MISSION
				DOWNLOAD
				ALAN
				EXCLUSIVE

				BIO
				PICK
				WALL
				FAMOU
				FACE
				OFFICIAL
				RULE
				TURN
			WAR	FULL
				COVERAGE
				CONGRESS
				BILL
				LEVEL
				MISUSED:ISRAEL:BOMB
				RESPOND:CHENEY
				HAGEL:CRITICISM
				LEAVE
				PASTRY:CHEF
				DENOUNCE:ATTACK
				PLEAD:TERROR:SUSPECT
				GUILTY
				HIRE:GIULIANI
				N.H:GOP:CHAIR
			REPORT	EARTH
				PROBE
				S.D:LAWMAKER:ABORTION:MEASURE
				NASA
				CHIEF
				LAW
				WOMEN

				CHALLENGE
		REVIEW	HILLARY:CLINTON	
			WIND	
			IOWA	
			HIT	
			CAMPAIGN	
			WHITE	
			2008	
			REPUBLICAN	
			EARLY	
			MOVE	
		POLICE		
	TRAVEL	CRUISE:EUROPEAN:FAMILY:TREND:PORT		
		HOTEL		
		BLUE		
		GREEN		
		DESTINATION		
		GUIDE		
		CARNIVAL		
	POLITIC:VIDEO	PROFILE		
		HOUSE		
		PRESIDENT:BUSH		
	SPORT	GOLF		
		SOCCER		
		TENNI		
		FOOTBALL		
		FREE		
		BASKETBALL		
		HOCKEY:TEAM		
		BASEBALL		
		BOX		
		SI:MEDIA:KIT		
		COLLEGE		
	BUSINESS:VIDEO	CAR		
		RETIREMENT		
		PLAN		
		PROFIT		
	U.S.:VIDEO	ATOM		
		IRAN		
		NUCLEAR		
		AMERICAN		
		INTERACTIVE		

		MILITARY		
		SOLDIER		
		WORK		

Subsumption Measure Results

These are the Subsumption results from the human judged taxonomies. Due to length they are incomplete. The format is:

```
Parent-term:Child-term:P(Child-term|Parent-term):running average of probability
```

Computer Hardware

topic-cutoff: 0.004

topic-trim-cutoff: 0.3

```
HARDWARE:CARD:0.5:0.9846154:0.5
CARD:VIDEO:0.9637097:0.9335938:0.7318548
VIDEO:GEFORCE:0.9473684:7.258064E-02:0.8036926
VIDEO:TV:0.9333333:0.2258064:0.8361028
VIDEO:SONY:0.5:0.141129:0.7688823
VIDEO:CAMERA:0.9615384:0.8064516:0.8009916
VIDEO:HD:0.8793104:0.2056452:0.81218
VIDEO:LITE-ON:0.3888889:2.822581E-02:0.7592686
VIDEO:ATI:0.8695652:8.064516E-02:0.7715237
CARD:SOUND:0.9949495:0.7695313:0.7938663
...
MEMORY:FLASH:0.9638554:0.7619048:0.7802655
MEMORY:2GB:0.8571429:0.1142857:0.7824621
MEMORY:CPU:0.7405064:0.5571429:0.7812966
MEMORY:FAN:0.9580838:0.7619048:0.7860746
MEMORY:128MB:0.9:8.571429E-02:0.7890727
HARDWARE:NETWORK:0.4401914:0.7076923:0.780127
NETWORK:WIRELESS:0.8478261:0.3732058:0.7818195
WIRELESS:NETGEAR:0.7692308:0.1086956:0.7815124
WIRELESS:FIREWALL:0.9:0.5869565:0.7843335
NETWORK:MICROSOFT:0.3392857:9.090909E-02:0.7739836
Average Prob: 0.7739836
Percent Correct: 74.4186046511628
```

topic-cutoff: 0.005

topic-trim-cutoff: 0.2

```
HARDWARE:CARD:0.5:0.9846154:0.5
CARD:VIDEO:0.9637097:0.9335938:0.7318548
VIDEO:GEFORCE:0.9473684:7.258064E-02:0.8036926
VIDEO:TV:0.9333333:0.2258064:0.8361028
VIDEO:SONY:0.5:0.141129:0.7688823
VIDEO:CAMERA:0.9615384:0.8064516:0.8009916
VIDEO:HD:0.8793104:0.2056452:0.81218
VIDEO:LITE-ON:0.3888889:2.822581E-02:0.7592686
VIDEO:ATI:0.8695652:8.064516E-02:0.7715237
VIDEO:128MB:0.8:6.451613E-02:0.7743713
CARD:SOUND:0.9949495:0.7695313:0.7944239
...
MEMORY:FLASH:0.9638554:0.7619048:0.7325475
FLASH:DRIVE:0.8926554:0.9518072:0.7382656
FLASH:2GB:0.7857143:0.1325301:0.7399018
FLASH:DISK:0.7647059:7.831325E-02:0.7407286
FLASH:1GB:0.8529412:0.1746988:0.7443483
FLASH:RAT:1:0.2710843:0.7523375
FLASH:PORTABLE:0.7083333:0.2048193:0.751004
FLASH:KINGSTON:0.7:8.433735E-02:0.7495039
FLASH:512MB:0.7857143:0.1325301:0.7505385
FLASH:256MB:0.7368421:8.433735E-02:0.7501581
Average Prob: 0.7501581
Percent Correct: 66.6666666666667
```

Sports

topic-cutoff: 0.003
topic-trim-cutoff: 0.2

```
SPORT:NFL:0.6592427:0.9704918:0.6592427
NFL:REGISTER:0.9885057:0.1915368:0.8238742
NFL:STATE:0.948718:0.1648107:0.8654888
NFL:OKLAHOMA:0.9148936:9.576838E-02:0.87784
NFL:SPORTSNATION:1:0.3608018:0.902272
NFL:QUICK:0.9:2.004454E-02:0.9018934
SPORT:NHL:0.6607143:0.9704918:0.8674392
NHL:RANK:0.9178082:0.4486607:0.8737353
NHL:ALL-STAR:0.9774011:0.3861607:0.8852537
NHL:ROSTER:0.9473684:4.017857E-02:0.8914652
...
STAND:FLORIDA:0.7321429:0.1261538:0.8899621
STAND:MICHIGAN:1:3.692308E-02:0.8934008
STAND:TEXA:1:7.692308E-02:0.8966311
NCAA:PLAYER:0.9513678:0.7864321:0.898241
NCAA:WOMEN:0.9611111:0.4346734:0.9000373
SPORT:SCHEDULE:0.6541787:0.7442623:0.893208
SCHEDULE:RECAP:0.7419355:6.628242E-02:0.8891195
SCHEDULE:BOX:0.9313725:0.5475504:0.8902315
SCHEDULE:VIDEO:0.8854167:0.4899136:0.890108
SCHEDULE:HUNT:1:5.475504E-02:0.8928553
Average Prob: 0.8928553
Percent Correct: 85
```

topic-cutoff: 0.003
topic-trim-cutoff: 0.4

```
SPORT:NFL:0.6592427:0.9704918:0.6592427
NFL:REGISTER:0.9885057:0.1915368:0.8238742
SPORT:NHL:0.6607143:0.9704918:0.7694876
NHL:RANK:0.9178082:0.4486607:0.8065677
NHL:ALL-STAR:0.9774011:0.3861607:0.8407344
NHL:ROSTER:0.9473684:4.017857E-02:0.8585067
NHL:STAR:0.8:4.464286E-02:0.8501487
SPORT:MLB:0.6584821:0.9672131:0.8261904
MLB:DEAL:0.9756098:8.928572E-02:0.8427925
MLB:BOND:0.95:4.241071E-02:0.8535132
...
RECAP:CHART:4.347826E-02:3.225806E-02:0.7697395
SCHEDULE:BOX:0.9313725:0.5475504:0.7731069
BOX:SIMMON:0.9938272:0.7892157:0.7776114
BOX:WIRE:0.875:0.7892157:0.7795592
BOX:NIGHT:0.9638554:0.7843137:0.7831728
BOX:CHAMPION:0.4782609:5.392157E-02:0.7773091
BOX:TV:0.8858696:0.7990196:0.7793574
BOX:TITLE:0.3421053:6.372549E-02:0.7712601
BOX:PETER:1:0.7892157:0.7754191
BOX:SCORECARD:0.2753623:9.313726E-02:0.7664895
Average Prob: 0.7664895
Percent Correct: 67.85714285714295
```

News

topic-cutoff: 0.005
topic-trim-cutoff: 0.2

```
NEW:WORLD:0.9916143:0.8240418:0.9916143
WORLD:ROAD:0.9090909:2.096436E-02:0.9503526
WORLD:RETIREMENT:0.75:2.515723E-02:0.8835685
WORLD:IRAN:1:0.2348008:0.9126763
WORLD:FORECAST:1:5.031446E-02:0.9301411
WORLD:CANADIAN:1:1.467505E-02:0.9417842
WORLD:WINDOW:0.7878788:5.450734E-02:0.9197978
WORLD:TEMPERATURE:1:1.886792E-02:0.929823
WORLD:PERSON:1:9.224319E-02:0.9376205
NEW:HEALTH:0.9936575:0.8188154:0.9432241
HEALTH:BIRD:1:1.902748E-02:0.9483856
```

```
HEALTH:VIDEO:0.9636363:0.448203:0.9496565
HEALTH:LIBRARY:1:1.268499E-02:0.9535291
HEALTH:HEART:1:2.959831E-02:0.9568484
HEALTH:DRUG:1:3.805497E-02:0.9597252
HEALTH:MOM:1:1.479915E-02:0.9622424
HEALTH:CLINIC:1:2.114165E-02:0.9644634
HEALTH:LABOR:1:1.902748E-02:0.9664376
HEALTH:BIG:1:0.2854123:0.9682041
HEALTH:CHINA:0.5294118:0.1141649:0.9462644
HEALTH:MENTAL:1:1.268499E-02:0.9488233
Average Prob: 0.9488233
Percent Correct: 95.2380952380952
```

topic-cutoff: 0.003

topic-trim-cutoff: 0.2

```
NEW:WORLD:0.9916143:0.8240418:0.9916143
WORLD:ROAD:0.9090909:2.096436E-02:0.9503526
WORLD:RETIREMENT:0.75:2.515723E-02:0.8835685
WORLD:IRAN:1:0.2348008:0.9126763
WORLD:FORECAST:1:5.031446E-02:0.9301411
WORLD:CANADIAN:1:1.467505E-02:0.9417842
WORLD:WINDOW:0.7878788:5.450734E-02:0.9197978
WORLD:TEMPERATURE:1:1.886792E-02:0.929823
WORLD:PERSON:1:9.224319E-02:0.9376205
NEW:HEALTH:0.9936575:0.8188154:0.9432241
...
BUSINESS:PROFIT:0.972973:8.035714E-02:0.6866588
NEW:U.S.:0.992629:0.7038327:0.6893194
U.S.:ATOM:0.5365854:0.1081081:0.6880028
U.S.:IRAN:0.8839286:0.2432432:0.6896773
U.S.:NUCLEAR:0.8833333:0.1302211:0.6913185
U.S.:AMERICAN:0.8076923:0.1031941:0.6922964
U.S.:INTERACTIVE:1:1.719902E-02:0.6948606
U.S.:MILITARY:0.9821429:0.1351351:0.6972348
U.S.:SOLDIER:0.9444444:0.1670762:0.6992611
U.S.:WORK:0.9130435:5.159705E-02:0.7009991
Average Prob: 0.7009991
Percent Correct: 63.4146341463415
```

Generalization/Specialization Quality Results

These are the Generalization/Specialization Quality results from the human judged taxonomies. The results are incomplete due to length. The format is:

```
Parent-term:Child-term:running Generalization/Specialization Quality running total
```

Computer Hardware

topic-cutoff: 0.004

topic-trim-cutoff: 0.3

```
HARDWARE:CARD:7.12972739711404E-03
CARD:VIDEO:7.83967893079147E-03
VIDEO:GEFORCE:7.9178856194704E-03
VIDEO:TV:8.22836283547948E-03
VIDEO:SONY:1.01778977033697E-02
VIDEO:CAMERA:1.08082499507103E-02
VIDEO:HD:1.13353672153172E-02
VIDEO:LITE-ON:1.18666427975409E-02
VIDEO:ATI:1.20912669823207E-02
CARD:SOUND:1.21714228740842E-02
...
MEMORY:FLASH:4.47801046254436E-02
MEMORY:2GB:4.50774041138781E-02
MEMORY:CPU:4.79019833734342E-02
MEMORY:FAN:4.84525429976885E-02
MEMORY:128MB:4.86049440728293E-02
HARDWARE:NETWORK:5.46713027945711E-02
NETWORK:WIRELESS:5.57060302311227E-02
WIRELESS:NETGEAR:5.59168649554397E-02
```

```
WIRELESS:FIREWALL:5.63740681631666E-02
NETWORK:MICROSOFT:5.80244398785758E-02
------------------------------------
Generalization/Specialization: 5.80244398785758E-02
 Average / 43 : 1.34940557857153E-03
------------------------------------
```

topic-cutoff: 0.005

topic-trim-cutoff: 0.2

```
HARDWARE:CARD:7.12972739711404E-03
CARD:VIDEO:7.83967893079147E-03
VIDEO:GEFORCE:7.9178856194704E-03
VIDEO:TV:8.22836283547948E-03
VIDEO:SONY:1.01778977033697E-02
VIDEO:CAMERA:1.08082499507103E-02
VIDEO:HD:1.13353672153172E-02
VIDEO:LITE-ON:1.18666427975409E-02
VIDEO:ATI:1.20912669823207E-02
VIDEO:128MB:1.23781744079139E-02
...
MEMORY:FLASH:3.49098508699634E-02
FLASH:DRIVE:3.63516330847343E-02
FLASH:2GB:3.67779859598559E-02
FLASH:DISK:0.037058234401599
FLASH:1GB:3.74289231279485E-02
FLASH:RAT:3.74289231279485E-02
FLASH:PORTABLE:3.83711049058686E-02
FLASH:KINGSTON:3.87723762740137E-02
FLASH:512MB:3.91987291491353E-02
FLASH:256MB:3.95422938252601E-02
------------------------------------
Generalization/Specialization: 3.95422938252601E-02
 Average / 36 : 1.09839705070167E-03
------------------------------------
```

Sports

topic-cutoff: 0.003

topic-trim-cutoff: 0.2

```
SPORT:NFL:2.33138129311759E-02
NFL:REGISTER:2.35017551483286E-02
NFL:STATE:2.42381564768176E-02
NFL:OKLAHOMA:2.49611564064423E-02
NFL:SPORTSNATION:2.49611564064423E-02
NFL:QUICK:2.51404053851315E-02
SPORT:NHL:4.83294614956872E-02
NHL:RANK:0.051588214670321
NHL:ALL-STAR:5.23357349798667E-02
NHL:ROSTER:5.25197030924198E-02
...
STAND:FLORIDA:0.141944059852529
STAND:MICHIGAN:0.141944059852529
STAND:TEXA:0.141944059852529
NCAA:PLAYER:0.144893810320894
NCAA:WOMEN:0.146190966768096
SPORT:SCHEDULE:0.164401035446416
SCHEDULE:RECAP:0.165698806677842
SCHEDULE:BOX:0.168252299461557
SCHEDULE:VIDEO:0.172163091059452
SCHEDULE:HUNT:0.172163091059452
------------------------------------
Generalization/Specialization: 0.172163091059452
 Average / 40 : 4.30407727648631E-03
------------------------------------
```

topic-cutoff: 0.003

topic-trim-cutoff: 0.4

```
SPORT:NFL:2.33138129311759E-02
NFL:REGISTER:2.35017551483286E-02
SPORT:NHL:4.66908112588843E-02
```

```
NHL:RANK:4.99495644335181E-02
NHL:ALL-STAR:5.06970847430638E-02
NHL:ROSTER:5.08810528556169E-02
NHL:STAR:5.17246798532486E-02
SPORT:MLB:7.50241051102732E-02
MLB:DEAL:7.52108130458425E-02
MLB:BOND:7.53950388636583E-02
...
RECAP:CHART:0.19825867787979
SCHEDULE:BOX:0.200812170663505
BOX:SIMMON:0.201000618280672
BOX:WIRE:0.205064540876345
BOX:NIGHT:0.206177985303137
BOX:CHAMPION:0.207711716121927
BOX:TV:0.211445719201146
BOX:TITLE:0.214081639142079
BOX:PETER:0.214081639142079
BOX:SCORECARD:0.218713628631962
------------------------------------
Generalization/Specialization: 0.218713628631962
 Average / 56 : 3.90560051128504E-03
------------------------------------
```

News

topic-cutoff: 0.005

topic-trim-cutoff: 0.2

```
NEW:WORLD:1.10615933828328E-03
WORLD:ROAD:1.37084361264356E-03
WORLD:RETIREMENT:2.32954424448661E-03
WORLD:IRAN:2.32954424448661E-03
WORLD:FORECAST:2.32954424448661E-03
WORLD:CANADIAN:2.32954424448661E-03
WORLD:WINDOW:4.05097131513425E-03
WORLD:TEMPERATURE:4.05097131513425E-03
WORLD:PERSON:4.05097131513425E-03
NEW:HEALTH:4.88144567664206E-03
HEALTH:BIRD:4.88144567664206E-03
HEALTH:VIDEO:7.06222154525354E-03
HEALTH:LIBRARY:7.06222154525354E-03
HEALTH:HEART:7.06222154525354E-03
HEALTH:DRUG:7.06222154525354E-03
HEALTH:MOM:7.06222154525354E-03
HEALTH:CLINIC:7.06222154525354E-03
HEALTH:LABOR:7.06222154525354E-03
HEALTH:BIG:7.06222154525354E-03
HEALTH:CHINA:1.65996691820336E-02
HEALTH:MENTAL:1.65996691820336E-02
------------------------------------
Generalization/Specialization: 1.65996691820336E-02
 Average / 21 : 7.90460437239696E-04
------------------------------------
```

topic-cutoff: 0.003

topic-trim-cutoff: 0.2

```
NEW:WORLD:1.10615933828328E-03
WORLD:ROAD:1.37084361264356E-03
WORLD:RETIREMENT:2.32954424448661E-03
WORLD:IRAN:2.32954424448661E-03
WORLD:FORECAST:2.32954424448661E-03
WORLD:CANADIAN:2.32954424448661E-03
WORLD:WINDOW:4.05097131513425E-03
WORLD:TEMPERATURE:4.05097131513425E-03
WORLD:PERSON:4.05097131513425E-03
NEW:HEALTH:4.88144567664206E-03
...
BUSINESS:PROFIT:0.226781334148232
NEW:U.S.:0.227611381326893
U.S.:ATOM:0.235218174998882
```

```
U.S.:IRAN:0.238610249727116
U.S.:NUCLEAR:0.240436124225285
U.S.:AMERICAN:0.242927199800051
U.S.:INTERACTIVE:0.242927199800051
U.S.:MILITARY:0.243202413610948
U.S.:SOLDIER:0.244281803307974
U.S.:WORK:0.244812339299248
------------------------------------
Generalization/Specialization: 0.244812339299248
 Average / 123 : 1.99034422194511E-03
------------------------------------
```

Human Judging Results

The human judged results show each judged pair and the average rating from all six human judges.

Computer Hardware

topic-cutoff: 0.004
topic-trim-cutoff: 0.3

HARDWARE	CARD	2.50	HARDWARE	MOTHERBOARD	3.00
CARD	VIDEO	2.67	MOTHERBOARD	INTEL	2.67
VIDEO	GEFORCE	2.33	MOTHERBOARD	SOCKET	2.17
VIDEO	TV	2.33	HARDWARE	MEMORY	2.83
VIDEO	SONY	1.83	MEMORY	BOX	1.67
VIDEO	CAMERA	2.17	MEMORY	512MB	2.67
VIDEO	HD	2.17	MEMORY	DDR	3.00
VIDEO	LITE-ON	1.83	MEMORY	OCZ	2.67
VIDEO	ATI RADEON	2.50	MEMORY	1GB	2.67
CARD	SOUND	2.50	MEMORY	256MB	2.67
CARD	PCI	2.67	MEMORY	RATING	2.00
HARDWARE	PC	2.67	MEMORY	FLASH	3.00
PC	REVIEW	1.83	MEMORY	2GB	2.67
PC	SYSTEM	1.83	MEMORY	CPU	1.83
SYSTEM	SERVER	2.17	MEMORY	FAN	2.00
SYSTEM	WINDOWS	2.17	MEMORY	128MB	2.67
SYSTEM	CISCO	2.67	HARDWARE	NETWORKING	2.33
SYSTEM	XP	2.83	NETWORKING	WIRELESS	3.00
SYSTEM	PRO	2.00	WIRELESS	NETGEAR	2.67
SYSTEM	LAPTOP	2.67	WIRELESS	FIREWALL	2.33
SYSTEM	APPLE	2.83	NETWORKING	MICROSOFT	2.17
SYSTEM	RAM	2.50			

topic-cutoff: 0.005
topic-trim-cutoff: 0.2

HARDWARE	CARD	2.50	PC	REVIEW	1.83
CARD	VIDEO	2.67	PC	ATHLON	2.50
VIDEO	GEFORCE	2.33	PC	PENTIUM	2.50
VIDEO	TV	2.33	HARDWARE	MOTHERBOARD	3.00
VIDEO	SONY	1.83	MOTHERBOARD	INTEL	2.67

VIDEO	CAMERA	2.17	MOTHERBOARD	SOCKET	2.17
VIDEO	HD	2.17	HARDWARE	MEMORY	2.83
VIDEO	LITE-ON	1.83	MEMORY	BOX	1.67
VIDEO	ATI RADEON	2.50	MEMORY	FLASH	3.00
VIDEO	128MB	2.17	FLASH	DRIVE	2.00
CARD	SOUND	2.50	FLASH	2GB	2.83
SOUND	CREATIVE LAB	2.00	FLASH	DISK	2.00
SOUND	AUDIO	2.17	FLASH	1GB	2.83
SOUND	SORT	2.17	FLASH	RATING	1.83
SOUND	SIIG	1.67	FLASH	PORTABLE	2.33
SOUND	MIDI	2.00	FLASH	KINGSTON	2.33
SOUND	GOLD	2.00	FLASH	512MB	2.83
HARDWARE	PC	1.67	FLASH	256MB	2.83

News

topic-cutoff: 0.005
topic-trim-cutoff: 0.2

NEWS	WORLD	2.50	HEALTH	VIDEO	2.20
WORLD	ROAD	2.00	HEALTH	LIBRARY	2.17
WORLD	RETIREMENT	2.17	HEALTH	HEART	2.50
WORLD	IRAN	2.67	HEALTH	DRUG	2.50
WORLD	FORECAST	2.17	HEALTH	OLDEST MOM	2.17
WORLD	CANADIAN	2.17	HEALTH	CLINIC	2.33
WORLD	WINDOWS VISTA	2.00	HEALTH	LABOR	2.50
WORLD	TEMPERATURE	2.17	HEALTH	BIG	1.67
WORLD	OLDEST PERSON	2.17	HEALTH	CHINA	1.83
NEWS	HEALTH	2.83	HEALTH	MENTAL	2.67
HEALTH	BIRD FLU	3.00			

topic-cutoff: 0.003
topic-trim-cutoff: 0.2

NEWS	WORLD	2.5	WAR	HIRE GIULIANI	2
WORLD	ROAD	2	WAR	N.H. GOP CHAIR	2.17
WORLD	RETIREMENT	2.17	TV	REPORT	2.67
WORLD	IRAN	2.67	REPORT	EARTH	2.17
WORLD	FORECAST	2.17	REPORT	PROBE	2.17
WORLD	CANADIAN	2.17	REPORT	S.D. LAWMAKER ABORTION MEASURE	2
WORLD	WINDOWS VISTA	2	REPORT	NASA	2.17
WORLD	TEMPERATUR E	2.17	REPORT	CHIEF	2.17
WORLD	OLDEST PERSON	2.17	REPORT	LAW	2.17
NEWS	HEALTH	2.83	REPORT	WOMEN	2.17

HEALTH	BIRD FLU	3	REPORT	CHALLENGE	2
HEALTH	LIBRARY	2	ENTERTAINMENT	REVIEW	2.5
HEALTH	HEART	2.5	REVIEW	HILLARY CLINTON	2.17
HEALTH	DRUG	2.5	REVIEW	WIND	1.83
HEALTH	OLDEST MOM	2.17	REVIEW	IOWA	1.83
HEALTH	CLINIC	2.5	REVIEW	HIT	1.83
HEALTH	LABOR	2.33	REVIEW	CAMPAIGN	2.17
HEALTH	BIG	1.67	REVIEW	WHITE HOUSE	2.17
HEALTH	CHINA	1.83	REVIEW	2008	2
HEALTH	MENTAL	2.83	REVIEW	REPUBLICAN	2.17
NEWS	ENTERTAINMENT	2.5	REVIEW	EARLY	2
ENTERTAINMENT	TV	3	REVIEW	MOVE	2
TV	QUESTION	2	ENTERTAINMENT	POLICE	2
QUESTION	AP	1.67	NEWS	TRAVEL	2.5
QUESTION	CHENEY	1.83	TRAVEL	CRUISE	3
QUESTION	COLTS	1.83	TRAVEL	HOTEL	3
QUESTION	CASE	1.83	TRAVEL	BLUE	1
QUESTION	LIVE	1.83	TRAVEL	GREEN	1
QUESTION	VOTE	2	TRAVEL	DESTINATION	2.83
QUESTION	PART	1.83	TRAVEL	GUIDE	3
QUESTION	STAND	1.83	TRAVEL	CARNIVAL	2.5
QUESTION	TRIAL	2.17	NEWS	POLITICS	3
TV	SHOW	3	POLITICS	PROFILE	2.17
SHOW	PHOTO	2	POLITICS	HOUSE	2.67
SHOW	JOHN	1.67	POLITICS	PRESIDENT	2.83
SHOW	TROUBLE	1.67	NEWS	SPORTS	2.67
SHOW	MISSION	1.83	SPORTS	GOLF	3
SHOW	DOWNLOAD	2	SPORTS	SOCCER	3
SHOW	ALAN	1.83	SPORTS	TENNIS	3
SHOW	EXCLUSIVE	1.83	SPORTS	FOOTBALL	3
SHOW	BIO	2	SPORTS	FREE	1.83
SHOW	PICK	2	SPORTS	BASKETBALL	3
SHOW	WALL	2	SPORTS	HOCKEY TEAM	3
SHOW	FAMOUS	2	SPORTS	BASEBALL	3
SHOW	FACE	2	SPORTS	BOXING	3
SHOW	OFFICIAL	2	SPORTS	SI MEDIA KIT	2.17
SHOW	RULE	1.83	SPORTS	COLLEGE	2.83
SHOW	TURN	1.83	NEWS	BUSINESS	2.83
TV	WAR	2	BUSINESS	CAR	2.33
WAR	FULL	1.33	BUSINESS	RETIREMENT	2.67

WAR	COVERAGE	2.83	BUSINE SS	PLAN	3
WAR	CONGRESS	2.17	BUSINE SS	PROFIT	3
WAR	BILL	2	NEWS	U.S.	2.33
WAR	LEVEL	2	U.S.	ATOM	1.5
WAR	MISUSED ISRAEL BOMB	2.33	U.S.	IRAN	2
WAR	RESPOND CHENEY	2.17	U.S.	NUCLEAR	2.67
WAR	HAGEL CRITICISM	2	U.S.	AMERICAN	2.83
WAR	LEAVE	2	U.S.	INTERACTIVE	1.5
WAR	PASTRY CHEF	1.17	U.S.	MILITARY	2.67
WAR	DENOUNCE ATTACK	2.17	U.S.	SOLDIER	2.5
WAR	TERROR SUSPECT PLEAD	2.17	U.S.	WORK	2.17
WAR	GUILTY	2.17			

Sports

topic-cutoff: 0.003
topic-trim-cutoff: 0.2

SPORTS	NFL	3.00	MLB	INJURY	2.17
NFL	REGISTER	2.00	MLB	PLAYER	3.00
NFL	STATE	2.17	MLB	STANDINGS	2.50
NFL	OKLAHOMA	2.17	SPORTS	NBA	3.00
NFL	SPORTSNATION	2.17	NBA	DRAFT	2.67
NFL	QUICK HIT	1.67	NBA	HISTORY	2.50
SPORTS	NHL	2.67	NBA	TORONTO	2.33
NHL	RANK	2.17	SPORTS	NCAA	3.00
NHL	ALL-STAR	2.33	NCAA	STATS	2.83
NHL	ROSTER	2.33	NCAA	STANDINGS	2.50
NHL	STARS	2.33	STANDINGS	FLORIDA	2.67
SPORTS	MLB	2.83	STANDINGS	MICHIGAN	2.50
MLB	DEAL	2.17	STANDINGS	TEXAS	2.50
MLB	BONDS	2.17	NCAA	PLAYER	3.00
MLB	REDS	2.50	NCAA	WOMEN	2.67
MLB	ODDS	2.50	SPORTS	SCHEDULE	2.67
MLB	SIGN	2.17	SCHEDULE	RECAP	2.67
MLB	HALL	2.17	SCHEDULE	BOXING	2.67
MLB	STATS	2.83	SCHEDULE	VIDEO	1.50
MLB	TRANSACTION	2.17	SCHEDULE	HUNTING	1.50

topic-cutoff: 0.003
topic-trim-cutoff: 0.4

SPORTS	NFL	3.00	STATS	LEADER	2.83
NFL	REGISTER	2.00	STATS	FUTURE	2.33
SPORTS	NHL	2.67	NCAA	STANDINGS	2.50
NHL	RANK	2.17	STANDINGS	STATE	2.33

NHL	ALL-STAR	2.33	STANDINGS	FLORIDA	2.67
NHL	ROSTER	2.33	STANDINGS	MICHIGAN	2.50
NHL	STARS	2.33	STANDINGS	TEXAS	2.50
SPORTS	MLB	2.83	NCAA	WOMEN	2.67
MLB	DEAL	2.17	WOMEN	WNBA	2.83
MLB	BONDS	2.17	WOMEN	STATE	2.00
MLB	REDS	2.50	WOMEN	FLORIDA	1.83
MLB	ODDS	2.50	WOMEN	TV	2.17
MLB	SIGN	2.17	WOMEN	SCOREBOARD	2.00
MLB	HALL	2.17	SPORTS	SCHEDULE	2.67
MLB	STATS	2.83	SCHEDULE	RECAP	2.67
MLB	TRANSACTION	2.17	RECAP	JONES	2.17
MLB	INJURY	2.17	RECAP	FULL	2.17
MLB	STANDINGS	2.50	RECAP	SHOT	2.17
MLB	PHOTO	2.50	RECAP	CHART	2.17
SPORTS	NBA	3.00	SCHEDULE	BOXING	2.67
NBA	DRAFT	2.67	BOXING	SIMMON	2.17
DRAFT	OKLAHOMA	2.20	BOXING	WIRE	1.83
DRAFT	STATE	2.17	BOXING	NIGHT	2.00
DRAFT	SPORTSNATION	1.17	BOXING	CHAMPION	2.33
DRAFT	HISTORY	1.83	BOXING	TV	2.00
SPORTS	NCAA	3.00	BOXING	TITLE	2.33
NCAA	STATS	2.67	BOXING	PETER	1.83
STATS	STATISTIC	2.83	BOXING	SCORECARD	2.33

CURRICULUM VITAE

Joseph Paul Elliott

8204 Vaughn Mill Road jpelli01@louisville.edu
Louisville, KY 40228 (502)-439-3311

EDUCATION

Masters of Engineering, Computer Engineering and Computer Science, Expected April 2007.
University of Louisville, Louisville, Kentucky
Thesis: Automatic Pure Anchor-Based Taxonomy Generation from the World Wide Web
Advisor: Dr. Antonio Badia

Bachelor of Science, Computer Engineering and Computer Science, April 2003
University of Louisville, Louisville, Kentucky

RESEARCH INTERESTS

- Artificial Intelligence
- Data Mining
- Information Sciences

WORK EXPERIENCE

Software Developer, 2001 - Present
Yum Brands Incorporated, Louisville, Kentucky