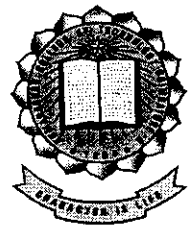
INFORMATION EXTRACTION FROM THE WEB



A PROJECT REPORT

Submitted by

ARTHI G P

SARADHAMANI A

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2010

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report"**INFORMATION EXTRACTION FROM THE WEB**" is the bonafide work of "**ARTHI G.P** and **SARADHAMANI A**" who carried out the project under my supervision.

SIGNATURE

Dr. S. Thangasamy

DEAN

Department of Computer Science & Engg
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore – 641606

SIGNATURE

Mrs. D.Chandrakala

SUPERVISOR

Assistant Professor

Department of Computer Science & Engg
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore – 641606

The candidates with University Register Nos. **71206104004 and 71206104040** were examined by us in the project viva-voce examination held on

.....16.04.10.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be but incomplete without the mention of the people who made this possible and whose constant guidance and encouragement crowns all efforts with success.

We are extremely grateful to **Dr. S. Ramachandran,** Principal, Kumaraguru College of Technology for having given us this opportunity to embark on this project.

We express our sincere and heartfelt thanks to **Dr. S. Thangasamy,** Dean, Department of Computer Science and Engineering, for his kind guidance and support.

We would like to express our sincere thanks to our project coordinator **Mrs. P. Devaki,** Assistant Professor for her valuable guidance during the course of the project.

We would also like to thank our class advisor **Mrs. R. Kalaiselvi,** Senior Lecturer for her constant support and guidance.

We would like to thank our guide **Mrs. D.Chandrakala,**Assistant Professor without whose motivation and guidance we would not have been able to embark on a project of this magnitude. We express our sincere thanks for her valuable guidance, benevolent attitude and constant encouragement.

We reciprocate the kindness shown to us by the staff members of our college, people at home and our beloved friends who have contributed in the form of ideas, constructive criticism and encouragements for the successful completion of the project.

ABSTRACT

ABSTRACT

The World Wide Web is today the main "All kind of information" repository and has been very successful in disseminating information to humans. Web mining, an application of data mining is the integration of information gathered by traditional data mining techniques with information gathered over the World Wide Web .Web pages often contain clutter(such as pop-up ads, unnecessary images) around the body of an article that distracts a user from actual information they are interested in. Information containers (which depict some useful information about a web page) for web documents include the structure of the document , linkage information in the form of incoming and outgoing links and the URL which uniquely identifies a web document.

This project is to extract information from the web analyzes the structural content of web pages through exploiting the information given by links and clustering them using the partitional clustering methods. The approach has been accessed on several URLs and the information clustered is uptodate.

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES

LIST OF ABBREVIATIONS

LIST OF ABBREVIATIONS

1.	DOM	Document Object Model
2.	HTML	Hyper Text Markup Language
3.	tf	Term Frequency
4.	idf	Inverse Document frequency
5.	URL	Uniform Resource Locator
6.	XML	Extended Markup Language
7.	HTTP	Hyper Text Transfer Protocol

CONTENTS

TABLES OF CONTENTS

INTRODUCTION

1. INTRODUCTION:

1.1 DATA MINING:

"Data mining is the process of discovering meaningful new correlations, patterns and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques." There are other definitions:

1) "Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner".

2)"Data mining is an interdisciplinary field bringing togther techniques from machine learning, pattern recognition, statistics, databases, and visualization to address the issue of information extraction from large data bases".

Data Mining, also popularly known as Knowledge Discovery in Databases (KDD), refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases.

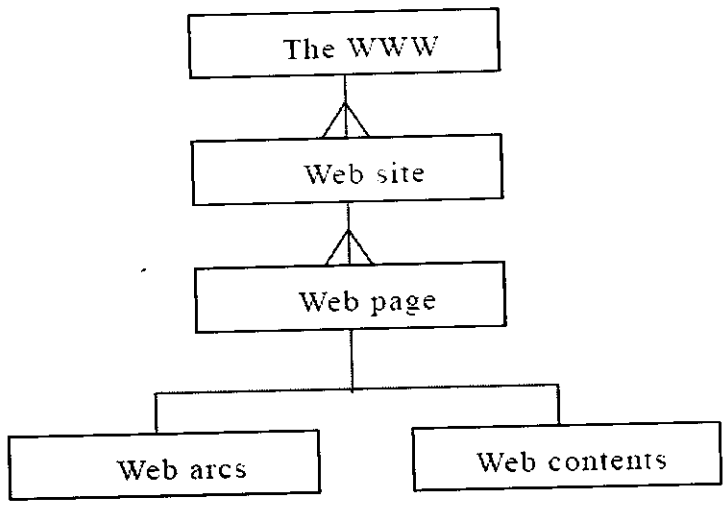
1.2 WEB MINING:

OVER the last decade, we have witnessed an explosive growth in the information available on the World Wide Web (WWW). Today, web browsers provide easy access to myriad sources of text and multimedia data. More than 1 000 000 000 pages are indexed by search engines, and finding the desired information is not an easy task. This profusion of resources has prompted the need for developing automatic mining techniques on the WWW, thereby giving rise to the term "web mining."

The web is a vast collection of completely uncontrolled heterogeneous documents. Thus, it is huge, diverse, and dynamic, and raises the issues of scalability, heterogeneity, and dynamism, respectively. Due to these characteristics, we are currently drowning in information, but starving for knowledge; thereby making the web a fertile area of data mining research with the huge amount of information available online. Data mining refers to the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

Web mining can be broadly defined as the discovery and analysis of useful information from the WWW. In web mining data can be collected at the server side, client side, proxy servers, or obtained from an organization's database. Depending on the location of the source, the type of collected data differs. It Also has extreme variation both in its content (e.g., text, image, audio, symbolic) and meta information, that might be available. This makes the techniques to be used for a particular task in web mining widely varying. Some of the characteristics of web data are

1) unlabeled;
2) distributed;
3) heterogeneous (mixed media);
4) semistructured;
5) time varying;
6) high dimensional.



The schema of the World-Wide Web

Figure 1

Therefore, web mining basically deals with mining large and hyper-linked information base having the aforesaid characteristics.

1.2.1 WEB'S CHARACTERISTICS:

- Large size
- Unstructured
- Different data types: text, image, hyperlinks and user usage information
- Dynamic content
- Time dimension
- Multilingual

1.2.2 STEPS IN WEB MINING:

- **Resource finding**: the task of finding intended web documents.i
- **Information selection and preprocessing:** Automatically selecting and preprocessing specific information from retrieved web resources.
- **Generalisation:** Automatically discover general patterns at individual websites as well as multiple sites.
- **Analysis:** validations or interpretation of the mined patterns.

1.2.3 Types of data on web and its extraction.

Data available on web is classified as structured data, semi structured data and Unstructured data.

Structured Data Extraction.

It is useful to extract structured data from web pages. Some of structured data are list, tree, and data in the form of table. Structured data can be extracted using Wrapper Generation.

Unstructured data Extraction.

Unstructured data is in the form of text document. It is related to text mining, natural language processing and machine learning and web question –answering.

Semi structured data extraction.

Semi structured data are not full and grammatical text. Semi structured data is Hierarchical structured. Semi structured data do not have a predefined structure.

There are several techniques to extract semi-structured data some of them are NLP techniques, wrapper generation, ontology, TINTIN. To extract such data common approach is to build a specific grammar which details the surrounding of each piece of data to extract.

1.3 WEB MINING CATEGORIES

Web mining may be of three types, namely, Web content mining(WCM), web structure mining (WSM), and web usage mining .

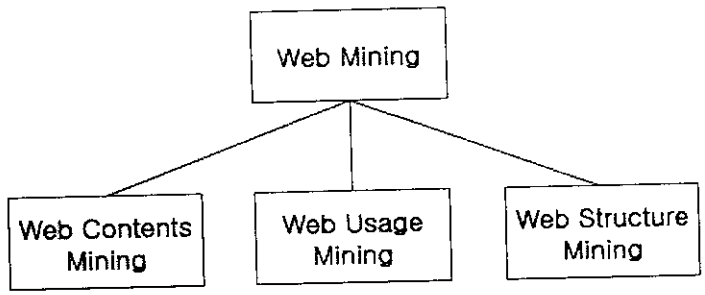


Figure 2 Web mining Classification

1.4 WEB STRUCTURE MINING (WSM)

Web Structure Mining extract patterns from hyperlinks in the web. A hyperlink is a structural component that connects the web page to a different location.It pertains to mining the structure of hyperlinks within the web itself (inter document structure unlike WCM, which pertains to intra document structure). Here, structure represents the graph of the links in a site or between sites.

Web Structure Mining is the process of discovering structure information from the Web .

- This type of mining can be performed either at the (intra-page) document level or at the (inter-page) hyperlink level.
- The research at the hyperlink level is also called *Hyperlink Analysis*

WSM reveals more information than just the information contained in documents. For example, links pointing to a document indicate the

popularity of the document, while links coming out of a document indicate the richness or perhaps the variety of topics covered in the document.

In recent years, Web link structure has been widely used to infer important information about Web pages. By analyzing the pages containing a URL, we can also obtain the anchor text that describes it.

1.4.1 Motivation to study Hyperlink Structure

- Hyperlinks serve two main purposes.
- Pure Navigation.
- Point to pages with authority on the same topic of the page containing the link.
- This can be used to retrieve useful information from the web.

Web structure mining has been largely influenced by research in

- **Social network analysis**
- **Citation analysis**
  - *in-links:* the hyperlinks pointing to a page
  - *out-links:* the hyperlinks found in a page.

By analyzing the pages containing a **URL**, we can also obtain

- **Anchor text**: how other Web page authors annotate a page and can be useful in predicting the content of the target page.

1.5 URL:

Uniform Resource Locators (URLs), also known as Uniform Resource Identifiers (URIs) or occasionally as Universal Resource Locators, are strings of letters, numbers, and special characters that constitute the addresses of documents, files, electronic mailboxes, images, and other

resources in cyberspace. One of the hallmarks of Tim Berners-Lee's invention of the World Wide Web was the ability to make all information on the Internet accessible by a simple click of the mouse, rather than through the tedious process of logging on to various servers and following their own unique interfaces. URLs are the means by which that is accomplished.

A Uniform Resource Identifier (URI) provides a simple and extensible means for identifying a resource. This specification of URI syntax and semantics is derived from concepts introduced by the World Wide Web global information initiative, whose use of these identifiers dates from 1990 and is described in "Universal Resource Identifiers in WWW" .The syntax is designed to meet the recommendations laid out in "Functional Recommendations for Internet Resource Locators" and "Functional Requirements for Uniform Resource Names".

Web pages are identified by their URL's. URL information is good for clustering because, they are small and abiquitious, making techniques based on just URL magnitudes faster than those which make use of the text content as well. We present a system that makes use only URL information to perform clustering of web.

Information container (which depict some useful information about a web page) for web documents include the structure of the documents (based on the markup language used),the unstructures text content, linkage information in the form of incoming and out going links and the URL which uniquely identifies a web document.

1.5.1 Why are URLs Important?

- If you understand URLs and Domain Names, you will usually be able to identify the origin of a Web page or of a piece of email which you have received.
- If you are reasonably certain of the origin of information, you may be in a better position to assess its quality and authenticity.
- If you understand URLs and Domain Names, you may be able to "guess" the likely names of sites, thereby reducing the amount of time you spend searching for information using search engines.
- An understanding of URLs, of Domain Names, and of the file structures and file names used routinely on the net is absolutely essential if you intend to construct and maintain Web resources yourself.

1.5.2 URL Motivation

URL'S are special due to various reasons. Firstly, URLs present structures information. Secondly, URL is the easiest information to obtain about a web page . Thirdly, url stands to be very small entities as opposed to other (useful) knowledge containers for a web page such as the text of the page, the title of the page etc.

General Syntax

URLs in HTTP can be represented in absolute form or relative to some known base URL , depending upon the context of their use. The two forms are differentiated by the fact that absolute URLs always begin with a scheme name followed by a colon. For definitive information on URL syntax and semantics, see "Uniform Resource Identifiers (URI): Generic Syntax and Semantics

The HTTP protocol does not place any a priori limit on the length of a URI. Servers MUST be able to handle the URI of any resource they serve, and SHOULD be able to handle URIs of unbounded length if they provide GET-based forms that could generate such URIs. A server SHOULD return 414 (Request-URI Too Long) status if a URI is longer than the server can handle

1.6 HTTP URL

The "http" scheme is used to locate network resources via the HTTP protocol. This section defines the scheme-specific syntax and semantics for http URLs.

http_URL = "http:" "//" host [":" port] [abs_path ["?" query]]

If the port is empty or not given, port 80 is assumed. The semantics are that the identified resource is located at the server listening for TCP connections on that port of that host, and the Request-URI for the resource is abs_path . The use of IP addresses in URLs SHOULD be avoided whenever possible . If the abs_path is not present in the URL, it MUST be given as "/" when used as a Request-URI for a resource . If a proxy receives a host name which is not a fully qualified domain name, it MAY add its domain to the host name it received. If a proxy receives a fully qualified domain name, the proxy MUST NOT change the host name.

1.7 URL SYNTACTIC COMPONENTS

The URL syntax is dependent upon the scheme. Some schemes use reserved characters like "?" and ";" to indicate special components, while others just consider them to be part of the path. However, there is enough uniformity in

the use of URLs to allow a parser to resolve relative URLs based upon a single, generic syntax.

This generic syntax consists of six components:

<scheme>://<net_loc>/<path>;<params>?<query>#<fragment>

each of which, except <scheme>, may be absent from a particular URL.

These components are defined as follows :-

scheme ":" ::= scheme name

"//" net_loc ::= network location and login information

"/" path ::= URL path

";" params ::= object parameters

"?" query ::= query information

"#" fragment ::= fragment identifier.

The order of the components is important. If both <params> and <query> are present, the <query> information must occur after the <params>.

1.8 APPLICATION AREAS OF WEB MINING

- ☐ E-tailers
  - ■ The ability to find new cross-sell opportunities, enable comprehensive prospect profiling, and improve customer satisfaction.
    - ☐ B2B and B2C Ventures
- ☐ Advertising-Based Sites
  - ■ When the revenue is advertising-based. Blindly serving ads to visitors will not result in a large click-thru rate. Instead, ads must be intelligently targeted to the user, providing the visitor with products and services that they are interested in.

- Entertainment sites
- Media Portals

- advertising Providers
- Information Repositories
  - Information overload is a problem that grows larger every day. Indexing, summarization, and other metadata tasks are time consuming. Semantic text analyzers are capable of automating these tasks, and create user navigation systems on the fly.
    - Libraries
    - Technical Support Sites
    - Media Sites
    - Content Providers
- Security applications
  - One of the largest text mining applications that exists is probably the classified ECHELON surveillance system.
- Software and Applications
  - Research and development departments of major companies, including IBM and Microsoft, are researching text mining techniques and developing programs to further automate the mining and analysis processes.

1.9 CLUSTERING

Clustering refers to the grouping of records, observations, or cases into classes of similar objects. A cluster is a collection of records that are similar to one another, and dissimilar to records in other clusters. Clustering differs from classification in that there is no target variable for clustering. The clustering task does not try to classify, estimate, or predict the value of a

target variable. Instead, clustering algorithms seek to segment the entire data set into relatively homogeneous subgroups or clusters, where the similarity of the records within the cluster is maximized and the similarity to records outside the cluster is minimized.

Existing clustering algorithms can be broadly classifed into

- partitional
- hierarchical

Partitional clustering algorithms attempt to determine partitions that optimize a certain criterion function. In contrast, a hierarchical clustering is a sequence of partitions in which each partition is nested into the next partition in the sequence.

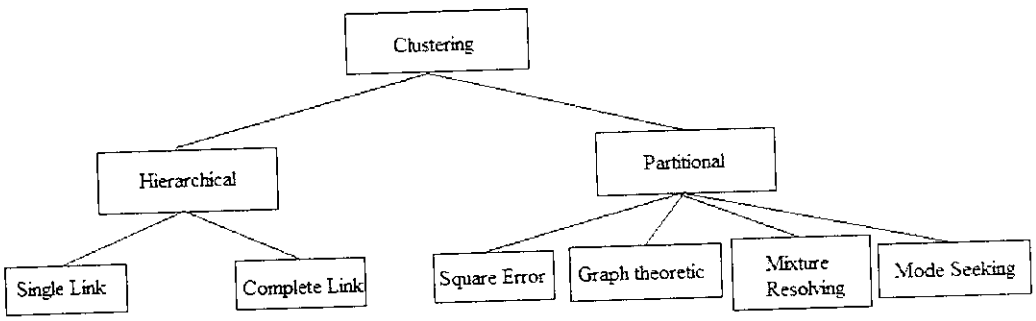


Figure 3 Different approaches to clustering can be described with the help of the hierarchy shown in figure .

1.9.1 Web document clustering

1.9.1.1 Text based Clustering

The text-based web document clustering approaches characterize each document according to its content, i.e. the words contained in it (or phrases or snippets). The basic idea is that if two documents contain many common words then it is very possible that the two documents are very similar. The approaches in this category can be further categorised according to the

clustering method used into the following categories: *partitional, hierarchical, graphbased, neural network-based* and *probabilistic algorithms*

Partitional Clustering.

The partitional or non-hierarchical document clustering approaches attempt a flat partitioning of a collection of documents into a *predefined* number of *disjoint* clusters. More specifically, these algorithms produce an integer number of partitions that optimize a certain criterion function (e.g. maximize the sum of the average pairwise intra-cluster similarities). Partitional clustering algorithms are divided into iterative or reallocation methods and single pass methods. Most of them are iterative and the single pass methods are usually used in the beginning of a reallocation method.

Hierarchical Clustering.

Hierarchical clustering algorithms produce a sequence of nested partitions. Usually the similarity between each pair of documents is stored in a *nxn* similarity matrix. At each stage, the algorithm either merges two clusters (agglomerative methods) or splits a cluster in two (divisive methods).

Fuzzy Clustering.

All the above approaches produce clusters in such a way that each document is assigned to one and only one cluster.Fuzzy clustering approaches. Fuzzy algorithms usually try to find the best clustering by optimising a certain criterion function. The fact that a document can belong to more than one clusters is described by a *membership function.* The membership function calculates for each document a membership vector, in

which the *i*-th element indicates the degree of membership of the document in the *i*-th cluster. The most widely used fuzzy clustering algorithm is Fuzzy c-means a variation of the partitional kmeans algorithm.

1.9.1.2 Link based clustering

Text-based clustering approaches were developed for use in small, static and homogeneous collections of documents. On the contrary, the web is a very large collection of heterogeneous and interconnected web pages. Moreover, the web pages have additional information attached to them (web document metadata, hyperlinks) that can be very useful to clustering. The link-based document clustering approaches characterize the documents by information extracted by the link structure of the collection. The underlying idea is that when two documents are connected via a link, then there exists a semantic relationship between them, which can be the basis for the partitioning of the collection into clusters. The use of the link structure for the clustering of a collection is based on citation analysis from the field of bibliometrics. Link based clustering is an area where Web content and Web structure mining overlap.

Furthermore, the development of link-based clustering approaches has proven that the links can be a very useful source of information for the clustering process. Although there is already much research conducted on the field of web document clustering, it is clear that there are still some open issues that call for more research.

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 PROBLEM DOMAIN:

The problem defines the clustering of link and its content for multiple news websites.The explosive growth and popularity of the World Wide Web has resulted in a huge amount of information sources on the internet. Web pages are identified by their URLs. In this problem, we provide a web-search interface, which presents search results as clusters of results. The interface takes in two parameters, one being the search query and the other being the type of clustering, whether hierarchical or partitional. Each cluster is represented by the set of keyword fragments from the URLs in the cluster. This problem demonstrates hierarchical clustering (using URLs), and topic identification for topical information using URLs (which is used for generating the keyword fragment descriptions to describe the sub-clusters).

The objective of a clustering problem is to group a set of links into a number of clusters. Different clustering algorithms have been used for this purpose. These algorithms can be classified into three main categories: 1) heuristic 2) hierarchical and 3) partition clustering methods . Fuzzy clustering algorithms are partitioning methods that can be used to assign links of the data set to their clusters. These algorithms optimize a subjective function that evaluates a given fuzzy assignment of objects to clusters. Various fuzzy clustering algorithms have been developed, of which the FCM algorithm is the most widely used in applications.

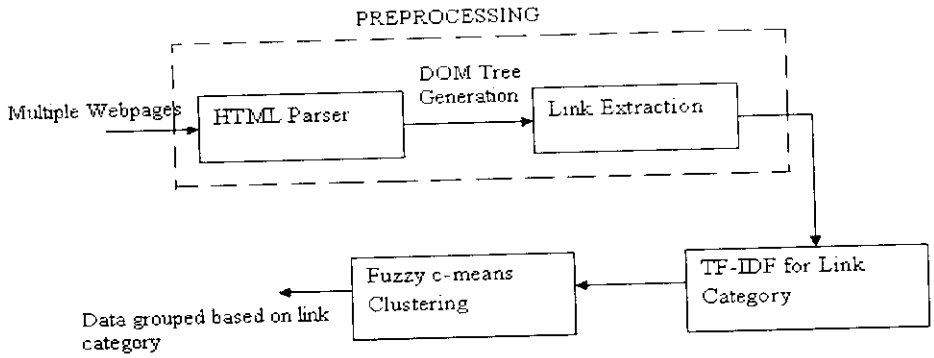


Figure 4 Architecture of Information Extraction

2.2 FUZZY C-MEANS CLUSTERING

Fuzzy C-Means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. This method is frequently used in pattern recognition.

2.3 USES OF CLUSTERING

- If a collection is well clustered, we can search only the cluster that will contain relevant documents.
- Searching a smaller collection should improve effectiveness and efficiency.

2.3.1 ADVANTAGES OF CLUSTERING

- Clustering can work to give users an overview of the contents of a document collection
- Also can reduce the search space

2.3.2 DRAWBACKS OF CLUSTERING

- Computationally expensive
- Difficult to identify which cluster or clusters should be searched

DETAILS OF THE METHODOLOGY

3. DETAILS OF THE METHODOLOGY

URL information is good for clustering because, they are small and ubiquitous, making techniques based on just URL information magnitudes faster than those which make use of the text content as well. We present a system that makes use of only URL information to perform clustering of webpages.

3.1 HTML Parser

HTML::Parser works by scanning HTML input, and breaks it up into segments by how the text would be interpreted by a browser. For instance, this input:

<A HREF="index.html">This is a link</A>

would be broken up into three segments: a start tag (<A HREF=" index.html">), text (This is a link), and an end tag (</A>). In order to analyze a web page for link extraction, we pass web pages through an open source HTML parser, openXMLwhich corrects the markup and creates a Document Object Model tree.

The Document Object Model is a standard for creating and manipulating in-memory representations of HTML (and XML) content. By parsing a webpage's HTML into a DOM tree, we can not only extract information from large logical units ,but can also manipulate smaller units such as specific links within the structure of the DOM tree. In addition, DOM trees

are highly editable and can be easily used to reconstruct a complete webpage.

The DOM tree is hierarchically arranged and can be analyzed in sections or as a whole, providing a wide range of flexibility for our extraction algorithm. Our information extractor navigates the DOM tree recursively, using a series of different filtering techniques to remove and modify specific nodes and leave only the link behind.

3.1.1 Steps for DOM Tree construction

This process first creates a stack to store the names of elements for which a start tag has been encountered, but for which no end tag has yet been encountered. Intially this stack is empty.

- '<? Or '<!--': This is a processing instruction or a comment. Scan the input until a matching'?> or '-->' is found and move the processing point after this match.
- '<!': this is a document type definition block.Scan the input until a matching '>' character is found.
- '<', then determine whether this represents the start of a start tag for an element.
- If the net symbol is a '/' then this is an end tag for an element.

After the DOM tree is completely parsed, the list of removed links is added to the bottom of the page. In this way, any important navigational links that were previously removed remains accessible.

3.1.2 Sequential flow of the process

The process, named "Board Parser", consists of HTML Parser .The primary role of the HTML Parser is to distinguish the plain text which was written by

the Internet user from HTML tags embedded in the same HTML document. Then, the HTML parser creates some object, named "HtmlDocument", which has a vector containing tags' information in the HTML document as a form of tree structure. The outline of "HtmlDocument" object creation is shown in Figure 5.

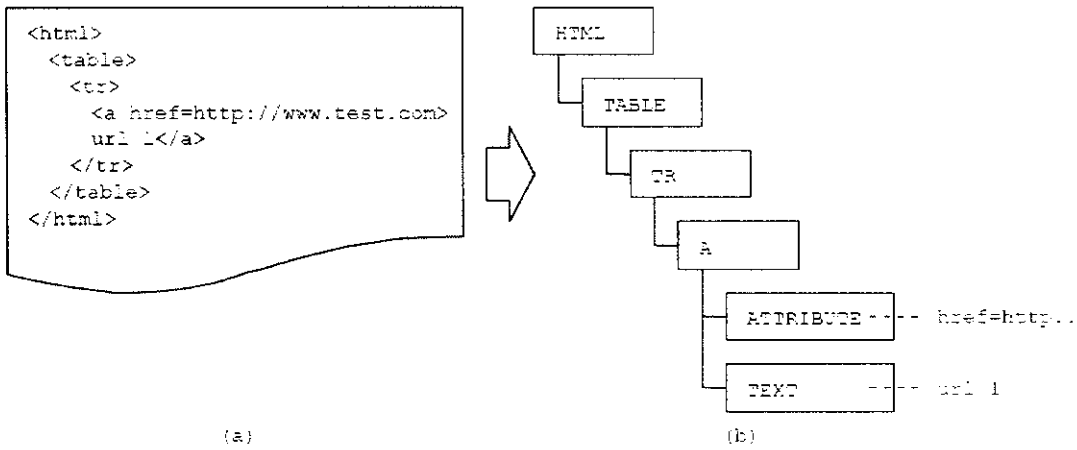


Figure 5 The creation of "HtmlDocument" object. (a) The plain text of HTML document (b) The tree structure of "HtmlDocument" object

3.2 FEATURE SELECTION

After extracting the link in the given URL using the HTML parser , the system calculates multiple important indices for the links in the URL . As for important indices of words and phrases in a corpus, there are some well known indices. **Term frequency divided by inversed document frequency(tf-idf)** is one of the popular indices used for measuring the important of the links. Tf-idf for each link 'l' can be defined as follows

The **term count** in the given document is simply the number of times a given term appears in that document. This count is usually normalized to

prevent a bias towards longer documents (which may have a higher term count regardless of the actual importance of that term in the document) to give a measure of the importance of the term t_i within the particular document d_j. Thus we have the *term frequency*, defined as follows.

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

where $n_{i,j}$ is the number of occurrences of the considered term (t_i) in document d_j, and the denominator is the sum of number of occurrences of all terms in document d_j.

The **inverse document frequency** is a measure of the general importance of the term (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient).

$$\mathrm{idf}_i = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

with

- $|D|$: total number of documents in the corpus
- $|\{d : t_i \in d\}|$: number of documents where the term t_i appears (that is $n_{i,j} \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to use $1 + |\{d : t_i \in d\}|$

Then

$$(\text{tf-idf})_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$$

A high weight in tf –idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. The tf-idf value for a term will always be greater than or equal to zero.

3.3 CLUSTERING

As the input of the web pages, we use the URLs of the news websites like Hindustan times, DD news, Latimes, dnaindia etc....we get the links clustered under various categories. All the links in the URL are clustered under some or the other phrases. This clustering of the links is performed using Fuzzy c means clustering .

FUZZY C MEANS CLUSTERING

Fuzzy clustering approaches, in this project , are non-exclusive, in the sense that each link can belong to more than one clusters. Fuzzy algorithms usually try to find the best clustering by optimising a certain criterion function. The fact that a link can belong to more than one clusters is described by a *membership function*. The membership function calculates for each link a membership vector, in which the *i*-th element indicates the degree of membership of the link in the *i*-th cluster.

Fuzzy C-Means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters.

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m} \left\| x_i - c_j \right\|^2 \qquad (1)$$

Where *m* is any real number greater than 1,.

uij is the degree of membership of *xi* in the cluster *j*; *xi* is the *ith* of d-dimensional Measured data ; *cj* is the d-dimension center of the cluster and ||*|| is any norm expressing the similarity between any measured data and the center.

Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership *uij* and the *cj* cluster centers by

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}} \qquad (2)$$

$$c = \frac{\sum_{i=1}^{N} u_{ij}^{m} x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \qquad (3)$$

This iteration will stop when

$$\max_{ij} \left\{ \left| u_{ij}^{(k+1)} - u_{ij}^{(k)} \right| \right\} < \varepsilon \qquad (4)$$

Where ε is a termination criterion between 0 and 1 and k are the iteration steps.

EXPERIMENTAL RESULTS

4. EXPERIMENTAL RESULTS:

Table 1 : Web pages and the useful links

Overview of the web pages content			
Number of web pages considered	2	5	8
Number of links extracted	73	262	345
Number of useful links	69	257	321

The table 1 consists of webpages considered, extracting the total number of links and useful links.

Table 2:Webpages and its category

Website name	Sports	Business	Politics	Tech	World
CNN	4	6	8	7	25
Yahoonews	12	11	5	16	27
BBC	14	21	7	8	26
Newyorktimes	16	20	5	7	18

The table 2 consists of four news websites and its general categories.

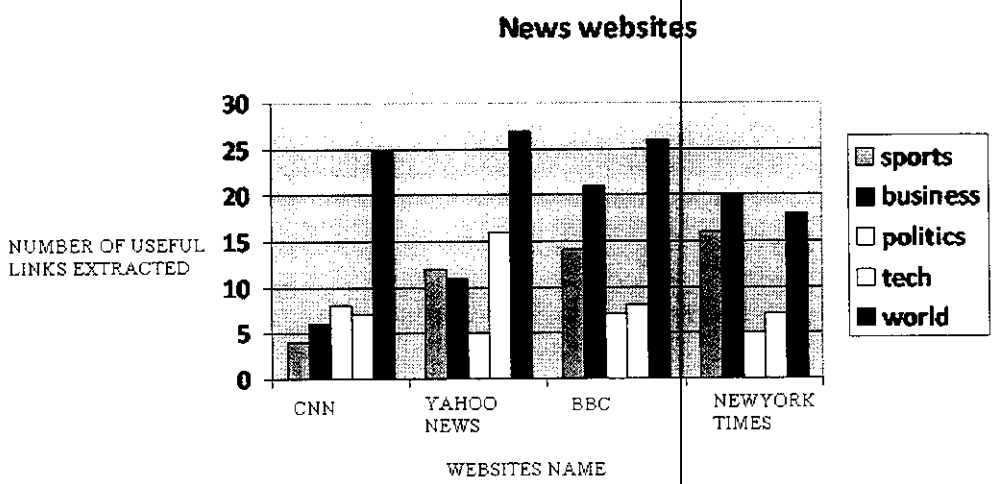


Table 3 :Index value for the category

Number of web pages	tf-idf	Category Name
2	3.828641	Sports
5	4.116232	World news
8	4.521789	Business

The table 3 consists of general categories and its index value from the webpages considered.

Table 4: Clustering using Fuzzy C- means

No of webpages	2	4	8
No of clusters	20	49	53
No of grouped links	64	233	297
Accuracy (%)	99.05	93.85	94.35
Error (%)	3.5	6.14	5.6
Time taken	00:49	01:35	02:17

The table 4 consists of number of webpages considered, clusters formed, grouped links and its percentage of accuracy and error using fuzzy c means clustering.

Table 5: Evaluation for different number of webpages

Number of webpages	Precision (%)	Recall (%)	Accuracy (%)	Error (%)	Time taken(mm:ss)
2	99.05	96.36	96.42	3.50	00:49
4	95.53	98.16	93.85	6.14	01:35
8	96.68	97.27	94.35	5.60	02.17

The table 5 consists of number of webpages considered and its evaluation.

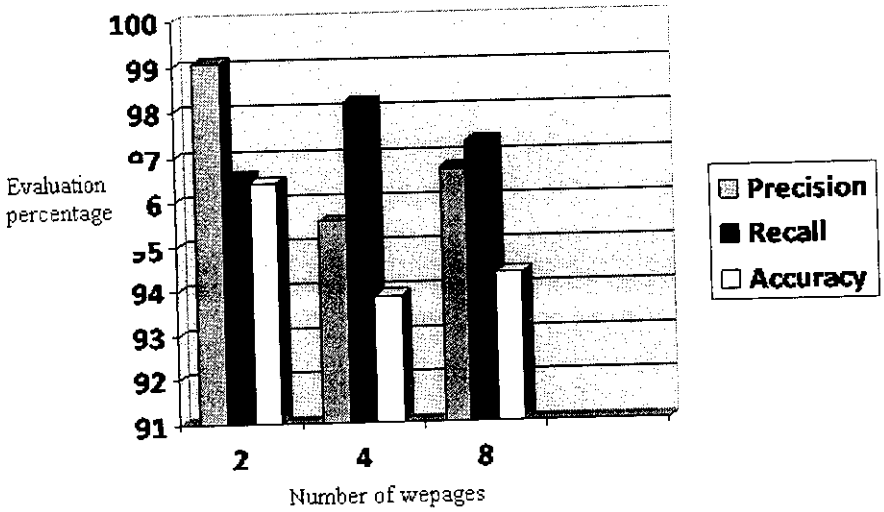
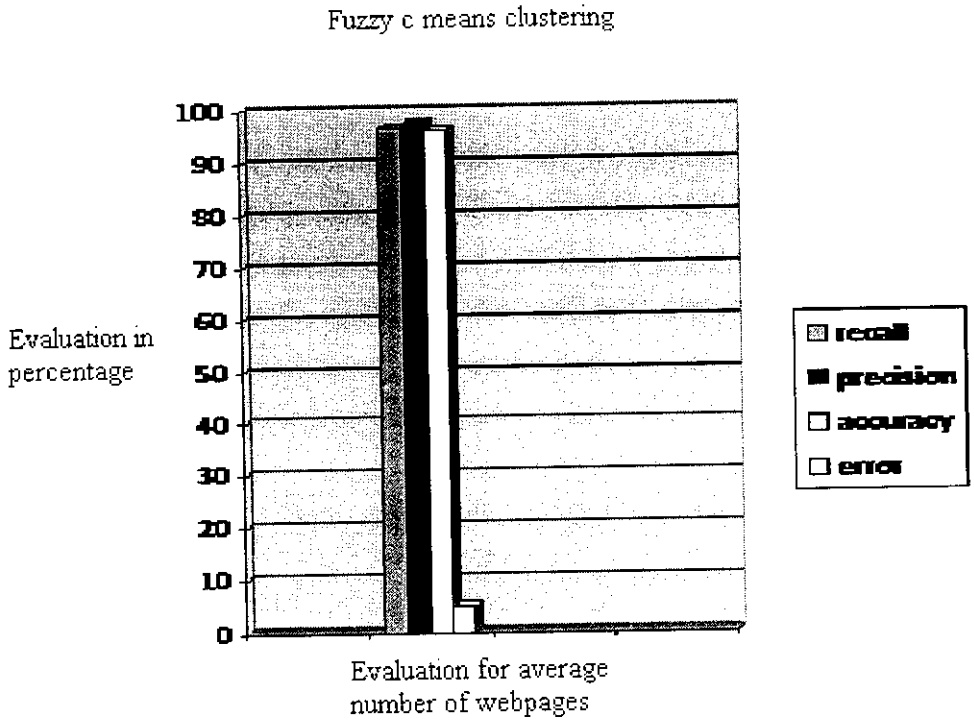


Table 6 : Statistics in Information Retrieval

Sl. no	Method	Recall	Precision	Accuracy	Error
1	Fuzzy C-means	96.26	97.40	96.06	5.08



SOFTWARE REQUIREMENTS

5. SOFTWARE REQUIREMENTS:

Language	:	c# .NET
Database	:	SQL Server-2008
Platform	:	Windows XP, Windows 7

5.1 C# Language:

C# is an elegant,simple,modern and type-safe object oriented language derived from C++ and Java that enables developers to build a wide range of secure and robust applications that run on the .NET Framework. C# used to create traditional Windows client applications, XML Web services, distributed components, client-server applications, database applications, and much more. Microsoft Visual C# 2010 provides an advanced code editor, convenient user interface designers, integrated debuggers and many other tools to facilitate rapid application development based on version 3.0 of the C# language and .NET Framework.

5.1.1 MAIN FEATURES OF C#

1. Pointers are missing in C#.
2. Unsafe operations such as direct memory manipulation are not allowed.
3. In C# there is no usage of "::" or "->" operators.
4. Since it's on .NET, it inherits the features of automatic memory management and garbage collection.
5. Varying ranges of the primitive types like Integer, Floats etc.
6. Integer values of 0 and 1 are no longer accepted as Boolean values. Boolean values are pure true or false values in C# so no more errors of "="operator and "=="operator.

7. "==" is used for comparison operation and "=" is used for assignment operation.

5.2 .NET PLATFORM:

C# programs run on .NET Framework, an integral component of Windows that includes a virtual execution system called the common language runtime(CLR) and a unified set of class libraries. The CLR is Microsoft's commercial implementation of the common language infrastructure (CLI), an international standard that id the basis for creating execution and development environments in which languages and libraries work together seamlessly.

Source code written in C# is compiled into an intermediate language (IL) that conforms to the CLI specification. The IL code,along with resources such as bitmaps and strings, is stored on disk in an executable file called an assembly, typically with an extension of .exe or .dll. an assembly contains a manifest that provides information in the assembly's types, version, culture and security requirements. When the C# program is executed, the assembly is loaded into the CLR,which might take various actions based on the information in the manifest.

Then if the security requirements are met, the CLR performs just in time (JIT) compilation to convert the IL code into native machine instructions. The CLR also provides other services related to automatic garbage collection, exception handling and resource management. Code that is executed by the CLR is sometimes referred to as "managed code", in contrast to "unmanaged code" which is compiled into native machine language that targets a specific system. Language interoperability is a key feature of the .NET Framework.

5.3 HARDWARE REQUIREMENTS:

Processor: Intel® Core(TM)2 CPU 1.87 GHz

RAM : 1 GB

Hard disk drive: 40 GB

CONCLUSION AND FUTURE ENHANCEMENT

6. CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

With the information overload, Web mining is a new and promising research issue to help users in gaining insight into overwhelming information on the Web. Web news provide a valuable resource for information, However, because each news page often contains news items of unrelated topics,web page-based clustering will seldom give useful results. In order to improve the results, we have in this project presented an approach based on extracting the links from the news web pages and mining separately. The removal of non-relevant information also makes news item extraction useful as a way of data cleaning, and the space needed for storing the links is much smaller.

6.2 FUTURE ENHANCEMENT:

Future work includes a more detailed study of data mining techniques applied on the news items as well as a closer integration of the techniques. .Our work supports and extends other work on using web structure to classify documents, and demonstrates the useful-ness of considering inbound links, and words surrounding them. The proposed system is not meant for optimization. It is applied in the urls to improve the performance of clustering.

APPENDICES

7. APPENDICES

7.1 SOURCE CODE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.IO;
using System.Xml.Linq;
namespace Optimizer.UI
{
   public partial class NewsUI : Form
   {
      public NewsUI()
      {
         InitializeComponent();
      }
      private void NewsUI_Load(object sender, EventArgs e)
      {
         this.Text = "Information Extarction Home";
         CatToLook = CatType.CatName;
```

```
        LinqDatas.NewsLinqDataContext MainUrl = new
Optimizer.LinqDatas.NewsLinqDataContext();
        var Datas = from k in MainUrl.News select new { Backlink =
k.backlink, Url = k.url, Category = k.catlevel, BaseUrl = k.baseurl };
        dataGridView1.DataSource = Datas.ToList();
    }
    delegate void SetTextCallback(string str1, string str2, bool t);
    delegate void SetDisplayCallback(IQueryable<string> t, string str2);

    string CatToLook = "";
    private void SetText(string str, string str1, bool t)
    {
      if (str1 == "p1")
      {
        if (this.label2.InvokeRequired)
        {
          SetTextCallback d = new SetTextCallback(SetText);
          this.Invoke(d, new object[] { str, str1, t });
        }
        else
        {
          label2.Text = str;
        }
      }
      else if (str1 == "p2")
      {
        if (this.label3.InvokeRequired)
```

```
    {
        SetTextCallback d = new SetTextCallback(SetText);
        this.Invoke(d, new object[] { str, str1, t });
    }
    else
    {
        label3.Text = str;
    }
}
else if (str1 == "p11")
{
    if ((this.label11.InvokeRequired) || (this.PP1.InvokeRequired))
    {
        SetTextCallback d = new SetTextCallback(SetText);
        this.Invoke(d, new object[] { str, str1, t });
    }
    else
    {
        label11.Text = str;
        if (t)
        {
            PP1.Visible = true;
        }
        else
        {
            PP1.Visible = false;
        }
```

```
        }
      }
    }

    private void button2_Click(object sender, EventArgs e)
    {
      SetText("Url Converstion Started", "p11", false);
      System.Threading.ThreadStart g = delegate {
CallProcessURl(CatToLook); };
      System.Threading.Thread t1 = new System.Threading.Thread(g);
      t1.Start();
    }
    public void CallProcessURl(string catid)
    {
      SetText("Url Processing Started", "p11", true);
      Optimizer.Analyzer.NewsAnalyzer G = new
Optimizer.Analyzer.NewsAnalyzer();
      SetText(G.ProcessUrl(CatToLook), "p11", false);
    }
    private void button3_Click(object sender, EventArgs e)
    {
6                    System.Threading.ThreadStart g = delegate {
CallProcessWord(CatToLook); };
      System.Threading.Thread t1 = new System.Threading.Thread(g);
      t1.Start();
    }
    private void CallProcessWord(string CatID)
```

```
    {
        SetText("Word Converstion Started", "p11", true);
        Optimizer.Analyzer.NewsAnalyzer G = new
Optimizer.Analyzer.NewsAnalyzer();
        SetText(G.ProcessWord(CatID), "p11", false);
    }
    private void button1_Click(object sender, EventArgs e)
    {
        timer1.Enabled = true;
        timer2.Enabled = true;
        //timer3.Enabled = true;
        if (checkBox1.Checked == true)
        {
            System.Threading.ThreadStart g = delegate {
StartupOptimize(CatToLook); };
            System.Threading.Thread t1 = new System.Threading.Thread(g);
            t1.Start();
        }       if (checkBox2.Checked == true)
        {
            System.Threading.ThreadStart g = delegate {
SecondOptimize(CatToLook); };
            System.Threading.Thread t1 = new System.Threading.Thread(g);
            t1.Start();
        }
        if (checkBox3.Checked == true)
        {
```

```
            System.Threading.ThreadStart g = delegate { BlogOptimize(CatToLook); };
            System.Threading.Thread t1 = new System.Threading.Thread(g);
            t1.Start();
        }
    }
    public void StartupOptimize(String Cat)
    {
      try
      {
        // XElement StartUrls = XElement.Load(@"D:\XML\CrawlingStartUp\NewsUrl.xml");
        // var Gourl = from TempUrl in StartUrls.Descendants("Opt") where TempUrl.Attribute("Visited").Value == "false" orderby TempUrl.Attribute("Pirority").Value descending select TempUrl.Attribute("Url").Value;
        int len = comboBox1.Items.Count;
        int len1 = len;
        foreach (String FinalUrl in comboBox1.Items)
        {
          string URL = FinalUrl;
          try
          {
            SetText("Currently Indexing : " + URL, "pl", false);
            HttpWebRequest req = (HttpWebRequest)WebRequest.Create(URL);
            HttpWebResponse resp;
```

```
                resp = (HttpWebResponse)req.GetResponse();
                SetText("Currently Reading : " + URL, "pl", false);
                StreamReader sr = new
StreamReader(resp.GetResponseStream());
                string pd = sr.ReadToEnd();
                sr.Close();
                SetText("Contents Readed : " + URL, "pl", false);
                Crawler.NewsProcess Process = new
Optimizer.Crawler.NewsProcess();
                SetText(Process.AddLink(pd, resp.ResponseUri), "p1",
false);
                Process.Parse(pd, FinalUrl, Cat);
                SetText("Page Indexed : " + URL, "pl", false);
                SetText("URL To LOOK : " + len + " - Optimized URL's : "
+ (len1 - 1).ToString(), "p2", false);
                len1 = len1 - 1;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
```

```
    }
    public void SecondOptimize(string Cat)
    {


        LinqDatas.NewsLinqDataContext MainUrl = new
Optimizer.LinqDatas.NewsLinqDataContext();
        IEnumerable<String> Gourl = from TempUrl in MainUrl.News
                                    where TempUrl.visited == "false" &&
TempUrl.pirority != "0" && TempUrl.pagetype == "WebPage"
                                    orderby TempUrl.pirority descending
                                    select TempUrl.url;
        int len = Gourl.ToArray().Length;
        SetText("URL To LOOK : " + len, "p4", false);
        int len1 = len;
        foreach (string FinalUrl in Gourl)
        {
          string URL = FinalUrl;
          try
          {
            len1 = len1 - 1;
            SetText("Currently Indexing : " + URL, "p3", false);
            HttpWebResponse resp;
            string pd = "";
            HttpWebRequest req =
(HttpWebRequest)WebRequest.Create(URL);
```

```
            resp = (HttpWebResponse)req.GetResponse();
SetText("Currently Reading : " + URL, "p3", false);
            StreamReader sr = new
StreamReader(resp.GetResponseStream());
            pd = sr.ReadToEnd();
            sr.Close();
            SetText("Contents Readed : " + URL, "p3", false);
            Crawler.NewsProcess Process = new
Optimizer.Crawler.NewsProcess();
            SetText(Process.AddLink(pd, resp.ResponseUri), "p1", false);
            Process.Parse(pd, FinalUrl, resp.LastModified.ToString());
            SetText("Page Indexed : " + URL, "p3", false);
            LinqDatas.NewsLinqDataContext Updatas = new
Optimizer.LinqDatas.NewsLinqDataContext();
            LinqDatas.New Upval = Updatas.News.Single(p => p.url ==
FinalUrl);
            Upval.datecrawl = DateTime.Now.ToString();
            Upval.visited = "true";
            Updatas.SubmitChanges();
            SetText("URL To LOOK : " + len + " - Optimized URL's : " +
(len1 - 1).ToString(), "p4", false);
        }
        catch (Exception ex)
        {
          // MessageBox.Show(ex.ToString());
        }
    }
```

```
    }       public void BlogOptimize(string Cat)
    {
      LinqDatas.NewsLinqDataContext MainUrl = new
Optimizer.LinqDatas.NewsLinqDataContext();
      IEnumerable<String> Gourl = from TempUrl in MainUrl.News
                  where TempUrl.visited == "false" &&
TempUrl.pirority != "0" && TempUrl.pagetype == "Blog"
                  orderby TempUrl.pirority descending
                  select TempUrl.url;
      int len = Gourl.ToArray().Length;
      SetText("URL To LOOK : " + len, "b1", false);
      int len1 = len;
      foreach (string FinalUrl in Gourl)
      {
        string URL = FinalUrl;
        try
        {
          len1 = len1 - 1;
          SetText("Currently Indexing : " + URL, "b2", false);
          HttpWebResponse resp;
          string pd = "";
          HttpWebRequest req =
(HttpWebRequest)WebRequest.Create(URL);
          resp = (HttpWebResponse)req.GetResponse();
SetText("Currently Reading : " + URL, "b2", false);
          StreamReader sr = new
StreamReader(resp.GetResponseStream());
```

```
                pd = sr.ReadToEnd();
                sr.Close();
                SetText("Contents Readed : " + URL, "b2", false);
                Crawler.NewsProcess Process = new
Optimizer.Crawler.NewsProcess();
                SetText(Process.AddLink(pd, resp.ResponseUri), "b2", false);
                Process.Parse(pd, FinalUrl, resp.LastModified.ToString());
                SetText("Page Indexed : " + URL, "b2", false);
                LinqDatas.NewsLinqDataContext Updatas = new
Optimizer.LinqDatas.NewsLinqDataContext();
                LinqDatas.New Upval = Updatas.News.Single(p => p.url ==
FinalUrl);
                Upval.datecrawl = DateTime.Now.ToString();
                Upval.visited = "true";
                Updatas.SubmitChanges();
                SetText("URL To LOOK : " + len + " - Optimized URL's : " +
(len1 - 1).ToString(), "b1", false);
            }
            catch (Exception ex)
            {
                // MessageBox.Show(ex.ToString());
            }
        }
    }
    int res = 0;
    private void button7_Click(object sender, EventArgs e)
    {
```

```
        //this.Close();
        //Application.ExitThread();
        if (rlsts.Count > 0)
        {
            var Rd = from k in rlsts group k by k.Category into g select new { key = g.Key, datas = from i in g select new { Backlink = i.BackLink, Url = i.Url } };
            foreach (var i in Rd)
            {
                try
                {
                    treeView1.Nodes.Add(i.key, i.key);
                    res++;
                    foreach (var it in i.datas)
                    {
                        treeView1.Nodes[i.key].Nodes.Add(it.Backlink + " , " + it.Url);
                    }
                }
                catch
                {
                }
            }
        }
        else
        {
            MessageBox.Show("Calculate Tdidf Frequency");
```

```
        }
        label5.Text = "Clusters : " + res.ToString();
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
        progressBar1.Value = 0;
    }
    private void timer2_Tick(object sender, EventArgs e)
    {
        progressBar1.Value += 5;
        if (progressBar1.Value >= 300)
        {
            progressBar1.Value = 0;
        }
    }
    private void button8_Click(object sender, EventArgs e)
    {
        System.Threading.ThreadStart g = delegate { StripStopUrl(); };
        System.Threading.Thread t1 = new System.Threading.Thread(g);
        t1.Start();
    }
    public void StripStopUrl()
    {
        SetText("Stripping Stopurl Started", "p11", true);
        LinqDatas.NewsLinqDataContext MainData = new
Optimizer.LinqDatas.NewsLinqDataContext();
        var qry = from i in MainData.News select i;
```

```
        foreach (LinqDatas.New g in qry)
        {
            Striper H = new Striper();
            string str = H.StripStopWord(g.url, "News");
        }
        SetText("Stripping Stopurl Ended", "p11", false);
    }
        }
    private void DragEnterStop(object sender, DragEventArgs e)
    {
        e.Effect = DragDropEffects.Copy;
    }
    private void button6_Click(object sender, EventArgs e)
    {
        Optimizer.LinqDatas.NewsLinqDataContext N = new
LinqDatas.NewsLinqDataContext();
        var Ns = from k in N.News select k;
        XElement XX =
XElement.Load(@"D:\webstavis\XCralwerPackedData\XCrawlerData.xml"
);
        foreach (Optimizer.LinqDatas.New s in Ns)
        {
            try
            {
                XElement Data = new XElement("Data",
                    new XAttribute("Url", s.url),
                                new XAttribute("BackLink", s.backlink),
```

```
                    new XAttribute("Category", s.catlevel),
                new XAttribute("Date", s.datecrawl));
            XX.Add(Data);

        }
        catch (Exception ex)
        {
        }
    }
    X.Save(@"D:\webstavis\XCralwerPackedData\XCrawlerData.xml");
    MessageBox.Show("Datas Packed !!");
}
private void button2_Click_1(object sender, EventArgs e)
{
    comboBox1.Items.Add(comboBox1.Text);
}
private void button3_Click_1(object sender, EventArgs e)
{
    LinqDatas.NewsLinqDataContext MainUrl = new
Optimizer.LinqDatas.NewsLinqDataContext();
    var todel = from k in MainUrl.News select k;
    MainUrl.News.DeleteAllOnSubmit(todel);
    MainUrl.SubmitChanges();
    MessageBox.Show("Deleted...");
}
List<WebData> lsts = new List<WebData>();
List<WebData> rlsts = new List<WebData>();
```

```
int ud = 0;
private void button4_Click_1(object sender, EventArgs e)
{
  XElement XRead =
XElement.Load(@"D:\webstavis\XCralwerPackedData\XCrawlerData.xml"
);
  var Rd = from k in XRead.Elements("Data") select new { Backlink =
k.Attribute("BackLink").Value, Cat = k.Attribute("Category").Value, Url =
k.Attribute("Url").Value };
  foreach (var i in Rd)
  {
    WebData w = new WebData();
    w.BackLink = i.Backlink;
    w.Category = i.Cat;
    w.Url = i.Url;
    lsts.Add(w);
  }
  string[] s = new string[lsts.Count];
  for (int i = 0; i <= lsts.Count - 1; i++)
  {
    s[i] = lsts[i].BackLink.ToString();
  }
  TFIDFMeasure Tdf = new TFIDFMeasure(s);
  for (int i = 0; i <= lsts.Count - 1; i++)
  {
    WebData w = new WebData();
    w.BackLink = lsts[i].BackLink;
```

```
            w.Category = lsts[i].Category;
            w.Url = lsts[i].Url;
            w.TdidfFreq = Tdf.GetInverseDocumentFrequency(i).ToString();
            rlsts.Add(w);
        }
        if (rlsts.Count > 0)
        {
            var Rdx = from k in rlsts group k by k.TdidfFreq  into g select new
{ key = g.Key, datas = from i in g select new { Backlink = i.BackLink, Url =
i.Url } };
            foreach (var i in Rdx)
            {
              try
              {
                treeView2.Nodes.Add(i.key, i.key);
                ud++;
                foreach (var it in i.datas)
                {
                  treeView2.Nodes[i.key].Nodes.Add(it.Backlink + " , " +
it.Url);
                }
              }
              catch
              {
              }
            }
        }
```

```
            else
            {
                MessageBox.Show("Calculate Tdidf Frequency");
            }
            dataGridView2.DataSource = rlsts.ToList();
            label6.Text = "Content Clusters: " + ud.ToString();
        }
    }
}
```

TF-IDF

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using System.Text.RegularExpressions;
namespace Optimizer
{
    public class TFIDFMeasure
    {
        private string[] _docs;
        private string[][] _ngramDoc;
        private int _numDocs = 0;
        private int _numTerms = 0;
        private ArrayList _terms;
        private int[][] _termFreq;
```

```
private float[][] _termWeight;
private int[] _maxTermFreq;
private int[] _docFreq;
  private void MyInit()
{
   _terms = GenerateTerms(_docs);
   _numTerms = _terms.Count;
   _maxTermFreq = new int[_numDocs];
   _docFreq = new int[_numTerms];
   _termFreq = new int[_numTerms][];
   _termWeight = new float[_numTerms][];

   for (int i = 0; i < _terms.Count; i++)
   {
      _termWeight[i] = new float[_numDocs];
      _termFreq[i] = new int[_numDocs];
      AddElement(_wordsIndex, _terms[i], i);
   }
   GenerateTermFrequency();
   GenerateTermWeight();
}
 private void GenerateTermFrequency()
{
   for (int i = 0; i < _numDocs; i++)
   {
      string curDoc = _docs[i];
      IDictionary freq = GetWordFrequency(curDoc);
```

```
            IDictionaryEnumerator enums = freq.GetEnumerator();
            _maxTermFreq[i] = int.MinValue;
            while (enums.MoveNext())
            {
               string word = (string)enums.Key;
               int wordFreq = (int)enums.Value;
               int termIndex = GetTermIndex(word);

               _termFreq[termIndex][i] = wordFreq;
               _docFreq[termIndex]++;
               if (wordFreq > _maxTermFreq[i]) _maxTermFreq[i] =
wordFreq;
            }
         }
      }
      private void GenerateTermWeight()
      {
         for (int i = 0; i < _numTerms; i++)
         {
            for (int j = 0; j < _numDocs; j++)
               _termWeight[i][j] = ComputeTermWeight(i, j);
         }
      }
      private float GetTermFrequency(int term, int doc)
      {
         int freq = _termFreq[term][doc];
         int maxfreq = _maxTermFreq[doc];
```

```
    return ((float)freq / (float)maxfreq);
}
  public float GetInverseDocumentFrequency(int term)
{
    int df = _docFreq[term];
    return Log((float)(_numDocs) / (float)df);
}
        public float ComputeTermWeight(int term, int doc)
{
    float tf = GetTermFrequency(term, doc);
    float idf = GetInverseDocumentFrequency(term);
    return tf * idf;
}
  public IDictionary GetWordFrequency(string input)
{
    string convertedInput = input.ToLower();
    Tokeniser tokenizer = new Tokeniser();
    String[] words = tokenizer.Partition(convertedInput);
    Array.Sort(words);
    String[] distinctWords = GetDistinctWords(words);
    IDictionary result = new Hashtable();
    for (int i = 0; i < distinctWords.Length; i++)
    {
       object tmp;
       tmp = CountWords(distinctWords[i], words);
       result[distinctWords[i]] = tmp;
    }
```

```
        return result;
    }
```

FUZZY C-MEANS CLUSTERING

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Optimizer
{
   public sealed class CMeansAlgorithm
   {
      private double Accuracy = Math.Pow(10, -5);
      private List<ClusterPoint> Points;
      private List<ClusterPoint> Clusters;
      public double[,] U;
      private double Fuzzyness;
        private double Eps = Math.Pow(10, -5);
      private double J { get; set; }
      public string Log { get; set; }
      public CMeansAlgorithm(List<ClusterPoint> points,
 List<ClusterPoint> clusters): this(points, clusters, 2.0)
      {
      }
       public CMeansAlgorithm(List<ClusterPoint> points,
 List<ClusterPoint> clusters, double fuzzy)
```

```
{
    if (points == null)
    {
        throw new ArgumentNullException("points");
    }
    if (clusters == null)
    {
        throw new ArgumentNullException("clusters");
    }
    this.Points = points;
    this.Clusters = clusters;
    U = new double[this.Points.Count, this.Clusters.Count];
    this.Fuzzyness = fuzzy;
    double diff;
    for (int i = 0; i < this.Points.Count; i++)
    {
        ClusterPoint p = this.Points[i];
        double sum = 0.0;

        for (int j = 0; j < this.Clusters.Count; j++)
        {
            ClusterPoint c = this.Clusters[j];
            diff = this.CalculateEulerDistance(p, c);
            U[i, j] = (diff == 0) ? Eps : diff;
            sum += U[i, j];
        }
        double sum2 = 0.0;
```

```
        for (int j = 0; j < this.Clusters.Count; j++)
        {
            U[i, j] = 1.0 / Math.Pow(U[i, j] / sum, 2.0 / (Fuzzyness - 1.0));
            sum2 += U[i, j];
        }
        for (int j = 0; j < this.Clusters.Count; j++)
        {
            U[i, j] = U[i, j] / sum2;
        }
    }
    this.RecalculateClusterIndexes();
}
private CMeansAlgorithm()
{
}
private void RecalculateClusterIndexes()
{
    for (int i = 0; i < this.Points.Count; i++)
    {
        double max = -1.0;
        var p = this.Points[i];

        for (int j = 0; j < this.Clusters.Count; j++)
        {
            if (max < U[i, j])
            {
                max = U[i, j];
```

```
                p.ClusterIndex = (max == 0.5) ? 0.5 : j;
            }
        }
    }
}
public void Step()
{
    for (int c = 0; c < Clusters.Count; c++)
    {
        for (int h = 0; h < Points.Count; h++)
        {
            double top = CalculateEulerDistance(Points[h], Clusters[c]);
            if (top < 1.0) top = Eps;
            double sumTerms = 0.0;
            for (int ck = 0; ck < Clusters.Count; ck++)
            {
                double thisDistance = CalculateEulerDistance(Points[h],
Clusters[ck]);
                if (thisDistance < 1.0) thisDistance = Eps;
                sumTerms += Math.Pow(top / thisDistance, 2.0 /
(this.Fuzzyness - 1.0));
            }
            U[h, c] = (double)(1.0 / sumTerms);
        }
    }
    this.RecalculateClusterIndexes();
}
```

```
    private double CalculateEulerDistance(ClusterPoint point, ClusterPoint
centroid)
    {
      //return Math.Sqrt(Math.Pow(p.X - c.X, 2) + Math.Pow(p.Y - c.Y,
2));
       double sum = 0.0;

       for (int i = 0; i < point.Dimention; i++)
       {
          sum += Math.Pow(point.Coords[i] - centroid.Coords[i], 2);
       }
       return Math.Sqrt(sum);
     }
     private double CalculateObjectiveFunction()
     {
        double Jk = 0;
        for (int i = 0; i < this.Points.Count; i++)
        {
           for (int j = 0; j < this.Clusters.Count; j++)
           {
              Jk      +=      Math.Pow(U[i,      j],      this.Fuzzyness)      *
Math.Pow(this.CalculateEulerDistance(Points[i], Clusters[j]), 2);
           }
        }
        return Jk;
     }
     private void CalculateClusterCenters()
```

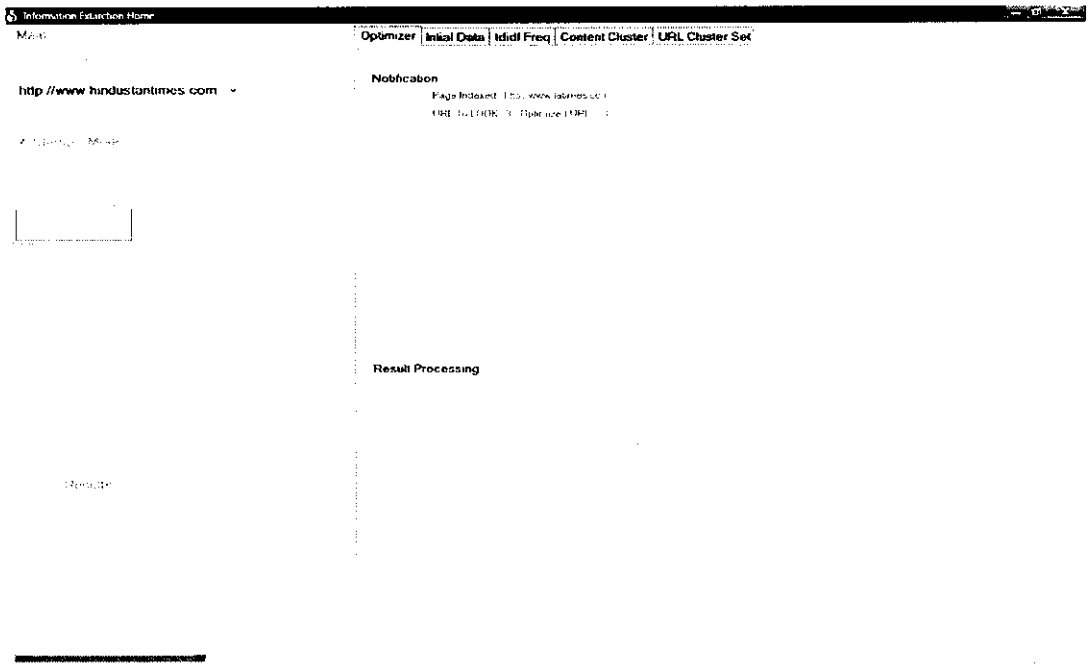
```
{
    for (int j = 0; j < this.Clusters.Count; j++)
    {
        ClusterPoint c = this.Clusters[j];
        //double uX = 0.0;
        //double uY = 0.0;
        double[] uC = new double[c.Dimention];
        double l = 0.0;
        for (int i = 0; i < this.Points.Count; i++)
        {
            ClusterPoint p = this.Points[i];

            double uu = Math.Pow(U[i, j], this.Fuzzyness);
            for (int k = 0; k < c.Dimention; k++)
            {
                uC[k] += uu * c.Coords[k];
            }
            //uX += uu * c.X;
            //uY += uu * c.Y;
            l += uu;
        }
        for (int k = 0; k < c.Dimention; k++)
        {
            c.Coords[k] = ((int)(uC[k] / l));
            //c.X = ((int)(uX / l));
            //c.Y = ((int)(uY / l));
        }
```

```
            this.Log += string.Format("Cluster Centroid: ({0}; {1})" +
System.Environment.NewLine, c.Coords[0], c.Coords[1]);
        }
    }
    public int Run()
    {
        return this.Run(this.Accuracy);
    }
    public int Run(double accuracy)
    {
        int i = 0;
        int maxIterations = 20;
        do
        {
            i++;
            this.J = this.CalculateObjectiveFunction();
            this.CalculateClusterCenters();
            this.Step();
            double Jnew = this.CalculateObjectiveFunction();
            if (Math.Abs(this.J - Jnew) < accuracy) break;
        }
        while (maxIterations > i);
        return i;
    }
}
public class ClusterPoint
{
```

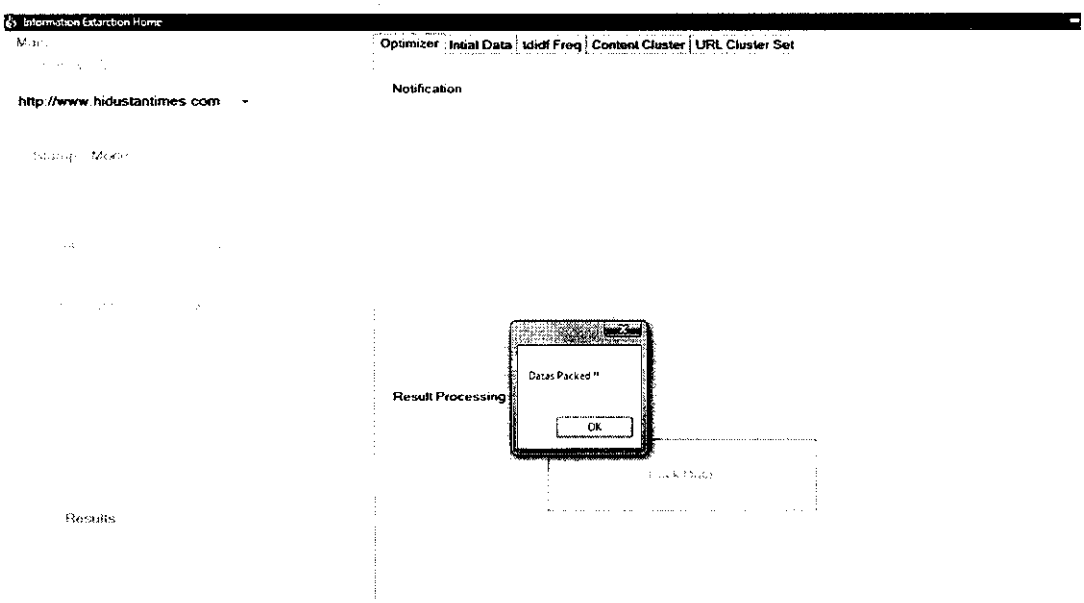
```
    public double X { get { return this.Coords[0]; } }
    public double Y { get { return this.Coords[1]; } }
    public object Tag { get; set; }
    public double ClusterIndex { get; set; }
    public List<double> Coords { get; set; }
    public int Dimention { get { return this.Coords.Count; } }
    public ClusterPoint(List<double> coords)
        : this(coords, null)
    public ClusterPoint(List<double> coords, object tag)
    {
        this.Coords = coords;
        this.Tag = tag;
        this.ClusterIndex = -1;
    }
    public ClusterPoint(double x, double y)
        : this(x, y, null)
    {
    }
    public ClusterPoint(double x, double y, object tag)
    {
        this.Coords = new List<double>();
        this.Coords.Add(x);
        this.Coords.Add(y);
        this.Tag = tag;
        this.ClusterIndex = -1;
    }
}
```

7.2. SCREENSHOTS

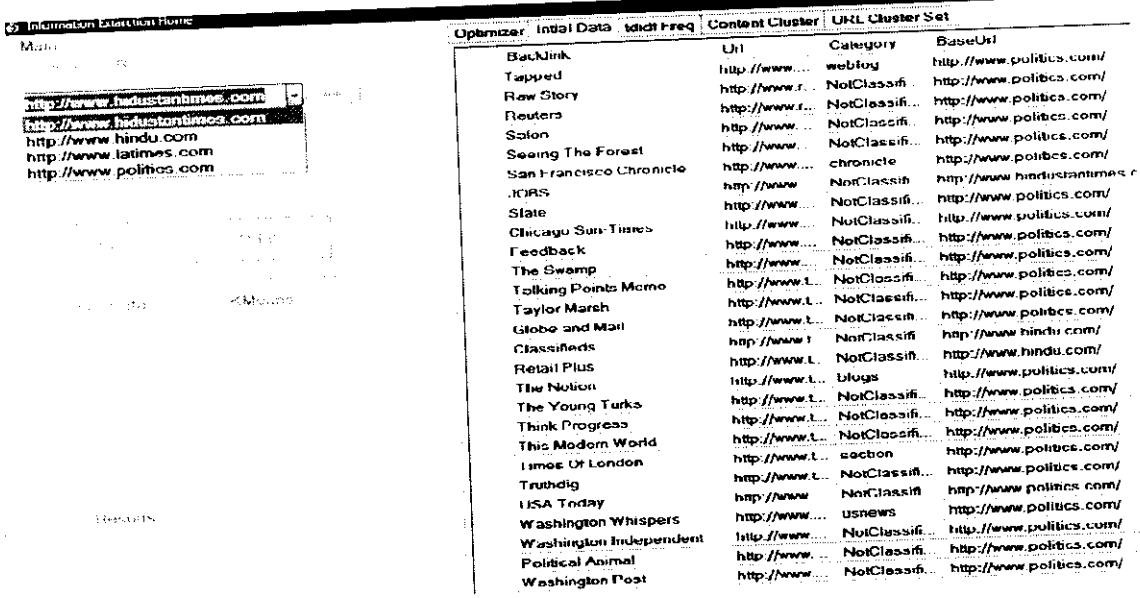
7.2.1 PAGE INDEXING



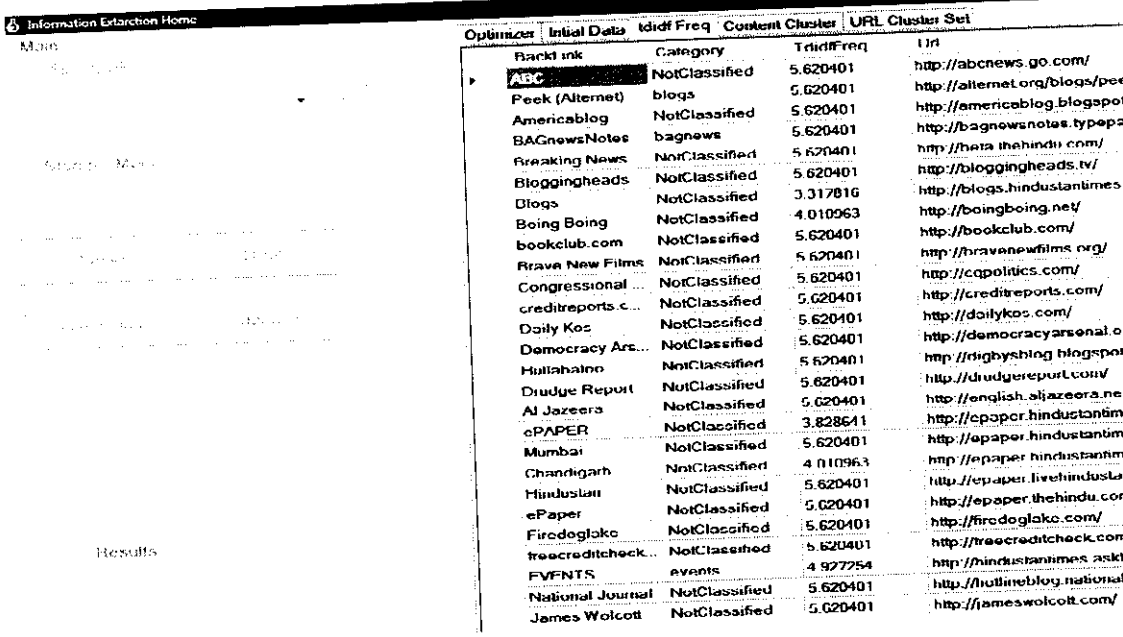
7.2.2 PACKING OF DATA



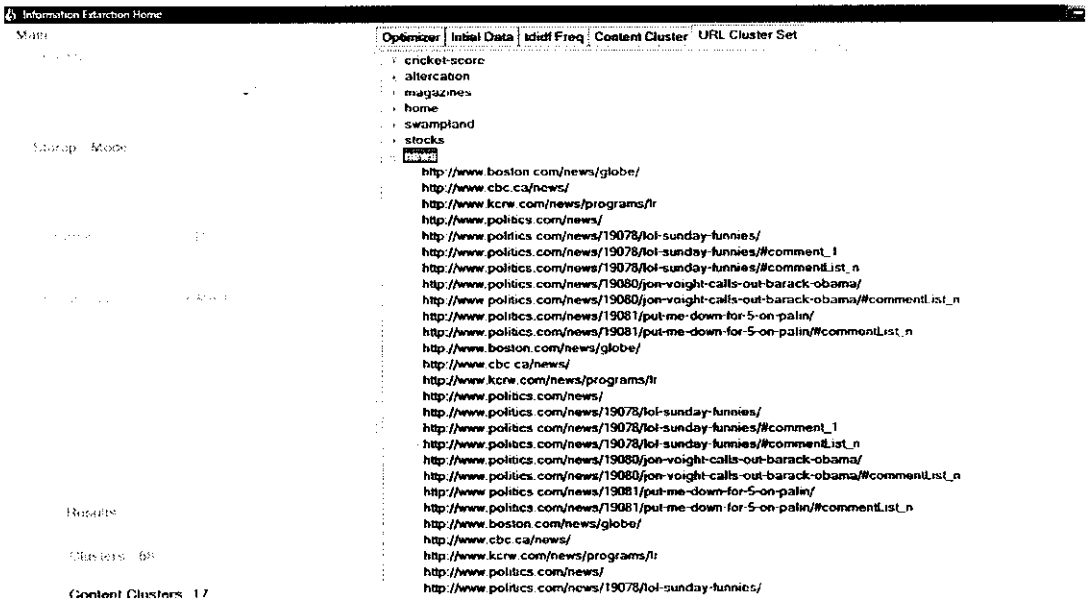
7.2.3 INITIAL DATA



7.2.4 TF-IDF CALCULATION BASED ON CATEGORY



7.2.5 CLUSTERING OF URL



7.2.6 LINKS STORAGE

REFERENCES

8. REFERENCES

1. Suhit Gupta, Gail Kaiser, "DOM based Content Extraction of HTML Document"2006.

2. Nikolaos K.Papadakis , Dimitrious Skoutas, "A System for Information Extraction Using Clustering Techniques" IEEE Transaction 2005

3.Deepak.P, Deepak Khemani, " UnURL :Unsupervised Learning from URL's,COMAD 2006 .

4.Shankar K Paul, Fellow Varun Talwar, " Webmining in Soft Computing Framework", IEEE Transaction 2002.

5.Raghu Krishnapuram, "Low Complexity Fuzzy Relational Clustering for webmining",IEEE Transaction 2001.

6.www.google.com