I, Abhijit Purshottam Joshi, hereby submit this as part of the requirements for the degree of:

Master of Science

in:

Computer Science

It is entitled:

Contextual Decomposition of Web Resources: Applying Semantic Graph Analysis to Personal URL Sets

Approved by:

Dr. Fred Annexstein
Dr. Kenneth Berman
Dr. Yizong Cheng
CONTEXTUAL DECOMPOSITION OF WEB RESOURCES: APPLYING
SEMANTIC GRAPH ANALYSIS TO PERSONAL URL SETS

A thesis Submitted to the
Division of Research and Advanced Studies
of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of
MASTER OF SCIENCE
in the department of Electrical and Computer Engineering
and Computer Science of the College of Engineering
2002

by
Abhijit Purshottam Joshi

Diploma in Advanced Computing, Center for Development of Advanced
Computing, Bangalore, India, 1998
B.E. (Mechanical), KES’s College of Engineering, India, 1996
M.S., University of Cincinnati, USA, 2002

Committee Chair: Dr. Fred Annexstein
Abstract:

In this work we use structural analysis of semantic graphs of categories and URLs to find clusters of URLs, which belong to more than one predefined categories. All the present manual as well as automatic classifiers give a deeply rooted and complex category structure, which is unusable for classification of sets of personal URLs. We have developed software which, given a set of URLs and their set of predefined categories, tries to find sets of personalized categories for a user using the properties of Bipartite URL-Category Semantic (BUCS) network. The structural analysis of this network shows that biconnected components in this graph contain sets of nodes which represent new and refined categories derived from the given hierarchical structure of categories. Also the sets of URLs in these biconnected components represent a more general topic formed by the interrelated individual subjects of their web pages. These sets of URLs are the personalized web communities bound by strong underlying structure of biconnected components in a semantic network of categories and URLs. These refined categories and personalized web communities can be used as part of user’s ‘Personalized Information Environment’.
Acknowledgements

This work has been possible mainly because of my thesis advisor Dr. Fred Annexstein, whose vision about the contextual decomposition of URL data by using a graph theoretic algorithm, propelled my thesis towards a definitive direction. I would like to express my deep gratitude for his able guidance and generous moral as well as financial support, which immensely helped me in overcoming major challenges during the work. I would like to thank Dr. Kenneth Berman for his valuable expert advice on critical issues, for bouncing off different ideas during the lab meetings and for being on my thesis committee. I would also like to thank Dr. Yizong Cheng for listening to my ideas, giving his opinion on them and for being on my thesis committee.

Many thanks to my colleagues Michal Kouril Sergei Gerasenko, Kedar Godbole and Mihalio Jovanovich whose guidance in programming techniques and help in debugging unconventional bugs has been instrumental in getting the personalized classifier software working. At times they made available small but valuable pieces of code which made some jobs such as regular expressions, searching and connecting to a website easy for me. I would also like to thank friends and colleagues, Hongmin, Srinivas, Chris, Swetlana, Sarah and Ambuja for their help in all kinds of critical problems during this work.

Finally I would like to thank my Mother and Father who have been constant source of inspiration all throughout my life.
TABLE OF CONTENTS

Contextual Decomposition of Web Resources: Applying Semantic Graph Analysis to Personal URL Sets

1. Chapter 1 .............................................................................................................4
   1.1. Introduction....................................................................................................4
   1.2. Overview of Some of The Previous Work in Web Classification...............7
   1.3. Problem Definition.......................................................................................10
   1.4. Problems with the Existing Solutions to Categorization.........................11

2. Chapter 2 Our Proposed Approach and its Theoretical Background..............13
   2.1. Structural Properties of BUCS Networks.................................................14
   2.2. Interpretation of Biconnected Components.............................................20
   2.3. Tools and Algorithms Used......................................................................22
       2.3.1. Algorithm to Find Biconnected Components....................................22
           2.3.1.a. Pseudo Code to Identify Articulation Points in a Graph..........23
           2.3.1.b. Time Complexity of the Algorithm..........................................24
           2.3.1.c. LEDA Function to Find Biconnected Components....................25
   2.4. Organization of the Rest of Contents......................................................26

3. Chapter 3 Design of the Software.................................................................27
   3.1 Method Followed to Create Primary Categories.......................................27
   3.2 Graph Drawing............................................................................................32
   3.3 Deleting Temporary Files..........................................................................34
   3.4 How to Use this Program/Software............................................................34
4 Chapter 4 Results and their Analysis .........................................................37

4.1 Examples of Positive Results .................................................................38

4.2 Effect of Size of MAX_Degree of URLs on the BUCS Network ...............60

4.3 Present Limitations of our Approach ......................................................61

4.4 Guidelines for Choosing A Primary Classifier .......................................67

5 Chapter 5 Conclusion and Future Directions .............................................68

5.1 Conclusion .............................................................................................68

5.2 Future Directions ..................................................................................69

5.3 Open Research Questions ......................................................................70

Bibliography .................................................................................................72

Appendix I: Textual Representation of Graphs ..............................................75

Appendix II: Data Collection from the History Folder ................................79
List of Figures

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1</td>
<td>Bipartite Graph of URLs and Categories</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Spring Layout of a Category-URL Graph</td>
<td>15</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>A Bipartite Category-URL Graph</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Patterns Observed in a Bipartite Category-URL Graph</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Graphical User Interface of the Software</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>BUeCS Network for Result 1</td>
<td>41</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>BUeCS Network for Result 2</td>
<td>45</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>BUeCS Network for Result 3</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>BUeCS Network for Result 4</td>
<td>57</td>
</tr>
<tr>
<td>Appendix I, Figure A</td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>Appendix I, Figure B</td>
<td></td>
<td>77</td>
</tr>
<tr>
<td>Appendix I, Figure C</td>
<td></td>
<td>78</td>
</tr>
</tbody>
</table>
Chapter 1

1.1 Introduction:

In this thesis we consider the problem of classification of web pages or URLs. We focus on the sub-problem of classification of personal set of URLs such as those contained in user’s Favourites folder or History folder. All the present attempts for classifying the web (e.g. Yahoo\textsuperscript{TM}! Directory [2], looksmart.com [3], dmoz.org [4], Google\textsuperscript{TM} [1]) are directed towards classifying the whole web or at least a significant part of the web. But these methods classify URLs according to a specific pattern, which generally are not personalized for a particular user. Also public directories like Yahoo! [2], Open Directory [4] have a complex and multi-layer hierarchical category structure. This makes it difficult for a user to classify his/her personal set of URLs using them. Typically people use Favorites\textsuperscript{φ} folder to store their personal URLs. The URLs in the History folder or the keywords used by users for web search are other examples of user’s personal profile. We tried using Google\textsuperscript{TM} [1] to classify the URLs from a user’s Internet Explorer\textsuperscript{TM}’s History folder. When we allowed a link to be categorized into at the most 3 or 4 categories, it created a deeply rooted hierarchical structure with number of categories almost equal to the number of URLs in the history file. It is difficult for a user to handle and keep track of so many categories. Also with this method the URLs get sparsely distributed among the categories. Hence, because of non-personalized nature of these categories, there is little practical utility of this method.

\footnote{Bookmarks folder for Netscape users.}

Note: Google is a trade mark of www.google.com and Internet Explorer is a trade mark of Microsoft, Inc.
There are a number of other automatic classifiers which use machine learning techniques to classify web pages e.g. probabilistic Bayesian Models [11, 12], Neural Networks [9, 13] etc. However these classifiers depend upon having some initial labeled training data from which category models are learned. But in case of these personal sets of URLs there is no training data, therefore these techniques cannot be used to classify them. Hence the limitations of these existing methods/classifiers when used to classify a personal set of URLs, prompted us to take a new approach to such classification. Here we propose a method to automatically generate personalized categories represented by a cluster of categories. We call these automatically generated categories “**Personal Meta-categories**” since these are sets of categories derived from a given hierarchically structured categories. Our approach to solve this problem of finding good personal meta-categories can be explained as follows:

Given a set of personal URLs, and an existing hierarchical category structure, the first order classification of these URLs can be done using any general URL classifier which may be an automatic classifier or a manual classifier e.g. Open Directory [4], Yahoo! [2] etc. If a URL is allowed to be classified into more than one category, then this creates many-to-many relationship between the two sets namely URLs and Categories. This relationship can be graphically represented as a semantic graph of categories and URLs. This is a bipartite graph with edges from a set of URLs to a set of Categories. [See Fig 2.3]. We considered the structural properties of this semantic graph to identify blocks or biconnected components in this graph. Each block contains one set of URLs and one set of Categories. We refer the sets of categories contained in these blocks as “**Personal Meta-categories**” derived from the existing hierarchical structure of categories. Also the sets of URLs contained in each blocks is a set of URLs on a related topic.
We refer to this set of URLs in each block as “Personalized Web Community” i.e. a set of related URLs.

Further experiments with various personal sets of URLs led us to consider the density of this semantic network. The density of this semantic graph is governed by the degree of the URLs i.e. the maximum number of categories into which a URL is allowed to be categorized. Higher the degree of URLs higher is the density of the semantic network. Our experiments showed us that degree of size 7 led us to a very high density creating a single huge biconnected component, which is of little practical utility for further processing. A degree of size 1 or 2 created a very sparse semantic graph with a very few biconnected components. We found that a degree of size 3 or 4 is optimal, in the sense that it creates medium sized biconnected components representing meaningful ‘Personal Meta-categories’. We developed a user interface as shown in Figure 3.1 and we conducted a set of experiments to validate our approach. The novelty of the approach lies in the use of biconnected components to generate ‘Personal Meta-categories’ from a given hierarchical structure of categories, specific to a user. This work is proposed to become a part of a framework called Personalized Information Environment (PIE).
### 1.2 Overview of some of the previous Work in web classification.

Categorization is one of the inherent human qualities. Unconsciously we are always comparing things and in turn categorizing things when we say, “this thing is like this other thing”. In fact human mind stores information after automatically classifying it inside the mind. There are many dimensions or views of the data in human mind. The same data can be viewed as categorized textually or semantically. For example sometimes we try to retrieve a name that starts with letter ‘A’ i.e. textually classified data. Or sometimes we try to retrieve the same name by topic like ‘Sports’ if that person is a sportsman/ sportswoman i.e. semantically classified data. This is the case with data in human memory, but after discovery of the paper and books, a lot of data is stored in external resources. We can apply the analogy of Random Access Memory and permanent storage (Hard Disk) to this human memory and books respectively. Hence in order to make the data retrieval easy, naturally there have been efforts to categorize this data. One of the most ancient classification schemes can be exemplified as ancient Asian Indian Classical Music, which has organized musical compositions into standard types called ‘Ragas’. Ragas are specific sequences of musical notes, which follow certain rules. Many songs can be composed based on a specific Raga. So, one of the ways of classifying songs is according to the ‘Raga’ followed by it.

As for the storage of information goes, books on the same subjects were traditionally stored together. The first basic classification schemes were developed by Melville Dewey. He developed a classification system, which is still used today in many libraries around the world. This system is called the Dewey Classification System. [5]. Another classification scheme called The Library of Congress Classification Scheme [6] which allows for greater precision in most fields and more room for expansion than the Dewey Decimal Classification System. Each
Library of Congress classification is represented by a set of capital letters and numbers. The first letter in the set indicates one of 21 major areas of knowledge. Another classification scheme called Superintendent of Documents (SuDocs) classification system [7] is designed to group together publications by the same government author. Within an agency or department, publications are grouped according to the subordinate organization. The purpose of this system is to uniquely identify, logically relate, and physically arrange each publication so that all publications of a single agency or department may be found together.

With the advent of the World Wide Web there has been an information explosion over the Internet. Web directories like Yahoo! [2], LookSmart [3], Google™ [1], Open Directory [4] have been used to classify the web pages. But these are organized manually and hence cannot keep pace with the increasing size of the web. Hence a lot of efforts have been made towards automatically classifying the web pages.

The Web-Classifiers themselves can be classified in many ways such as: Flat Structure vs. hierarchical, binary classifiers, statistical and machine learning classifiers, text classifiers, link structure analysis based classifiers, supervised clustering, library classifiers, hyperlink classifiers etc. Here we mention some of these efforts made by previous researchers towards hierarchical classification of the web content.

A lot of statistical and machine learning techniques have been applied to text categorization e.g. multivariate regression models [8, 9], nearest neighbor classifiers [10], probabilistic Bayesian models [11, 12], decision trees [12], neural networks [9, 13], symbolic rule learning [14, 15], and support vector machines [16, 17]. These approaches, all depend on having some initial labeled training data from which category models are learned. Once category models are trained, new
items can be added with little or no additional human effort [21]. Most of the above mentioned classifiers ignore hierarchical structure and treat each category or class separately, thus “flattening” the class structure. A binary classifier is learned to distinguish each class from all other classes. In this type of classifier, an item may fall into none, one, or more than one category. They also can be considered as m-ary problem where best matching category is chosen. McCallum, et al. [18] describe some interesting three hierarchical collections – Usenet news, Science sub-category of Yahoo!, and company web pages. They used a probabilistic Bayesian framework (naïve Bayes), and a technique called “shrinkage” to improve parameters estimates for the probability of words given classes. The idea is to smooth the parameter estimate of a node by interpolation with all its parent nodes given by the hierarchical organization of classes. Mladnic and Grobelnik [19] examined issues of feature selection for hierarchical classification of web content, but they did not compare hierarchical models with flat non-hierarchical models. Chakrabarti et al. [20] also experimented with web content as well as patent and Reuters data. They use hierarchical structure both to select features and to combine class information from multiple levels of the hierarchy. In the work done by Susan Dumais and Hao chen et al [21], they explore the use of hierarchical structure for classifying a large, heterogeneous collection of web content. The hierarchical structure is initially used to train different second level classifiers. In the hierarchical case, a model is learned to distinguish a second-level category from other categories within the same top level. In the flat non-hierarchical case, a model distinguishes a second-level category from all other second-level categories. They use support vector machine (SVM) classifiers. For the hierarchical approach they found small advantages in accuracy using a sequential Boolean decision rule and a multiplicative decision rule.
In most of the above approaches, it is necessary to have a training set of documents so as to create a model for classification. But in case of personal set of URLs, there is no training set available. Hence these techniques cannot be directly applied to classify the personal set of URLs. In the Next section we define the problem that we are trying to solve here.

1.3 Problem Definition

Presently a user has his/her profile made up of a set of URLs, which may come from:

a) A file stored on their machines in the form of Favorites/Bookmarks.

b) A file stored on their machines in the form of History folder or

c) A set of keywords used by the user to search on the internet.

The URLs in those sets represent a user’s interest or may be their profession. Typically these URLs have to be organized manually. Usually when users perform search over search engines they don’t store the results. Next time when they need similar information they have to either search through the history folder or perform the search again. Presently Microsoft’s Internet Explorer™ stores the URLs in the history folder sorted by date or website. But these are not categorized by topics they represent, which would have been useful for future searches. In short there is no provision for a user to automatically organize their personal data. So we can define the problem in simple terms as, given a personal set of URLs, there needs a way to categorize these URLs into a personalized categories (rather than generalized categories). In addition there is no mechanism for the machine to learn about user’s profile and perform personalized searches.
for the user. Neither is there a feature, which will go out on the Internet and fetch URLs of websites, which may be of interest to the user and classify them into appropriate folders.

### 1.4 Problems with Existing Solutions to Categorization

Classifying user’s Personal set of URLs can be achieved in a number of ways. As discussed earlier in Section 1.2 about earlier attempts at organizing the web contents, these sets of URLs can be processed through any one of these proposed classifiers. A Probabilistic Bayesian text classifier like [13, 16] or an Artificial Intelligence based classifier like [22, 25, 1, 4] each may or may not categorize these URLs into appropriate categories depending upon the training set they use. Of course a manual classifier like Open Directory [4] or Yahoo! Directory [2] will give more accurate results than any other. These manual classifiers have a flat structure i.e. they treat each category or subcategory at the same level and as an individual category. In addition to that these classifiers tend to give deeply rooted category structure, which is difficult to maintain or remember by human beings. For example if you look for a category for the link to university of Cincinnati i.e. http://www.uc.edu, on Google™ Directory [1] then it gives following results:

Category: Reference > Education > Colleges and Universities > North America > United States > Ohio > University of Cincinnati

Whereas a user may like to categorize this links in his/her personal set in a more relevant and short category name as:

*Category: Education > Colleges and Universities > University of Cincinnati*

Another example is, if you lookup for category for the link for Cincinnati Airport (http://www.cvgairport.com) on Google™ Directory then it gives the following result.
Whereas a user may find it convenient to store that link in his/her favorites categorized as:

**Category: Transportation > Airports > Cincinnati International Airport**

By observing these two results you will find that a user will not like to have his/her personal URLs stored in such deeply rooted and complicated categories, for following reasons:

- When a user classifies his/her data they usually use names of categories that are closely related to the topic and easy to remember.

- A user will not tend to categorize his/her data by demographical location of the company or the organization. They will rather name the category by the name of the company/organization or the product it offers.

- A user would usually have not more than 3 or 4 levels of subcategories and not as deep as 7 or 8 levels. The reason being a human mind can efficiently remember 3 or 4 level deep categories, but needs special efforts to remember deeper levels of categorization than that.

- The URLs, which get categorized into more than one category, usually have something in common. Any two categories, which have a significant overlap, can either be merged or can have a higher-level category over them which can be a superset of these two. A user would naturally spot the similarity between sets of URLs by observation and form a super-category for them. But most of the earlier approaches for automatic classification don’t form these kinds of hierarchical structure as such.

These limitations of the present approach make it difficult for the user to use them for personalized classification of the URLs.
Chapter 2

Our Proposed Approach and its Theoretical Background

Our proposed approach tries to tackle some of these limitations with the earlier approaches. It categorizes a user’s personal set of URLs into primary and ‘Personal Meta-categories’. Each URL in this list is categorized into primary categories using any general classifier specified in section [1.2], preferably one that does not need a training set, e.g. Open Directory [4]. An URL can be classified into more than one category. The maximum number of categories into which a link is allowed by user to be categorized by the classifier is called its Max_Degree. Similarly the actual number of categories into which a link is categorized is called its Degree. If URLs and their categories are treated as nodes then their resulting graph is a bipartite graph with edges from a set of URLs to a set of categories.

Figure 2.1 shows a bipartite graph formed by URLs and Categories.

![Bipartite Graph of URLs and Categories](image-url)

Figure 2.1
If we allow a degree of more than 1 for each URL then it forms a semantic network of URLs and Categories. We call this network a “Bipartite URL-Category Semantic (BUCS) network”. BUCS network has some interesting structures. Our observations of BUCS network are as follows:

### 2.1 Structural Properties of BUCS Networks:

A ‘BUCS Network’ with Max_Degree for URLs as 4 is shown in the figure 2.2. It is a snapshot taken from the LEDA graph drawing software.
If you arrange these nodes in the form of a bipartite graph then they would look something similar Figure 2.3.
This graph is **bipartite** as you can see here.

Figure 2.3
A Bipartite Category-URL Graph with Nodes = 475 & Edges = 337

powered by LEDA
Here we see that because of URLs having degree more than 1, there are points of overlap between categories. The more the overlap between two categories more closely related are those two categories in context of URL. But when more than two categories are involved in such overlap structure they form some interesting structures$^1$.

We observe following six basic clusters in this bipartite graph as shown in Figure 2.4.

---

$^1$Note: There are a few empty categories seen at the top of the categories stack in the figure 2.3. The reason for these empty categories is as follows: The category for a URL e.g. http://www.linux.org is given by Google™ Directory as Computers > Software > Operating Systems > Linux > Kernel.

Now in our software, so as to generate the final sub-category i.e. Kernel, it has to create all the previous parent categories starting from ‘Computers’. Since these parent categories may or may not contain any URLs, they can sometimes be empty. We have not removed them from the graph so as not to lose track of their numerical index, because in our software, the categories are assigned a new index as soon as they are generated. If we delete those empty categories then there will be discontinuities in the numerical indices of the categories and it is later on problematic from the computational point of view.
U - Indicates Url

C - Indicates Category

\begin{figure}[h]
\centering
\begin{subfigure}[b]{0.3\textwidth}
\centering
\includegraphics[width=\textwidth]{example1}
\caption{(a)}
\end{subfigure}
\begin{subfigure}[b]{0.3\textwidth}
\centering
\includegraphics[width=\textwidth]{example2}
\caption{(b)}
\end{subfigure}
\begin{subfigure}[b]{0.3\textwidth}
\centering
\includegraphics[width=\textwidth]{example3}
\caption{(c)}
\end{subfigure}
\begin{subfigure}[b]{0.3\textwidth}
\centering
\includegraphics[width=\textwidth]{example4}
\caption{(d)}
\end{subfigure}
\begin{subfigure}[b]{0.3\textwidth}
\centering
\includegraphics[width=\textwidth]{example5}
\caption{(e)}
\end{subfigure}
\begin{subfigure}[b]{0.3\textwidth}
\centering
\includegraphics[width=\textwidth]{example6}
\caption{(f)}
\end{subfigure}
\caption{Patterns Observed in the Category-Url Bipartite Graph}
\end{figure}
These are the constituents of the bipartite graph structure formed by URLs and categories. By intuition we see that structures (e) and (f) are strongly connected structures because they have mutual overlap. Whereas, structures (a), (b), (c) and (d) are weakly connected structures because they don’t have a mutual overlap.

Here are some relevant definitions:

Definition of Connected Components:
Given a graph $G = (V, E)$, with set of nodes $n > 1$ and number of edges $e \geq 0$, a component of this graph is connected if it there is exists a path between every two nodes.

Definition of Biconnected Components:
A vertex $v$ of a connected graph $G$ is an articulation point (also called cut vertex) if the graph $G - v$ obtained from $G$ by deleting vertex $v$ and all incident edges is not connected. A biconnected component of $G$ (also called a block of $G$) is a maximal subgraph $B$ such that $B$ has no articulation points. [22]

If you apply this definition to structures (e) and (f) in Figure 2.4, then you will see that they are biconnected components and hence form strongly connected structures.
2.2 Interpretation of Biconnected components

Biconnected components form a strong underlying structure in the Category-URL Graph for the set of related categories and URLs. Biconnected components show that the categories and URLs contained in them are closely related in the context of the given personal set of URLs.

Definitions:

If \( C = \{C_1, C_2, C_3, \ldots C_n\} \) (where \( n \geq 1 \)) is a set of Categories and

\( U = \{U_1, U_2, U_3, \ldots U_m\} \) (where \( m \geq 1 \)) is a set of URLs, then

Let \( B = \{C_1, C_2, C_3, \ldots C_j, U_1, U_2, U_3, \ldots U_k\} \) \((j,k, \geq 0, j \leq n \text{ and } k \leq m)\) be a BUCS Network such that

\[ V(B) \subseteq C \cup U \]

And

\[ E(B) \subseteq C \times U \]

If \( B^* \subset B \) is a Biconnected Component in this BUCS network then

\( C \cap V(B^*) \) is called “Personal Meta-category” and

\( B^* \cap U \) is called a “Personalized Web Community”.

The set of Categories in the Biconnected component in a BUCS network represents generalized topic which can be a superset of all those categories. Hence this set of Categories is called as ‘Personal Meta-categories’.
The set of URLs in the biconnected components in a BUFS network represents a personalized web community. This web community is different from the one defined in M. Kleinberg’s [24] paper in the manner as follows: The community formed by the hubs and authorities is a part of a ‘simple web graph’ of URLs whereas the community formed here by the biconnected components is a part of a ‘semantic graph of URLs and their Categories’ formed due to mutual reinforcement of the topic they represent. Also the community formed by the hubs and authorities represent a generalized topic on the web whereas the community formed here in a BUFS Network represents user’s personal/ professional interests.

An Intuitive explanation for calling the set of categories contained in the biconnected components of a BUFS network “Personal Meta-categories”, can be given as: This set of categories together represents a broader topic formed by the interrelated subjects represented by individual categories in the set. For example, a set of categories {Psychology, Philosophy, Communication} forming a biconnected component together represents a broader topic of Social Science. Hence this set of categories actually represents a ‘Personal Meta-category’ which may be called as “Social Science” (or any other name suggested by the user). A similar argument is valid for calling the set of URLs contained in these biconnected components of the semantic graph as a “Personalized Web Community”. Since the URLs in that set have been categorized into these related categories, these must be related. Hence these URLs, together must represent a community on the topic of their ‘Personal Meta-category’, i.e. here Social Science.
2.3 Tools and Algorithms Used

As mentioned earlier, it is possible to create a BUCS network by treating categories and URLs as nodes with edges directed from URLs to the categories. We use graph algorithms and visualizations technique to analyze this graph. The tool used is software called LEDA Graph Drawing Software by Algorithmic Solutions Software GmbH, Germany [25]. LEDA stands for Library of Efficient Data types and Algorithms, which is a rich library of data types and algorithms. It has data structures for graph drawing where we can test some standard algorithms or test new algorithms of our own. The subsection 2.3.1 explains the Algorithm to find the biconnected components and Appendix I, explains some textual graph representation techniques which will help in understanding some of the steps in the flow of the program.

2.3.1 Algorithm to Find Biconnected Components

A graph G is not biconnected if and only if there is a single vertex, the removal of which disconnects the graph. Such a vertex is called an articulation point. An edge belongs to exactly one component. But each articulation point belongs to more than one biconnected component. The main task of the algorithm to find all the biconnected components of a graph would be to find these articulation points in the graph. The blocks between these articulation points are the biconnected components of the graph.

The algorithm used here for finding the articulation points of a connected graph G is based on the Depth First Search (DFS) Tree $T_v$ roots at a vertex $v \in V(G)$. Following Propositions are very useful in devising an algorithm for finding the articulation points in a connected graph.
**Proposition I:** Given a vertex $r$ of a connected graph $G = (V, E)$. Consider the DFS tree $T$, rooted at vertex $r$. Then for any vertex $c$ at depth at least two in $T_r$, there exists a path in $G$ not containing $parent(c)$ that joins $c$ and $r$ if, and only if, there exists a bypass edge for $c$. [22]

**Proposition II:** Given a vertex $r$ of a connected graph $G = (V, E)$, consider the DFS tree $T$, rooted at vertex $r$. Then a vertex $v \neq r$ is an articulation point in $G$ if, and only if, there exists a child $c$ of $v$ in $T_r$ having no bypass edge for $c$. [22]

An Algorithm for finding articulation points in $G$ is based on Proposition II and it uses two kinds of numbering called *DFS Numbering*, and *Lowest Numbering*. The first numbering $DFSNum$, is simply the order in which the vertices are (first) accessed by a depth-first search rooted at vertex $r$. The second numbering $Lowest$, is computed recursively as follows:

For a given $v \in V(G)$, $Lowest[v]$ is defined recursively as follows [22]:

\[
Lowest[v] = \min \left\{ \min \{ DFSNum[w] : vw \in E, w \neq parent(v) \} ; \right. \\
\left. \min \{ Lowest[c] : c \text{ is a child of } v \text{ in } T_r \} \right\} ;
\]

For each vertex $v \in V(G)$:

1. $color[u] = gray$
2. $Lowest[u] = d[u] = count = count + 1$
3. for each vertex $v \in Adj[u]$ do

### 2.3.1.a Pseudo Code to Identify Articulation Points in a Graph [23]

**BICONNECTED_COMPONENTS** ($G, u$)

1. $color[u] = gray$
2. $Lowest[u] = d[u] = count = count + 1$
3. for each vertex $v \in Adj[u]$ do
4  if v = p[u] then  /* v isn't parent of u
5       if color[v] = white then
6          Put edge (u, v) on stack S  /* push new edge onto stack S
7          p[v] = u
8          BICONNECTED_COMPONENTS (G, v)
9       if Lowest[v] >= DFNum[u] then   /* u is articulation point, the portion of G
10              /* is searched by DFS (v) forms a biconnected-component
11              Output next component
12       Pop S up to, and including (u, v)
13       Lowest[u] = min {Lowest[u], Lowest[v]}
14   else  /*v  has already been visited
15       if DFNum[v] < DFNum[u] then   /* v is an ancestor of u
16          Lowest[u] = min {Lowest[u], DFNum[v]}
17          Put edge(u, v) on stack S   /* (u, v ) is back edge
18       color[u] = black

2.3.1.b  Time Complexity of the Algorithm

The complexity of the algorithm to find biconnected components is basically the complexity of
the DFS algorithm. The complexity of the DFS is very easy to work out. For every u belong to
V, all v->u are considered. This takes 2m steps. Each edge is stacked and later unstacked, and a
number of comparisons are performed in order to compute the Lowest-points. Also each node is visited exactly once, hence \( n \) operations. So the total complexity is \( O(n + 2m) \sim O(n + m) \).

When we calculated the actual running time of the algorithm, we found following result. For a graph with 4003 nodes and 2971 edges it took about 3 seconds to find all the 27 biconnected nodes. The visual picture of this graph is shown in Figure 4.3. Our Observation was that for a graph of size about 4000 nodes and 3000 edges, the average time to find biconnected components is \( t < 4 \) seconds on a machine with two 1.7 GHz Intel Processors and 1 GB RAM.

### 2.3.1.c LEDA Function to Calculate Biconnected Components

```cpp
int BICONNECTED_COMPONENTS(graph G, edge_array<int>& compnum)
```

BICONNECTED_COMPONENTS computes the biconnected components of the undirected version of \( G \). A biconnected component of an undirected graph is a maximal biconnected subgraph and a biconnected graph is a graph which cannot be disconnected by removing one of its nodes. A graph having only one node is biconnected.

Let \( c \) be the number of biconnected component and let \( c' \) be the number of biconnected components containing at least one edge, \( c - c' \) is the number of isolated nodes in \( G \), where a node \( v \) is isolated if is not connected to a node different from \( v \) (it may be incident to self-loops).

The function returns \( c \) and labels each edge of \( G \) (which is not a self-loop) by an integer in \([0...c' - 1]\). Two edges receive the same label if and only if they belong to the same biconnected component. The edge labels are returned in compnum. Be aware that self-loops receive no label since self-loops are ignored when interpreting a graph as an undirected graph.

This algorithm has running time \( O(|V| + |E|) \) [25].
2.4 Organization of the rest of contents

In the Chapter 3 we will explain the setup for the experiments and flow of the program for finding out the biconnected components. In Chapter 4 we give supporting examples of results from various sets of URLs and their interpretation. This is followed by Conclusion and Future Directions in the Chapter 5.
Chapter 3

Design of the Software

This chapter explains the design of the personalized classifier software. Appendix I describes how to create extract URLs from the History folder of the Internet Explorer™. Section 3.1 describes the method followed by us to do primary categorization of the URLs. Section 3.2 explains the main core of the programs i.e. procedure followed by us to draw Category-URL Bipartite Graph and find the biconnected components in it. Sections 3.4 talk about instructions for using the software.

3.1 Method Followed to Create Primary Categories

In this section we will explain the method that we have developed to create personal Meta-categories and personalized web communities. Here is the description to give an overall idea of working of the software: We use the code given in Appendix I to extract set of URLs from the History Object of user’s browser (Here Internet Explorer™). We then filter this list of URLs using a Perl script called ‘hist’, to remove the unwanted URLs such as Advertisements and Images. The final URL-list is used to create the primary categories and then the Personal Meta-categories for the user. But for the purpose of experimentation, when there are not enough real-life data available, we have developed a method to simulate a user’s profile by generating a URL set using keyword search on the internet. We ask user to enter a few keywords of his/her choice into our user interface and it then make separate searches on Google™ [26] web search engine.
with these keywords. We choose top 90 URLs from each search result per set of keywords and make a list of URLs. These URLs are categorized using Google™ web directories [1] and a bipartite graph of Category-URL is created. We find biconnected components in this graph to create ‘Personal Meta-categories’. The detailed working of the software follows now.

This personalized classifier software has a simple user interface which is shown in Figure 3.1.

A user can enter a keyword into the textbox and click the button “Add Query String”. The keyword will then be displayed in the text-area and will be added to the list of keywords in the file named “search query”. A user can enter as many keywords as he/she wants. Then the user should click button “Calculate Biconnected Components”. This runs the C Shell Script called “mainprogram”, which is given below:
mainprogram

#!/bin/csh
./get_data
./inlink7d4
./overlap
./drawgraph
./bipartite.o
./labelnew
./biconnected.o
./urlmap
./segregate

In this script, 'mainprogram' is a basically a C shell script which calls a number of other Perl scripts and executable programs in sequential order, process the raw data containing the list of URLs. The first Perl script get_data reads keywords one by one from the file search_query and calls a Java program called geturldata.java with keywords as command line arguments. The Java program makes a connection with http://www.google.com using HttpURLConnection object and submits the keyword to the search engine. It then fetches the top 90 results from the search and saves web pages in files [local_path]/temp_input_data/Search_result_page[n]. The Perl script get_data then parses those files and extracts URLs from them. These URLs are stored in the file urllist.txt. This creates the raw input data to be used further.

After collecting all the URLs using those keywords, Perl script inlink7d4 is invoked. This script is at the heart of the software. This script takes URLs one by one from the urllist.txt file and calls with it the program getcategory.java. This program submits the URL to http://directory.google.com. The results page is fetched by the program and saved as Rawcategoryfile[n] (n= 1,2,...n) in the directory [local_path]/output_temp/. This file is then parsed by the Perl script inlink7d4 to find out the category of the URL given by Google™ directory [2]. This category has the following form: for example, category for URL http://www.mit.edu/
Category: Reference > Education > Colleges and Universities > North America > United States > Massachusetts > Massachusetts Institute of Technology.

This is a very deep and complex category. So if the category has more than four levels then we take the first three levels and attach to it directly the last level. For example, the shortened version of this category will be

Category: Reference > Education > Colleges and Universities > Massachusetts Institute of Technology.

This category makes more sense from the personal use point of view of the user. It is easy to remember and manage.

The URL submitted to the Google™ directory sometimes doesn’t return any categories. For example the URL

http://www.strath.ac.uk/Departments/Geography/g/undergrad_courses/third_yr/conservation_of_natural_resources.htm

does not return any category results from the Google™ directory. Now if we truncate this URL by removing the last part conservation_of_natural_resources.htm and try to get the category we still don’t get any category listing. In this manner we keep on truncating the URL till we get the category. When we eventually reach to one level above the base URL and if we still don’t get the category listing, we stop trying for the category. For example we get the category listing for the following truncated form of the URL above.

Url: http://www.strath.ac.uk/Departments/Geography/

Category: Colleges and Universities > Europe > United Kingdom > Scotland > University of Strathclyde > Departments > Arts and Social Sciences
The script inlink7d4 has a variable called “degree” which controls the number of categories into which a URL may be categorized. We found after experimentation that a degree between 3 and 5 is optimal to get evenly distributed biconnected components. A higher degree like 7 gives a highly connected graph structure with a single big biconnected component. A very low degree like 2 gives a sparse graph structure with very few biconnected components. The script creates a file with the name of the category and stores the respective URLs in that file. For example the file Computers_Computer_Science_Theoretical contains following URLs

http://pauillac.inria.fr/algo/AofA/

http://sigact.acm.org/sigact/

http://www.cs.cmu.edu/afs/cs.cmu.edu/user/avrim/www/Randalgs97/home.html

All the categories created so far are stored in a main file called “Categorymain”. The script checks this file before creating any new category so that there is no duplication of categories. While creating categories it creates a unique index for each category. It also creates a unique index for each URLs that gets categorized. URLs which are not categorized do not get an index. This gives a primary categorization of the URLs contained in the user’s profile.

As a byproduct of process of finding categories the script also gives some useful information about the URL and their categories. It creates a file called “graphdegree”. This file contains the URLs and their respective degrees. Another file called “overlap” is created. It contains the information about the overlap between any two categories taken at a time. It has category indices and indices of URLs contained those categories separated by colon. The category names can be looked up by looking in the file “catindexfile”, which matches category names with their unique index. Similarly, URL names can be looked up by looking in the file “Uurlindexfile”.

31
After the execution of this script, next script that gets executed is “overlap”. This script essentially finds out the overlap between any two categories taken at a time. It uses sequential brute force algorithm to calculate overlap between two category files at a time. The file “overlap” contains the information about the overlap between any two categories, whereas file “cuoverlap” contains only the categories which have overlap of at least one URL between them.

3.2 Graph Drawing

The Perl Script ‘drawgraph’ uses the data collected so far in different files and draws a graph of categories and URLs as nodes. It uses the file catindexfile to get the number of categories and number of categories and file Urlindexfile to get the number of URLs in the graph. It now knows the total number of nodes in the graph. In order to know which URL belongs to which category, it uses the file ‘cuoverlap’. This file contains the information about categories and their member URLs in the form of adjacency list. Hence this file actually represents the edges in the graph. The ‘drawgraph’ uses this file and knows about number of edges and their source and destination nodes. This script creates a textual representation of the graph in the most standard form that was shown in Figure A in Appendix I. The name of this graph file created is ‘visual.graph’ which can be read by the graph drawing software LEDA. The script ‘drawgrap’ creates a file called ‘ucatnumber’ which contains the number of categories and URLs in the graph.
In order to create a bipartite graph which is in the ‘gml’ format we use the program ‘bipartite.o’. This program is executable form of the program bipartite.c. This program reads the file ‘visual.graph’ and file ‘ucatnumber’. The file ‘ucatnumber’ tells the program how many nodes to create in each of the two sets of the bipartite graph. It reads the information about the edges from the file ‘visual.graph’. The edges are always directed from URLs towards the categories. The graph file created by the program bipartite.o is called ‘web.gml’. This graph representation has default labels for nodes i.e. sequential indices generated by the ‘graphwin’ program’. Those default node labels are just integers and hence convey very little information about, whether a node is a category or a URL. In order to create more meaningful node labels, we need to use a small script called ‘labelnew’. This script labels categories as C1, C2… and URLs as U1, U2…. The final graph created by the ‘labelnew’ is called ‘user_graph.gml’. This is the bipartite graph with proper node labels and edges directed from URLs to categories.

The next step is to find the biconnected components in this graph. The program biconnected.o contains the algorithm for finding the biconnected components. This algorithm is explained in the section 2.3. The program assigns same edge number to all the members of a biconnected component and creates an edge array structure. For each edge in a biconnected component, we print the label of its source node into a file called ‘bicomp_urlindexes.txt’. This label is in fact the index of the node that represents the URL. Now that we have the list of indices that represent the URLs in the biconnected component, we must use another program to match those indices to the actual URLs. This is achieved by using the script called ‘urlmap’ It creates a file called, ‘bicomp_urllist.txt’. This file contains the sets of URLs from different biconnected components. But these need to separated by a blank line in order to differentiate between different
components. The script ‘segregate’ does the job of creating sets of URLs in the biconnected components. This is stored in the file ‘user_out_file.txt’. This is the final output of the program. This file is then displayed as result in the text area of the user interface for user’s perusal. Similarly we generate a file called ‘user_out_file_cats.txt’, which contains the sets of categories in each biconnected components.

### 3.3 Deleting Temporary Files

After you get the biconnected components which are saved in the file ‘user_out_file.txt’, and ‘user_out_file_cats.txt’, you can save those results in other directory with appropriate filename. So as to perform a new run of the program, you have to first cleanup the old files. This includes some data files as well as temporary files. The script ‘cleanup’ does this job. It deletes all the temporary files created for input, i.e. `[local_path]/temp_input_data/Search_result_input_*`

Also it deletes all the temporary output files, i.e.

`[local_path]/output_temp/Rawcategoryfile_*`

It also moves all the data files created in the last run of the program to a temporary directory called `[local_path]/categories/`

So if you need to refer to any of the files from the last, run you can find it in this directory.

### 3.4 How to Use this Program/Software.

The first step towards using the software is to cleanup the old files. If you want to save the last query and the last results, save these files in some other directory with appropriate names. The files to be saved are ‘search_query’, ‘user_out_file.txt’, and ‘user_out_file_cats.txt’ respectively.
After this you can run the script ‘cleaning’. The detailed working of this script is given at the end of section 3.3. This will delete or move all the old files so that there is no mixing of old and new files. The next step is to recompile the program for graphical user interface i.e. ‘TextEntryBox.java’.

Ideally, this should not be necessary, but sometimes this program creates some temporary class files and doesn’t run properly if run without recompilation. This problem needs some fixing. The method to compile this program is:

```
javac TextEntryBox.java
```

Now the program is ready to be run. Run the program as:

```
java TextEntryBox
```

A Graphical user interface will appear as shown in figure 2.1. You may enter a single keyword or a set of keywords separated by spaces in the text filed and then click the button “Add Query String” after every keyword/s. These keywords will appear in the text area of the graphical user interface.

After you are done entering the keywords you may click the button “Calculate Biconnected Components” to run the program. This will start the program and a number of debugging statements may get output to the background screen. After entering the keywords if you think you want to abort the execution without pressing the button to calculate the biconnected components, then you may click the cross icon on the top right hand corner of the graphical user interface. But once the program starts running, there is no other way to stop the program than killing the processes that this program has spawned. The most important process that will be needed to be killed, is the one named ‘inlink7d4’. This program, as explained earlier is at the heart of the software and does all the processing of the data.
The program may run for a while depending upon following two factors:

1. Number of different keywords entered by the user.
2. The degree of the URLs set in the program inlink7d4.

Higher the value of these two variables, longer will be the running time of the program.

After the program is finished running, the results will appear in the text area of the graphical user interface. The results are also stored in the text file ‘user_out_file.txt’ and ‘user_out_file_cats.txt’. The keywords entered by the user are saved in the file ‘search_query’. You may save these files with appropriate names in some other directory for your reference.
Chapter 4
Results and Their Analysis

We conducted a number of experiments with different sets of URLs to substantiate our theory that, biconnected components in a BUCS network represent ‘Personal Meta-categories’ over the given hierarchical structure of categories. Various sets of personal URLs available to us were in the form of user’s History folders. We used the code given the Appendix II to extract the set of URLs from Internet Explorer’s History Object. Some further processing of this set was necessary to remove the links to advertisements and images which would not actually get categorized. Also in order to avoid redundancy of results and so as to control the density of the BUCS network, we removed the repeated URLs from this set. Since we wanted to experiment with sets of personal URLs of users from various spheres of society, we also tried to simulate the generation of these Personal sets of URLs for some of the results. A general Observation of User’s profiles showed that, a user visits links on the topics of his/her interest on the web by first searching with keywords on the web search engines. Hence we simulated a user’s web searching by submitting keywords the web search engine like Google™ and generating a representative set of URLs. The User Interface shown in Fig 3.1 is used to submit keywords and then it invokes the ‘mainprogram’ to create a BUCS network based on these URLs. The biconnected components are located in this network to give the ‘Personal Meta-categories’ and personalized web communities. Here are some examples of results and their analysis.

First result is using a set of URLs collected from a history folder of an actual user. Rest of the three results are using simulated profiles of users from different spheres of society e.g. geology,
bioinformatics etc. In second such example we have also introduced some noise into the input data by inserting unrelated keywords, to observe their effect on our method.

4.1 Examples of Positive Results

4.1.a Result 1

In this example of the results, we have taken a profile of a person from his/her History folder. It is evident from the results that the this is a profile of a person who probably a Mathematician or Computer Scientist. We experimented with various personal URL sets using different values of the MAX_Degree for the BUCS network. Most of the experiment gave some unique and meaningful results. The results given here are from a BUCS network with MAX_Degree of 3 that contained 890 URLs and 1319 categories and a BUCS network with MAX_Degree 4 that contained 830 URLs and 1504 categories. We give here the most meaningful biconnected components from both these Graphs. The visual representation of this BUCS network is shown in Figure 4.1.

Result 1: Personal Meta-category I (MAX_Degree =4)

Reference>Dictionaries
Computers>Software>Information_Retrieval
Computers>Computer_Science>Academic_Departments>Massachusetts
Science>Math>Publications>Search_Engines
Science>Math>Combinatorics>Graph_Theory
Computers>Artificial_Intelligence>Neural_Networks>People
Reference>Libraries>Library_and_Information_Science>Digital_Library_Development
Computers>Computer_Science>Organizations
Computers>Computer_Science>Distributed_Computing>Publications
Computers>Computer_Science>Database_Theory
Computers>Parallel_Computing>?il=1
Science>Math>Combinatorics>Software
Computers>Software>Operating_Systems>Resources
Computers>Computer_Science>Conferences
This is a rather big set of categories, but when we used a degree of 3 instead of 4, it gave us a much smaller biconnected component, covering fewer topics in mathematics and algorithms. But we found this more meaningful in the sense that it collects most topics related to math and computer science, which is in agreement with the user’s profile.

Here are some more meaningful set of categories from a biconnected component of BUCS network with MAX_Degree equal to 4.

**Result 1: Personal Meta-category I (MAX_Degree = 4)**
- Science>Math>Combinatorics
- Science>Math>Combinatorics>Graph_Theory
- Science>Math>Recreations>Polyominoes

**Result 1: Personal Meta-category II (MAX_Degree = 4)**
- Computers>Software>Operating_Systems>Software
- Computers>Software>Databases>Informix
- Computers>Software>Networking>Message_Queueing
- Computers>Software>Operating_Systems>DOS

**Result 1: Personal Meta-category III (MAX_Degree = 4)**
- Science>Astronomy>Images
- Science>Technology>Space>Publications
- Science>Technology>Space
Figure 4.1 BUCS Network for Result 1
It even catches user’s interest in music and musical instruments in the following biconnected component.

**Result 1: Personal Meta-category IV (MAX_Degree = 4)**

- Arts > Music > Instruments > Harmonica
- Arts > Music > Instruments > Resources

Here is an interesting set of categories in a biconnected component, which catches two different topics. Even though they don’t seem to be related at the first glance, they actually got related recently very closely because of new invention about ‘Primality Testing Method’ at Indian Institute of Technology, India.

**Result 1: Personal Meta-category IV (MAX_Degree = 3)**

- Science > Math > Number_Theory > Primality_Tests
- Computers > Computer_Science > Academic_Departments > India

Here are some other Personal Meta-categories from a BUUCS network with MAX_Degree = 3.

Their topics are quite self explanatory.

**Result 1: Personal Meta-category V (MAX_Degree = 3)**

- Computers > Software > File_Management > Windows
- Computers > Security > Internet
- World > Espanol > Computadoras > Hacking

**Result 1: Personal Meta-category VI (MAX_Degree = 3)**

- Reference > Dictionaries
- Kids_and_Teens > School_Time > Reference_Tools > Dictionaries
- Reference > Dictionaries > Vocabulary_Lists > Word-a-day
- Health > Pharmacy > Drugs_and_Medications

These sets of Personal Meta-categories show that our approach groups together, not only the sets of categories which are closely related but also which are from totally different areas, but they are related to each other semantically or contextually. For example in one of our experiment, we
had collected together URLs on various topics including some on Psychology, Philosophy and Communication. These topics even though are separate fields of study; they are closely related from personal point of view of the user. When we applied our method to the BUCS network formed by these URLs, we found a significant sized biconnected component which collected together the URLs and categories from these three semantically related topic. The user may name that Personal Meta-category something like

Research > Social Science References

Here is the personalized web community of the user from a BUCS network with MAX_Degree 3. This contains URLs related to web search engines.

**Results1: Personalized Web Community I (Topic: Web Search Engines)**

http://cuiwww.unige.ch/meta-index.html
http://notess.com
http://people.yahoo.com
http://searchenginewatch.com
http://searchenginewatch.com/links
http://searchenginewatch.com/reports/sizes.html
http://search.yahoo.com
http://search.yahoo.com/bin
http://search.yahoo.com/search
http://vivisimo.com
http://www2002.org
http://www.acm.org/sigir
http://www.acm.org/sigs
http://www.alltheweb.com
http://www.altavista.com/sites/search
http://www.brightplanet.com
http://www.brightplanet.com/deepcontent
http://www.infotoday.com/searcher
http://www.inktomi.com
http://wwwinvisibleweb.com
http://www.mpi-sb.mpg.de
http://www.mpi-sb.mpg.de/LEDA
http://www.netlib.org/leda
http://www.profusion.com
http://www.searchenginewatch.com
http://www.slim.indiana.edu
http://www.teoma.com
http://www.yahoo.com
Result 1: Personalized Web Community II (*Topics: Mathematics Research*)

http://mathworld.wolfram.com  
http://www.research.att.com/~njas/sequences  
http://www.ams.org/mathscinet  
http://www.math.niu.edu/~rusin/known-math/index

4.1.b Result 2:

First example is of a URL set which represents a person who is probably a computer scientist or a mathematician. This example is specifically given here so that it can be compared with the results from the actual data in the previous example and it will be seen that simulated data gives quite similar results. We created a set of 559 URLs using following keywords:

**Result 2: Keywords**

- algorithms  
- mathematics  
- data structures  
- computer science  
- vlsi design  
- music and mathematics  
- world famous scientists  
- proclivity inclination for music  
- einstein as a philosopher

We then created a BUQS network of Max_Degree of URLs equal to 3 and number of categories 1092. We found 8 biconnected components in this network, of which, a single major biconnected component was representing a ‘Personal Meta-category’ for the user. The rest of the components were too small to be of any practical implication. The categories in this component
Figure 4.2 BUCS Network for Result 2
are on various topics of mathematics and algorithms. They together represent a single ‘Personal Meta-category’ which may be called as “Resources > Mathematics and Algorithms”

Result 2: Personal Meta-category I

Computers>Algorithms>Conferences>Past_Conferences
Computers>Algorithms>il=1
Computers>Artificial_Intelligence>Academic_Departments
Computers>Computer_Science>Academic_Departments>California
Computers>Computer_Science>Academic_Departments>Canada
Computers>Computer_Science>Academic_Departments>Maryland
Computers>Computer_Science>Academic_Departments>New_York
Computers>Computer_Science>Academic_Departments>North_Carolina
Computers>Computer_Science>Academic_Departments>United_Kingdom
Computers>Computer_Science>Theoretical
Computers>Human-Computer_Interaction>Departments
Computers>Parallel_Computing>?il=1
Computers>Programming>Graphics>Algorithms_and_Data_Structures
Computers>Programming>Languages>Directories
Computers>Programming>Languages>FAQs,_Help,_and_Tutorials
Computers>Programming>Languages>Modula-2
Computers>Security>Intrusion_Detection_Systems
Computers>Software>Databases>Data_Warehousing
Computers>Software>Information_Retrieval
Computers>Software>Operating_Systems>Scripting
Computers>Software>Operating_Systems>TinyOS
Reference>Education>Colleges_andUniversities>Research
Reference>Education>Educators>Software
Regional>North_America>United_States>Education
Science>Educational_Resources>?il=1
Science>Math>Academic_Departments>United_States
Science>Math>Combinatorics>Calendars
Science>Math>Combinatorics>Graph_Theory
Science>Math>Combinatorics>References
Science>Math>Combinatorics>Software
Science>Math>Directories
Science>Math>Education
Science>Math>Education>Associations
Science>Math>Education>Directories
Science>Math>Education>Events
Science>Math>Education>Lesson_Plans
Science>Math>Education>Teaching_Resources
Science>Math>Geometry>Computational_Geometry
Science>Math>Geometry>Software
Science>Math>Logic_and_Foundations>Logicians
Science>Math>Number_Theory
Science>Math>Operations_Research
Science>Math>Organizations
Science>Math>Publications>Collections
Science>Math>Research=Mathematicians
Science>Technology>Space>Spacecraft_and_Satellite_Design
Society>Organizations>Nonprofit_Resources
Society>People>Personal_Homepages>Garg
World>Dansk>Videnskab>Matematik

46
The set of URLs in this biconnected component represents a personalized web community. In this case that community is of web pages which are on the topic of mathematics and various algorithms. Here is the Set of URLs from the Biconnected component:

**Result 2: Personalized Web Community I**

http://archives.math.utk.edu/
http://archives.math.utk.edu/topics
http://archives.math.utk.edu/topics/
http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210
http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/ds_ToC.html
http://dimacs.rutgers.edu/
http://mathforum.org/library
http://mathforum.org/library/
http://math.rice.edu/~lanius/Lessons/
http://mathworld.wolfram.com/
http://pauillac.inria.fr/algo/AofA/
http://sigact.acm.org/sigact/
http://standards.nctm.org/
http://www.ai.mit.edu/people/ellens
http://www.c3.lanl.gov/mega-math/
http://www.c3.lanl.gov/mega-math/workbk
http://www.cis.ohio-state.edu/~parent
http://www.cs.berkeley.edu
http://www.cs.berkeley.edu/
http://www.cs.bham.ac.uk/
http://www.cs.brown.edu/cgc
http://www.cs.brown.edu/courses
http://www.cs.colorado.edu/
http://www.cs.colorado.edu/~main
http://www.cs.columbia.edu/
http://www.cs.dartmouth.edu/
http://www.cs.duke.edu/
http://www.cs.jhu.edu/
http://www.cs.nott.ac.uk/
http://www.cs.pitt.edu/~kirk/algorithmcourses/
http://www.cs.princeton.edu/
http://www.cs.sunysb.edu/
http://www.cs.sunysb.edu/~algorithm
http://www.cs.sunysb.edu/~algorithm/
http://www.cs.tamu.edu/
http://www.cs.toronto.edu/
http://www.cs.ubc.ca/
http://www.cs.ucsb.edu/
http://www.cs.umd.edu/
http://www.cs.unc.edu/
http://www.cs.york.ac.uk/
http://www.cut-the-knot.com/
http://www.eduplace.com/math/
http://www.eli.sdsu.edu/courses
http://www.enc.org/
The Visualization of the BUCS network which contains this biconnected component is shown in Figure 4.2.

4.1.c Result 3: Geologist/Environmentalist Profile

Here is another example of results that we achieved by simulating the profile of a Geologist or an Environmentalist. This person is interested in varied topics like geography, environment, classical music, sociology, anthropology, outdoor sports (e.g. hunting) and investment information like stocks, banking etc. This example shows, how biconnected components in a BUCS network of optimal degree of URLs actually categorize a highly heterogeneous set of URLs into highly relevant ‘Personal Meta-categories’. We generated a URL set using following keywords:

Result 3: Keywords

gеography library
conservation of natural resources
environmental protection agency
study of human origin
technology and anthropology
human geography
cultural diversity and the world
Note that, we have added some non relevant keywords such as, “dance clubs” etc. so as to be able to simulate profile of an actual user, where he/she has total freedom of usage of keywords.

This representative set of user’s personal URLs contained 1564 URLs. When these were categorized into primary categories with a URL Max_Degree of 3, it created 2438 Primary Categories. Figure 4.3 shows the BUCS network created with this set of 4003 nodes. It contained total 27 biconnected components, of which 3 components were quite meaningful, representing user’s basic interests. Here is the set of categories from the first biconnected component:

**Result 3: Personal Meta-category 1 : (Topics : Geography, Environmental Science)**

Reference>Education>Colleges_and_Universities>Libraries_and_Museums
Reference>Maps>Libraries
Reference>Maps>Historical
This set of categories represents a ‘Personal Meta-category’ such as:

Research > Social Sciences > Geography and Environmental Science
Edges in the same biconnected component have the same number and articulation points are shown in red.

Figure 4.3    BUCS Network for Result 3
We have following meaningful sets of categories from Biconnected components. Their topics are self-explanatory.

**Result 3: Personal Meta-category II: (Topics: Anthropology Resources)**

Science>Social_Sciences>Anthropology>Organizations  
Science>Social_Sciences>Anthropology>Directories

**Result 3: Personal Meta-category III: (Topics: Anthropology)**

Science>Biology>Evolution>Multiregional_Theory  
Science>Biology>Evolution>Human  
Science>Biology>Flora_and_Fauna>Pan_paniscus  
Science>Social_Sciences>Anthropology>Directories  
Science>Social_Sciences>Archaeology>Prehistory

Here is another Personal Meta-category representing user’s interest in business and investment information.

**Result 3: Personal Meta-category IV (Topics: Financial Investment)**

Business>Investing>Commodities,_Futures>Exchanges  
Arts>Music>Concerts_and_Events>Trade_Shows  
Business>Investing>Commodities,_Futures>Performance_Data  
Business>Investing>Stocks_and_Bonds>Hedge_Funds

This represents a ‘Personal Meta-category’ which may be called as:

**Personal Finances > Investment Information**

The set of URLs contained in these biconnected components forms a personalized web communities based on the user’s topics of interests.

Here are some other meaningful Persoanl Meta-categories. It is up to the user, how he/she wants to name these categories.

**Result 3: Personal Meta-category VI (Topics: Businesses, Financial Services)**

Business>Financial_Services>Banking_Services>United_States  
Business>Small_Business>Start_Up  
Business>Major_Companies>Publicly_Traded>F
Our method even catches the user’s interest in classical music in the following set of categories:

**Result 3: Personal Meta-category VIII (Topics: Classical Music)**

Arts>Music>Reviews>Classical
Arts>Music>Composition>Mozart, Wolfgang_Amadeus
Arts>Music>Composition>Bibliographies

This Personal Meta-category may be called as

*Personal Interests > Classical Music*

The personalized web community formed by one of the biconnected components is given here. It contains the set of URLs related to the topic of geography and environmental science.

**Results 3: Personal Web Community (Topic: Geography and Environmental Science)**

http://atlas.gc.ca/site/english/
http://dnr.state.il.us/
http://fisher.lib.virginia.edu
http://geog.tamu.edu
http://ltpwww.gsfc.nasa.gov
http://memory.loc.gov/ammem/gmdhtml/gmdhome.html
http://muse.jhu.edu/journals
http://oddens.geog.uu.nl
http://onlinebooks.library.upenn.edu
http://www.aafla.org
http://www.aces.uiuc.edu/~hcd
http://www.albany.edu
http://www.colorado.edu
http://www.dcnr.state.al.us/
http://www.dcnr.state.al.us/agfd/
http://www.dnr.state.al.us/
http://www.dnr.state.sc.us/
http://www.dnr.state.wi.us/
http://www.dnr.state.wv.us/
http://www.environment-agency.gov.uk/
http://www.epa.state.il.us
http://www.ftw.nrcs.usda.gov
http://www.geo.ed.ac.uk/home
http://www.geog.gla.ac.uk
http://www.geo.unizh.ch
http://www.ihrinfo.ac.uk/maps
http://www.ithaca.edu
4.1.d Result 4: Bioinformatics Resources

Our next example shows how biconnected components in a BUCS network detect the topical
closeness of two totally different sets of categories and bring them together to form a secondary
category over them. With the new advent of Bioinformatics, Biology/Genomics and Information
Technology have come together. We have used a personal set of URLs of a person who is
interested in Genomics as well as Programming Languages used to detect protein sequences in
genes. The set of keywords used to generate a representative set of URLs are as follows:

Result 4: Keywords
perl
Bioinformatics
computational genomics
java
dna genome
genome sequence analysis
When this set of URLs was used to create a BUCS network of MAX_Degree 5, it contained about 589 URLs and 1035 categories. The BUCS network found to contain 14 biconnected components. The graphical representation of this BUCS network is shown in Figure 4.4. Here is the set of categories, which were contained in the first major biconnected component.

**Result 4: Personal Meta-category I (Topics: Bioinformatics and Programming Resources)**

- Business>E-Commerce>Education>Courses
- Business>Healthcare>Legal
- Computers>Artificial_Intelligence>Neural_Networks>Papers
- Computers>Artificial_Intelligence>Neural_Networks>People
- Computers>Artificial_Intelligence>Neural_Networks>Research_Groups
- Computers>Artificial_Intelligence>Robotics>Directories
- Computers>Computer_Science>Academic_Departments>California
- Computers>Internet>Protocols>Networking_Resources
- Computers>Programming>Internet>References
- Computers>Programming>Languages>Articles
- Computers>Programming>Languages>Asia
- Computers>Programming>Languages>Awk
- Computers>Programming>Languages>Beginner_Level
- Computers>Programming>Languages>Books
- Computers>Programming>Languages>Class_Libraries
- Computers>Programming>Languages>Coding_Standards
- Computers>Programming>Languages>Directories
- Computers>Programming>Languages>Documentation
- Computers>Programming>Languages>FAQs
- Computers>Programming>Languages>FAQs,_Help,_and_Tutorials
- Computers>Programming>Languages>Gentle_Tutorials
- Computers>Programming>Languages>JavaScript
- Computers>Programming>Languages>Magazines_and_E-zines
- Computers>Programming>Languages>Modules
- Computers>Programming>Languages>North_America
- Computers>Programming>Languages>Perl
- Computers>Programming>Languages>Resources
- Computers>Programming>Languages>Scheme
- Computers>Programming>Languages>Tutorials
- Computers>Programming>Languages>Unix_Reconstruction
- Computers>Programming>Languages>Utilities
- Computers>Programming>Languages>XML
- Computers>Software>Databases>MySQL
- Computers>Software:Object-Oriented
- Health>Medicine>Medical_Specialties>Microbiology
- Health>Medicine>Reference>Medline
This set of categories actually represents a Personal Meta-category, which may be called as:

**Research > Bioinformatics and Programming Resources**

Here we see that a user would actually like to keep all the information about Bioinformatics in one category. But when he/she searches on the web and adds those links to his/her favorites folder, they may not get sorted that way unless there is a manual effort. Our approach tries to solve this problem by recognizing the relative vicinity of the topics and getting them together under one ‘Personal Meta-category’.

**Result 4: Personal Meta-category II**
Edges in the same biconnected component have the same number and articulation points are shown in red.

Figure 4.4 BUCS Network for Result 4
This set of categories in the biconnected component is another example of how Biology resources under Bioinformatics have got collected together. A user may call this Meta Category as:

*Research > Bioinformatics & Biology Resources*

Here is another set of categories from a biconnected component, which collects meaningful and related categories together

**Result 4: Personal Meta-category III**

| Science > Biology > Bioinformatics |
| Science > Chemistry > Molecular_Modeling |
| World > FranCais > Sciences > Travaux_Pratiques |
| Computers > Programming > Languages > Software |

Here is the set of URLs from this biconnected component, which represents a personalized web community on the topic of Bioinformatics and its programming language resources.

**Result 4: Personalized Web Community I (Topics: Bioinformatics and its Programming Language Resources)**

http://citeseer.nj.nec.com
http://developer.java.sun.com/
http://genome-www.stanford.edu
http://genome-www.stanford.edu/Saccharomyces
http://hotwired.lycos.com/webmonkey/programming
http://java.about.com
http://javaboutique.internet.com/
http://java.oreilly.com/
http://javascript.com/
http://java.sun.com/docs/books/tutorial/
http://language.perl.com/
http://language.perl.com/faq/
http://lenti.med.umn.edu/
http://mindprod.com
http://perl.apache.org/
http://stein.cshl.org/WWW/
http://stein.cshl.org/WWW/software/CGI/
http://www.ahpcc.unm.edu
http://www.ccs.neu.edu/home
http://www.cio.com/archive
http://www.cis.hut.fi
http://www.cpan.org/
http://www.cs.cmu.edu/Web
http://www.cse.ucsc.edu
http://www.cse.ucsc.edu/research/compbio/
http://www.cs.technion.ac.il
http://www.cs.ucdavis.edu
http://www.cs.ukc.ac.uk
Here is another set of URLs from a biconnected component forming a personalized web community on the topic of Bioinformatics research.
All the three examples above show that, in a BUCS network of optimal Max_Degree of URLs (usually between 3 and 5), a set categories in each biconnected component represent ‘Personal Meta-categories’ and set of URLs in each biconnected component represents a personalized web community. In the next section we will see the effect of size of Max_Degree of URLs on the BUCS network.

4.2 Effect of size of Max_Degree of URLs on the BUCS Network:

Max_Degree, as explained earlier is the maximum number of categories into which a URL is allowed to be categorized. We observed that this factor controls the density of the BUCS network. The density of the BUCS network plays a major role in finding meaningful biconnected components. Relationship between size of Max_Degree and density is direct, i.e. bigger the size of Max_Degree, higher is the density. A very low degree of 1 or 2 gives a sparsely connected graph structure, which has many single nodes and very few biconnected components. On the other hand a high degree, such as 7 or 8 gives a highly connected BUCS network, which again has a very few biconnected components, because there usually is a single huge block connecting most of the nodes. But this block is of little practical value because; it doesn’t represent a meaningful category. We found that the optimal value of degree is between 3 to 5 which gives medium sized (containing 5 to 80 categories) biconnected components, depending on the
closeness of the topics they represent. Here we would like to mention that after choosing the optimal size of Max_Degree, there is one more factor which governs the size of the Biconnected components, that is homogeneity of the topics represented by original set of URLs. By homogeneity we mean the closeness of the topic represented by the set of URLs. Higher the homogeneity, higher will be the overlap between any two categories taken at a time, and in turn higher will be the sizes of the biconnected components. Homogenity of a set of URLs can be stated in simple term as “semantic or contextual closeness” of their topics. For example a URL on the topic of Perl Programming language and a URL on the topic of Java Programming language are ‘semantically close’ to each other. Hence there is a higher chance that they would be found under one biconnected component. Example of contextual closeness is, a URL on topic of Psychology and a URL on topic of Communication may be ‘contextually close’ if both of these URLs are talking about a topic which overlaps these two subjects. So that each of these URLs could get classified under primary categories of ‘Psychology’ and ‘Communication’. This will create a mutual overlap between these two categories and hence a biconnected component will be formed with them. In the following section we will examine the effect of MAX_Degree on the formation of BUCS network and some limitations of our approach.

4.3 Present Limitations of our Approach

Our present approach has some limitations and there are a lot of places for improvement. Here we will summarize a few conspicuous limitations. Our approach while finding ‘Personal Meta-
categories’ makes no assumptions about the primary classifier. It was observed that the hierarchical structure of primary categories plays an important role in giving correct or expected biconnected components.

Following are the limitations of the method to validate our approach:

1) We use Google™ directories to find the primary categories for the URLs. Most of the times Google™ gives correct results, but like any other classifier, there is no guarantee. Its first result is usually accurate. Our observation is that usually first 3 to 4 results from Google™ directory are most relevant and after that the quality deteriorates fast. This puts a limitation on two things; one is size of Max_Degree and quality of the biconnected components. If we choose a Max_Degree value higher than 4 then there is a chance that we get some non-relevant categories for our URLs. These categories may or may not become part of the biconnected components. In case they become part of them, then biconnected components are sometimes seen to have non-relevant categories. This limitation on the quality of results produced by Google™ directory results comes because of following reason: Google™ Directory [2] does not have all the possible URLs categorized. Hence when it is queried with a URL, which is not in its directory, it gives categories for the most relevant web pages that contain that URL. We assume here that the categories of these web pages are relevant to the URL we have queried. Our Observation is that a lot of times this method works, but there is no fixed level of accuracy. It must be noted that, this limitation is a limitation of our method to validate our approach and it is not the limitation of the approach itself.
2) Our approach works based on an assumption that, a primary classifier could classify a URL into more than one category. Biconnected components are formed because there is an overlap between categories. So if there was a classifier which would make binary decisions while classifying the URLs, i.e. a URL can get categorized into at the most one category, then our approach wont work. Because there will be no overlap between the categories. So selection of appropriate primary classifier is necessary for our approach to work.

3) Another limitation due to use of some classifiers like Google™ is that, its category structure affects the BUCS network to some degree. Google™ uses a flat category structure i.e., each category is different than other and at the same level. Now it depends how much detailed categories each classifier uses. . For example Google™ uses following category structure for its category ‘Media News’ i.e. *Television News:*

```
Arts > Television > Networks > ABC
Arts > Television > Networks > FOX
Arts > Television > Networks > NBC
Arts > Television > Networks > UPN
```

Now in this case there is a separate category for each News Channel. So a URL for that news channel will get categorized into only that particular category and it will not fall into a general category called

*Arts > Television > Networks >*

Now, because of this detailed category structure, URLs will get sparsely categorized and there will be little overlap between two News categories. Hence it will be difficult to get a biconnected component which will connect all the news URLs together. That means a secondary category like
“Television > Networks“ is difficult to get formed using biconnected components in this case.

Sometimes the primary classifier has categories based on regional basis e.g the category for URL for Cincinnati Enquirer: http://enquirer.com/today

Regional > North America > United States > Ohio > Localities > C > Cincinnati > News and Media > Newspapers

Since the category is so detailed and specific, it is difficult for any other URL to get categorized into this category, which may be semantically close to this URL. Hence the overlap between the categories will be less and a biconnected component will not be able to locate this closeness of the topic for these URLs.

From this discussion it becomes clear that performance of our approach also depends on the structure and working of the underlying primary classifier, upon which the ‘Personal Meta-categories’ are going to be built.

All these effects are pronounced in the following example, where news sites from all over the world are not found to be contained in one single biconnected node. Instead, they are distributed into different biconnected components, grouped by region or topic they represent e.g. business, sports, technology etc.

Here is the list of keywords used to generate the list of URLs. Intentionally we have created this list mixed with some other keywords, so that resulting set of URLs will able to simulate a personal set of a user’s URLs.

Results 5: Keywords

- ebay
- half.com
- times of india
- indian express
- cricket
- cnn money articles
- cnn news online
- washingtonpost
- tom's hardware
The set of URLs generated with these keywords contained 631 URLs. When it was used to create a BUCS network with Max_Degree of 5, it gave 1360 categories. It has total 22 biconnected components contained in it. Following sets of categories are from various biconnected components:

Here are some examples of how some categories are based on very specific region and hence cannot form a meaningful biconnected component. In this section we are also showing URLs contained in the biconnected component and their respective categories, i.e. we give the sets of biconnected components in the following format,

**Category : URL contained in it**

- Computers_Internet_E-mail_Occasions: http://www.indiatimes.com
- Regional_Asia_India_Guides_and_Directories : http://www.indiatimes.com
- Regional_Asia_India_S: http://www.indiatimes.com/

- Regional_Asia_India_Arts_and_Entertainment:2:http://www.indiaexpress.com/
- Home_Family_Pregnancy_Indian: http://www.indiaexpress.com/

- News_Newspapers_Regional_India:1:http://www.hindustantimes.com/

Here is a biconnected component which catches two related News site URLs together, but does not catch all the news sites together. This is probably because some of its categories have deteriorated in quality. It means that the categories of some of the URLs don’t seem to be relevant to the topic of the URL because the degree used for this BUCS network was high i.e. 5.
Here is an example of how sometimes the primary classifier is unable to classify a URL into relevant categories, because probably the URL is not in the directory actually and instead it is giving out categories of most relevant web pages that contain that URL. Please note that this is a limitation of the primary classifier and not of our approach.

Here is a Biconnected Component which forms a meaningful ‘Personal Meta-category’ which may be called as “Online News Papers”. It catches most of the online US based News Papers, but it does not catch all News Papers online that were contained in the given URL set. This is again because of various reasons specifies earlier in this section.
These are some of the examples of limitations of the underlying primary classifiers which in turn might affect the performance of our second order classifier.

4.4 Guidelines for Choosing the Primary Classifier

Based on these observations we have some suggestions for choosing the primary classifier:

1) The primary classifier should be a flat category structure classifier.

2) It should be able to give more than one category for URLs if there exists any.

3) It would be helpful if it can give the quantitative or qualitative level of relevancy of a URL and its category. This will help in eliminating or limiting the non-relevant categories for the URLs.

4) It is better to have a classifier which does not create deeply rooted categories. We found that a primary classifier with an average depth of 6 to 7 is suitable.

5) Categories starting with “Regional” category as base category are to be avoided because these usually not suitable for personal classification. For example a user will not like to store URLs related to Software Programming by the region in which company of that Software is located. Instead they will store it under more relevant topic as “Computer Programming” or “Software” etc.
Chapter 5

Conclusion and Future Directions

5.1 Conclusion

In this research we have taken one minuscule step further towards the personalization of content management methods. Our new approach of applying link structure analysis to semantic graph of categories and URLs has given very positive results in creating Personal Meta-Categories. These results and others show that it meets the goal of automating personalized categorization of URLs to a fair closeness. The method finds primary categories for the user’s list of URLs and acts as suggestion to the user for combining some sets of categories into their Meta-categories in the context of the given set of URLs. User can control the quality of results by changing the value of the MAX_Degree of URLs in the category-URL graph. It is also observed that the method scales very well for graphs of sizes close to 4000 URLs. It is predicted from the results that it can even scale to higher sized URL lists. This proposed method here is very simple and intuitive. Its personalized nature can find it a place in the ‘Personalized Information Environment’ framework to further enhance the user’s ability to manage their profile.

Another way of looking at the Personal Meta-categories is, it acts as a profile generator for a user. It can be seen from the results that these Meta-categories actually are representing user’s prominent topics of interests i.e. his/her profile. This profile may be further utilized to tailor user’s information environment using other techniques.
### 5.2 Future Directions

There are lot of directions for improvement and utilization of this method. Presently we have developed a prototype of what can be a complete product for personal use of a user. We can generally divide the future directions in two parts, one regarding the improvement in implementation of the method and second regarding application of the method. Following are some of the places for improvement in the implementation of the method.

- First of all, the quality of results can be improved by using a better primary classifier, i.e. a flat category structure classifier which can classify URLs with more accuracy and can predict the percentage relevancy to each category.

- Right now due to the limitations of the primary classifier, all the URLs do not get categorized. This can be improved by using classifier with a broader range.

- While generating categories we need to use searching and comparisons in category list so as to avoid generating duplicate categories. Presently we are using files to store the categories and use sequential search methods to search categories. But this method will be inefficient for a large number of categories. Hence we suggest use of faster search algorithms will further increase the speed. One of the suggestions is to store the category names and URLs in the form of a binary tree arranged alphabetically. This will enable us to use binary search while avoiding redundancy of URLs and Categories. Another method is to use such alphanumeric indices for the categories and URLs that will sort them alphabetically first and then according to their length. The alphabetical part of the index will tell which letter(s) the URL starts with and numerical part will tell the length of the string. This will enable efficient
string searching. This is just similar to using some kind of hash function. So there always is a possibility of using a more efficient hash function.

- Also use of a database instead of a file to store categories and URLs will speed up the search and retrieval of data.

This method of creating Meta-categories has applications in personalization of user information environment. Following are some of the ways in which this method can be utilized:

- This method can be offered as a web service for the users to organize their bookmarks or personal sets of URLs. But it will have to use a local classifier instead of the classifier that we are presently using. Our present classifier takes a longer time to generate categories because it connects to Google Directory [1] to retrieve the categories.

- Another application this method is to work as a helper application for the user. We perform automated searches on the Internet with the user specified keywords to create a simulated set of URLs. This functionality can be used as a part of user’s Personalized Information Environment by learning about user’s profile from the keywords he/she uses to search on the Internet and then by performing these searches again in the background. We can then apply our method to those set of URLs and create Personal Web Communities of URLs and Meta-categories, which may be useful for the user in the future.

5.3 Open Research Questions:

- Presently the percentage of biconnected components in the total number of categories is small. This can be further improved by not using a deeply rooted category structure
like Google Directory [1]. Because such a structure gives sparsely distributed categories and lesser overlap. But if we use a moderately deep-rooted category structure (e.g. about 7 levels deep) then we may get even distribution of URLs over all categories and may be more quantity of biconnected nodes. But it is still an open question to research weather it will give better quality Meta-categories or not.

• Another sub problem of this open question is to study the effect of URL distribution on the quality of the Meta-categories.

• It seems that the future goal of this research would be to find an optimal range of values for,
  • MAX_Degree of the URLs in the Category-URL Graph
  • Depth of hierarchy in the category structure of the primary classifier.
  • Average URL distribution among the primary categories.

• There is another different direction of research than above two. Presently we are using a flat category structure classifier. It will be interesting to experiment with a hierarchical (Tree Structure) classifier where it will be different kind of semantic graph to study. It will be interesting to see weather the level of a category in the hierarchy tree will make any difference while combining two categories to form a Meta-category.

Further work can be done on this method by using different primary classifiers and comparing their results. At the end of this research we hope that this method will help people create a better personalized working environment in the future!
Bibliography


Appendix

Appendix 1

Textual Representation of Graphs

In this section we will explain the textual representation techniques for graphs. This will serve as a background to understand the scripts in our program which convert graph from one representation and then draw them. There are different graph representation standards that can be read by various graph-drawing softwares. A graph can be represented in a very basic manner by representing nodes by symbol |{}| and edges by a pair of integers that represent the node IDs. Figure B shows the textual representation of the visual graph shown in figure A. There are no labels for the graph nodes or edges in this standard. Graph Drawing Software automatically gives labels to nodes according to their index. GML is another textual graph representation standard. This is more descriptive graph standard. In this we can have custom labels and textual string embedded in the graph. Figure C shows a GML representation of the same graph shown in Figure A.
A simple Bipartite Graph

Figure A
LEDA.GRAPH
void
void
8
|{}|
|{}|
|{}|
|{}|
|{}|
|{}|
|{}|
|{}|
|{}|
|{}|

11
1 4 0 |{}
1 5 0 |{}
1 6 0 |{}
1 7 0 |{}
2 5 0 |{}
2 6 0 |{}
2 7 0 |{}
3 4 0 |{}
3 5 0 |{}
3 7 0 |{}
3 8 0 |{}

Figure B

Simple Bipartite Graph Representation
Creator "LEDA write_gml"

```
graph [ 
    directed 1
    node [ 
        id 0
        label "0"
    ]
    node [ 
        id 1
        label "0"
    ]
    node [ 
        id 2
        label "0"
    ]
    node [ 
        id 3
        label "0"
    ]
    node [ 
        id 4
        label "0"
    ]
    node [ 
        id 5
        label "0"
    ]
    node [ 
        id 6
        label "0"
    ]
    node [ 
        id 7
        label "0"
    ]
    edge [ source 0 target 3 ]
    edge [ source 0 target 4 ]
    edge [ source 0 target 5 ]
    edge [ source 0 target 6 ]
    edge [ source 1 target 4 ]
    edge [ source 1 target 5 ]
    edge [ source 1 target 6 ]
    edge [ source 2 target 3 ]
    edge [ source 2 target 4 ]
    edge [ source 2 target 6 ]
    edge [ source 2 target 7 ]
]
```

Figure C. GML Representation of Graph
Appendix II

Data Collection from the History Folder

This section explains one of the methods to create the raw data for our experiments. A user’s History folder of the Internet Explorer™ is actually his/her ‘unorganized’ profile. Here we attempt to organize these personal URLs in the Hierarchical form. For this first we need to extract URLs from the History folder. Though Window’s Explorer displays contents of the History folder as list of URLs, actually they are not stored in the form of a text file. History folder in Windows is actually History (COM) object. We need to implement an interface that connects to this History Object and invoke the function that extracts URLs from the structure that stores the URLs in the History folder. The way this works is explained as follows:

History folder in Microsoft Windows is actually a COM object in the Window’s System directory. In order to gain access to its methods and data structures we need to implement an interface called IUrlHistoryStg2. It inherits from the interface IurlHistoryStg which in turn inherits from the Iunknown.

IurlHistoryStg interface manages Microsoft® Internet Explorer history for the current user. It has following member methods:

IUrlHistoryStg Members

AddUrl: Places the specified URL into the history. If the URL does not exist in the history, an entry is created in the history. If the URL does exist in the history, it is overwritten.
DeleteUrl: Deletes all instances of the specified URL from the history.

EnumUrls: Returns an interface to an enumerator of the history's visited URLs.

QueryUrl: Queries the history and reports whether the URL passed as the pocsUrl parameter has been visited by the current user.

Among all these functions we are interested in the method ‘EnumUrls’. Here is the description of that function:

*IUrlHistoryStg::EnumUrls Method*

Returns an interface to an enumerator of the history's visited URLs.

*Syntax*

HRESULT EnumUrls(IEnumSTATURL** ppEnum);

*Parameters*

ppEnum

[out] Pointer that receives a pointer to the interface to a history enumerator.

This History enumerator called STAT URL contains the list of URLs which we need to extract.

*Return Value*

Returns S_OK if successful, or an error value otherwise.

Thus by implementing the method ‘EnumUrls’ on the History Object we can get the list of URLs and save it in a text file. This file will be further processed by a Perl script called ‘hist’, which filters out the junk like advertisements and search engine results. The reason for filtering the search engine results is that user usually visits the URLs from the search engine results which
he/she considers relevant to their work. So these URLs will be automatically added to the History Object. In addition to this, if we include all the possible search engine results then we will be adding those URLs to the list which were not intended to be visited by the user. But when we simulate a user’s profile we include the search engine results by performing those searches and taking the top 100 results for each query. This gives a URL set close to an actual user’s profile if we select proper keywords and mix them with some other general/non-relevant keywords for randomness. This file thus created is used by our program to categorize by finding biconnected components.