San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

Fall 2011

URL Recommender using Parallel Processing

Ravi Kishore Penta
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the Computer Sciences Commons

Recommended Citation
Penta, Ravi Kishore, "URL Recommender using Parallel Processing" (2011). *Master's Projects*. 200.
https://scholarworks.sjsu.edu/etd_projects/200

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

URL Recommender using Parallel Processing

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Ravi Kishore Penta

December 2011

© 2011

Ravi Kishore Penta

ALL RIGHTS RESERVED

SAN JOSÉ STATE UNIVERSITY

The Undersigned Writing Project Committee Approves the Project Titled

URL Recommender using Parallel Processing

by

Ravi Kishore Penta

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Soon Tee Teoh, Department of Computer Science	12/02/2011
Dr. Chris Tseng, Department of Computer Science	12/02/2011
Mr. Kiran Yellanki, Department of Computer Science	12/02/2011

Abstract

The main purpose of this project is to section similar news and articles from a vast variety of news articles. Let's say, you want to read about latest news related to particular topic like sports. Usually, user goes to a particular website and goes through some news but he won't be able to cover all the news coverage in a single website. So, he would be going through some other news website to checking it out and this continues. Also, some news websites might be containing some old news and the user might be going through that. To solve this, I have developed a web application where in user can see all the latest news from different websites in a single place. Users are given choice to select the news websites from which they want to view the latest news. The articles which we get from news websites are very random and we will be applying the DBSCAN algorithm and place the news articles in the cluster form for each specific topic for user to view. If the user wants to see sports, he will be provided with sports news section. And this process of extracting random news articles and forming news clusters are done at run time and at all times we will get the latest news as we will be extracting the data from web at run time. This is an effective way to watch all news at single place. And in turn this can be used as articles (URL) recommender as the user has to just go through the specific cluster which interests him and not visit all news websites to find articles. This way the user does not have to visit different sites to view all latest news. This idea can be expanded to not just news articles but also in other areas like collecting statistics of financial information etc. As the processing is done at runtime, the performance has to be improved. To improve the performance, the distributed data mining is used and multiple servers are being used which communicate with each other.

Table of Contents

1. Introduction 7

2. How Application Works 11

3. Tools 13

3.1 Eclipse 13

3.2 Tomcat Server 13

3.3 Java 14

4. Concepts 15

4.1 Data Mining 15

4.1.1 Process 15

4.1.2 Pre-processing: 15

4.1.3 Data mining 15

4.1.4 Results validation 16

4.2 Text Mining 16

4.3 Approaches to Text Mining 16

4.4 Considerations for "Numericizing" Text 17

5. Architecture 19

5.1 Sequential Approach 19

5.2 Parallel approach 20

5.3 Architecture of Traditional Data Mining 21

5.4 Architecture of Distributed Data Mining 22

6. Preliminary Work 24

6.1 Making Java Sockets to work 24

6.2 Reading the data from User (web interface) and displaying the response to web interface 25

6.2.1 To Read Data in Java Servlet *26*

6.2.2 To Display data on web interface *26*

6.3 Extraction of web pages based on user input 26

6.4 Parsing the downloaded web pages to get more URL's relevant to user input 27

7. Implementation 29

7.1 Natural Language Processing 29

7.1.1 Parsing for paragraph data *29*

7.1.2 Tokenizing the web page words *30*

7.1.3 Removing stop words *30*

7.1.4 Stemming the words *30*

7.1.5 Find the inverted index *31*

7.2 DBSCAN clustering algorithm 32

7.2.1 Algorithmic steps for DBScan Clustering *34*

7.2.2 Advantages *36*

7.2.3 Disadvantages *36*

7.3 Naming of Clusters 36

8. Results 38

9. Analysis 41

10. Conclusion 42

11. References 43

List of Figures

Figure 1: Architecture of Sequential Processing 19

Figure 2: Architecture of Parallel Processing 21

Figure 3: Detailed Architecture of Parallel Processing..... 21

Figure 4: Architecture of Distributed Parallel Processing 23

Figure 5: Screen Shot of Home Page of the URL Recommender..... 28

Figure 6: Screen Shot of List of URL's extracted at Run time for the given Keyword..... 28

Figure 7: Screen Shot of Clusters formed for extracted articles..... 37

List of Tables

Table 1 : Graph showing the time comparisons of sequential processing and parallel processing through servers .....
38

Table 2: Graph showing the time comparisons of sequential processing and parallel processing through servers and multithreading within..... 39

Table 3: Graph showing the time comparisons of sequential processing, parallel processing through servers and parallel processing through servers and multithreading(all together). 40

1. Introduction

Now Internet has tons and tons of websites on almost every topic and some of the services on world wide web (www) like google, yahoo, wikipedia, blogs, communities, forums, etc., provides intelligence to narrow down the relevant content for the topic. But gathering relevant information from these varied sources is time consuming and may need very high processing power. Even though the news websites has many articles, they won't be able to cover about all the latest news accurately. Some websites might be focusing on sports and some on politics etc. So, my web application gathers articles from multiple sources and clusters them into their proper section which in turn can be used as Article (URL) recommender. If the user is interested in a particular section of news, he can only view that section and leave out all other sections alone. So in a way it acts as an Article (URL) recommender. And there is no Article (URL) recommender which tells the user if he likes a particular article, then he is recommended to view the other limited list of articles related to that topic. In my project, I have implemented a web based article (URL) recommender which gives recommendations based on the specific topic he visits.

The main advantages of article (URL) recommender compared to several other search engines are

1. User has to go through a small list of web articles and sites to find the cluster of web articles which interests him.
2. Users don't have to skim through a variety of websites to find all the latest news which interests him.
3. Users can find the data which interests him quickly compared to search engine results.

An example of a web application which provides recommendations based on the content, the user goes through is Amazon. In Amazon, if User goes through or buys a product in his site, then recommendations are presented to user relevant to that product he went through or bought. And these recommendations are presented only for that session (till the website was open) unless, the user is signed. However, all the recommendations are confined to internal products and not outside their website.

My web application is similar to Amazon's recommendations but it is applied on the World Wide Web (internet). So, my web application gets results from all over the internet and makes recommendations. This application uses the major news websites like Fox news, CNN etc to extract the websites and content of those sites. So, the application is a level up on the search engines. Recommendations are made using a popular data mining algorithm called "DBSCAN Clustering Algorithm".

This web application is developed using Java, Java sockets, Java Servlets and Java Server Pages (JSP). To improve the time efficiency, application is implemented in several levels of multi-threading. This application provides a web based interface where users can chose the websites from which they need to see the news in a clustered manner. Where in, websites (articles) in each cluster are relevant to each other and if the user likes one of the website, he is recommended to visit other web pages in the same cluster.

As the lists of web articles of a particular topic are huge, there should be a mechanism which efficiently gathers links and does the recommendations. To do that, the web applications get the list of website from multiple servers where in, each server is gets web pages from a particular

source like CNN and not from any other source. Once the lists of web pages are extracted, a data mining algorithm is implemented locally on that particular source and shown to the user. If user likes to have combined set of result from all the sources, and then distributed data mining is applied to combine all the sources and show the list of websites in clusters. Also, within each source, each web page is extracted parallel using multi-threading concept of java. This application uses multi level parallelism, by using different servers and using multi threading.

Also, my web application shows the amount of time it took to extract and the amount to time it took to mine the data and show the recommendations. At the end of the report, results of how parallelized dataset generator and data mining performs better than a single machine approach are presented.

2. How Application Works

The home page of the web application consists of set of websites which the user can choose from. The web news articles are extracted from the websites which user chooses. The website provides are very popular and contains good web articles. As said before, they may not have all the web news. Once the user chooses the websites, the web articles are extracted from those selected sources at runtime. As we are extracting at runtime, to improve efficiency and performance, we are using multiple servers and multi threading concepts. The extracted web articles are random and not properly sectioned. Once the web articles are extracted, the clustering algorithm called DBSCAN algorithm is applied on the extracted to section it proper. DBSCAN algorithm is density based algorithm and it doesn't have the restriction of specifying the number of clusters to be formed beforehand and this is the reason, this algorithm best suit our requirement. This clustering process includes preprocessing and application of algorithm. Preprocessing involves in stemming, tokenizing, removing stop words and find inverted index. After clustering process, we get a set of clusters, we need to name the cluster so that the user knows what cluster contains and what kind of articles each cluster has. To specify name for each cluster, I have taken the keywords from URL, title and gave them the highest frequencies within that document, this process is done while preprocessing and not after clustering process is done. Then we find the inverted index, the inverted index actually contains few keywords which represent each document and as we gave highest frequency for title and URL keywords, they will be included in inverted index. So, after the clustering process is done, we got a set of clusters. Now, I took the highest frequency word from all the documents with in the cluster and kept it as the name for that cluster. As URL and title has the highest frequency for any document, the word chosen will be a URL keyword or title keyword from any one of the documents within the cluster. The same process is done for other clusters too. That's it; we get a set of sectioned web

articles with a specific name for the cluster. And results turnout to be pretty good as there was no restriction of number of clusters to be formed.

3. Tools

3.1 Eclipse

Eclipse IDE is desktop IDE (Integrated Development Environment) used to develop software applications (including web applications). This IDE has different version for different kind of purposes like Eclipse IDE for Java, Eclipse IDE for J2EE. The IDE supports full life cycle of development of software. Eclipse IDE support different languages like JAVA, XML, Java Servlets, Java Server Pages. The IDE also supports deployment, testing and to test portability of software. It is supported in Linux, Mac OSx and Windows environments.

Main Features of Eclipse include:

1. IDE for Java Developers provides auto and incremental compilation and has the support of code assist which makes the life of developer easy.
2. IDE also supports cross-referencing and superior editing with validations on it.
3. It also has the support of Data base connectivity and several plug-in can be used too.
4. It has inbuilt Java environment and development kit which helps the developer to start coding right away with any cumbersome settings.
5. It also has the support of web server like Apache Tomcat which allows the developer to deploy web applications.

3.2 Tomcat Server

Tomcat Server is a web server used to deploy the developed web application. Because of its easy of installing and set up, it has become the industry standard for web application. It is a java based container used to run java servlets and java server pages on the web. It can also handle the many number of requests based on the set up of thread pool. So, it is basically used to server web pages on the web, both static and dynamic pages.

User makes a request on web browser and the request is sent to web server. This web server processes the request and fetches the required information/page, then sends it back to web browser. Also, Tomcat is reliable, free and easy to configure.

3.3 Java

Java is programming language used to develop the server side implementation in my web application (Article recommender). As java is portable and platform independent (write once and run anywhere), it has become one of the most popular programming language. There is a wide variety of Java based languages like Java Server Pages, Java Servlets and different API's being developed and supported in Java. Now days, Java has become the most used software and web application development environments.

In my web application, I have mainly used the following

1. Java Server Pages: These are used to dynamically generate web pages and it's based on java technology. This is also termed as JSP and it is a upper level abstraction of Java Servlets. JSP are compiled to Java Servlets using JSP compiler
2. Java Servlets: Java Servlets are java classes used to fetch response to the request sent from the web browser. These are also used to read request and display web content.
3. Java Sockets: Java sockets used to connect from one system/server to other using sockets. Socket acts an end point in a two way communication link between two systems running on the network. Every socket has to be bound to port number so that, the data send through it (TCP layer) can be identified by the application destined.

4. Concepts

4.1 Data Mining

The process of Data mining involves finding patterns from huge data sets by applying different of statistical algorithms. As a result of data mining, we get meaningful patterns from unstructured huge data sets. Hence, it is given the name of Knowledge Discovery.

4.1.1 Process

The process of data mining mainly involves three steps. Firstly, preprocessing step takes place, where we actually do natural language processing and then comes the application of data mining algorithm. Finally, we do validation of results.

4.1.2 Pre-processing:

In this stage, natural language processing is done. This will be discussed in detail in later sections.

4.1.3 Data mining

There are four types of data mining algorithms applied on data sets. The following are the four types.

1. Clustering
2. Association rules
3. Regression
4. Classification.

Out of all these four types, we are only interested in clustering in our application. In clustering, we group similar kind of data (documents together) in to separate cluster. And this clustering is done on unstructured data.

4.1.4 Results validation

In this phase, the patterns which are found from previous step are validated. Patterns which are found as results of data mining need not be valid all the time. At times, the patterns found will not present at all in the data set. It condition said to be over fitting. This problem can be solved by finding the good patterns from data set, then take raw untrained data set and apply these patterns on this untrained dataset. As a result of this, we get a new set of patterns, if patterns found are the same as desired one. Then the results are legit else it's not good.

4.2 Text Mining

Text mining is the process of converting textual content into numerical data. The textual data provided is unstructured. Once the text is converted to numerical data, the numerical data will be useful for applying different data mining algorithm. As a result of application of data mining algorithm, we can find either different clusters of similar documents or find summary of the documents based on the term frequency. Mainly, the clustering algorithm is applied on unstructured data and it is considered as unsupervised learning method.

4.3 Approaches to Text Mining

The process of converting textual data into numerical data is done by finding the term frequencies of each word, removing stop words and stemming words. Term frequency of count of each word, how many times it appears in that document. Stop word are the words which are very trivial for the meaning of sentences. Some examples of stop words are "a", "an", "the" etc. So, these kinds of words are removed. Stemming is process of converted a word to its root form. For example, if the word under consideration is "playing". The stemmed form of this word would be "play". Even 'played', 'plays' etc will be stemmed to "play" as it is root form of that word. This is done before find term frequency, so that we get the accurate term frequencies of each unique word.

4.4 Considerations for "Numericizing" Text

Stop-words:- Stop word are the words which are very trivial for the meaning of the sentence and hence, it is okay to exclude these from consideration. Few stop work s are “of”, “his”, “you”, “ours” etc. The list of stop word is easily available in internet and these set of stop words are included, when converting data to numerical data.

Stemming:- As said before, stemming is the process of shrinking different forms of same word to its original form. It’s like converting the verb form a word to noun form. Few examples of stemming a word are “processed”, “processing” etc. The stemmed form of the word is “process”. The process of converting the all other forms of same word to its root form is called stemming. There are many popular algorithms for this. I have used porter stemmer algorithm which is open source.

Transforming Word Frequencies

Once term frequencies are found for each document. The Inverted index is found using this term frequencies which is used to summarize or cluster the documents.

Inverse document frequencies:

Inverse document frequency is found by taking a term from a single document and check across all other documents under consideration for a match. If it matches in more than certain threshold, then it is included in the inverted list. This is done for each term in each document. As a result of this, we get unified table of document vectors which represent all the documents under consideration. Below is the formula used to find out the inverse document frequency.

$$idf(i,j) = \begin{cases} 0 & if\ wf_{i,j} = 0 \\ (1+\log(wf_{i,j}))\log\dfrac{N}{df_i} & if\ wf_{i,j} \geq 1 \end{cases}$$

In the formula above, wf(i,j) is the word frequency for a document. And document frequency is found by df. Df(i) is the document frequency of the i'th word. Idf (i,j) is the inverse document frequency, where 'i' represents the word and 'j' represent the document.

5. Architecture

5.1 Sequential Approach

Architecture is the back bone of any software/system. Architecture plays a major role in the success of any product. In architecture, one has to take complexity and size into consideration to make a system/software feasible.

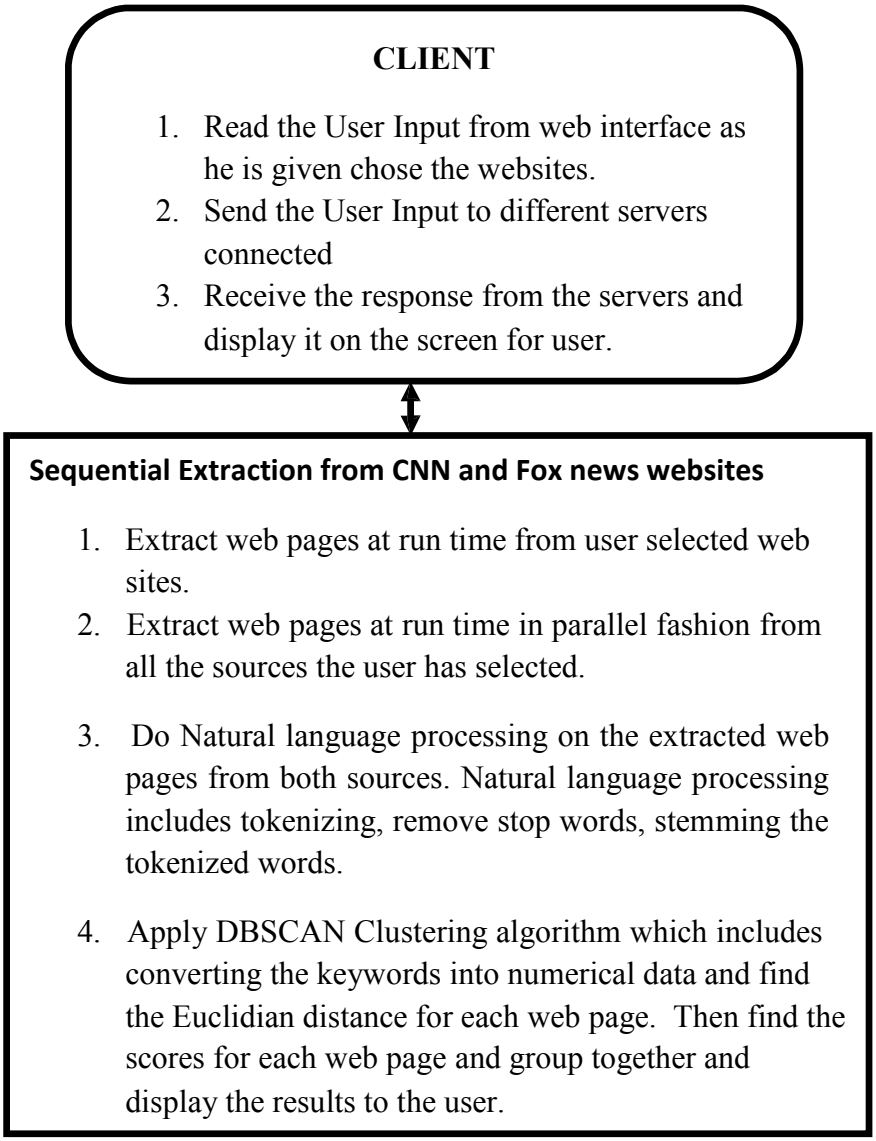


Figure 1: Architecture of Sequential Processing

In my project, I am analyzing two architectures based on time complexity. As mentioned before, my web application has a web interface where user can chose from predefined set of websites

and news articles are displayed in a clustered manner, where each cluster will have articles relevant to a specific topic. When a user likes a web page from a particular cluster, then the user is recommended with other (articles) URL's in the same cluster.

In the first architecture, user is provided with a web client, where in user selects his choice of websites and that user input is send to a single server at the backend. The server extracts URL's from those each selected sources (websites) in a parallel fashion i.e the articles from one website are extracted and after that the second website is taken into consideration and extraction is done. So, extraction is done sequentially. Then a clustering algorithm called DBSCAN is applied on the combined web pages. In the figure 1, you can see detailed block diagram of first architecture.

5.2 Parallel approach

Overview:

In the second architecture, I applied parallel programming to better the time complexity compared to first architecture. As in my web application, I am extracting web pages from two different websites. I decided to extract web pages from each source (search engine) in a separate server, so that the time to extract the web pages is almost half. In the figure 2, you can see the generic block diagram of second architecture.

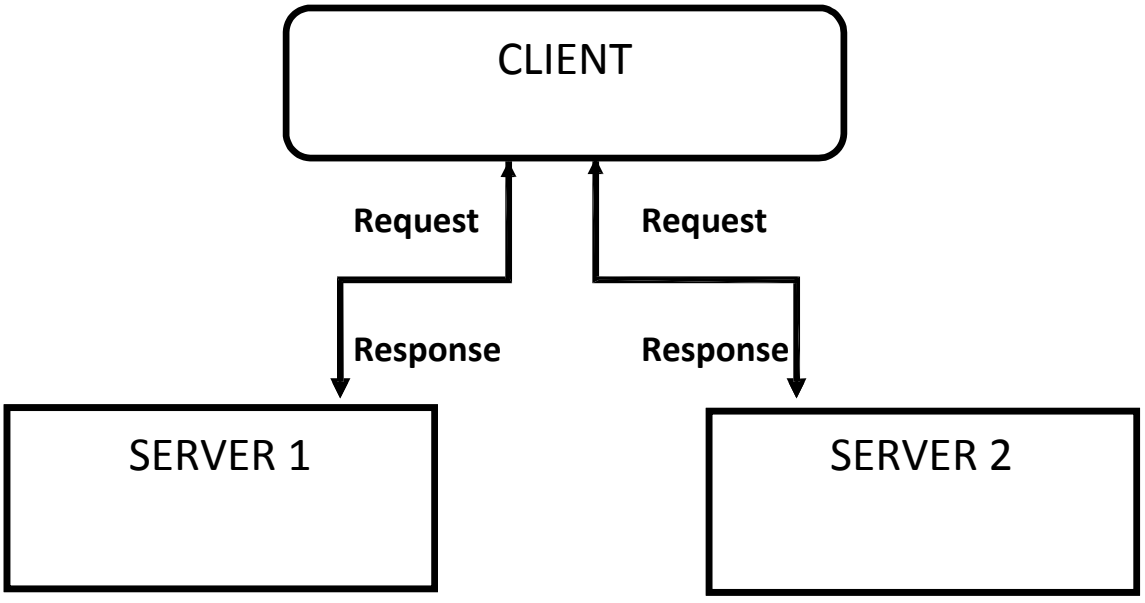


Figure 2: Architecture of Parallel Processing

5.3 Architecture of Traditional Data Mining

As mentioned before, user input is taken from web interface and send to two servers at the backend. Where in each server extracts web pages from different sources and then a clustering algorithm is applied on the web pages extracted independent of web pages of other servers. Clusters are formed within the web pages of each server and clusters are displayed to user from each server independently. In the figure 3, you can see a detailed block diagram of above explanation.

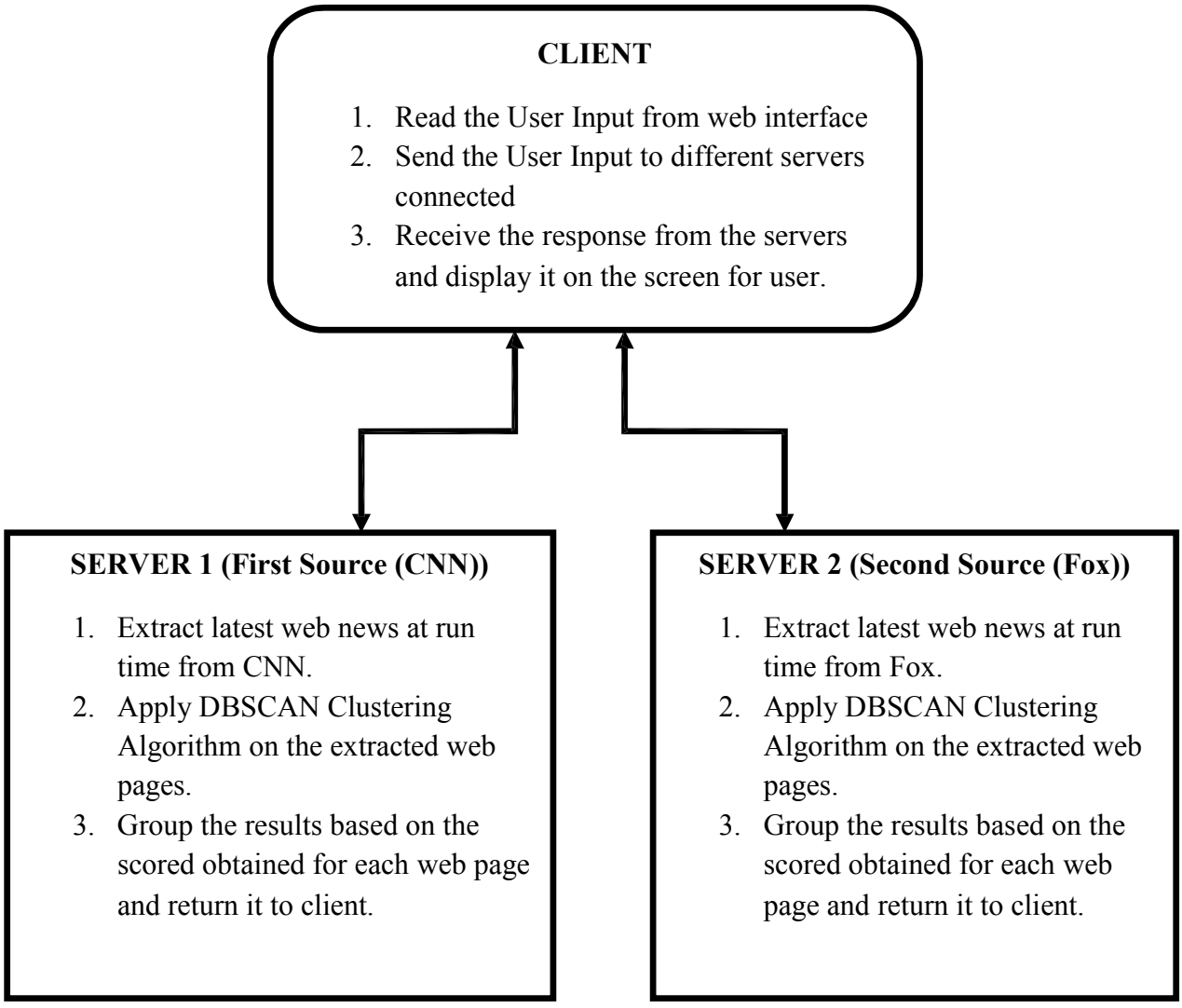


Figure 3: Detailed Architecture of Parallel Processing

5.4 Architecture of Distributed Data Mining

There are few short comes in the block diagram 3. Mainly, the clusters of one server are independent of other servers. Also, parallelism can be applied when extracting web pages from one server. As you can see from figure 4, once the user input is sent to the multiple servers, each server extracts web pages in a parallel fashion, where as in the previous case, the web pages are extracted in a sequential manner. This will improve the time complexity more. So, parallelism is applied in two layers, one by having multiple servers and other by extracting web pages from each server in parallel fashion.

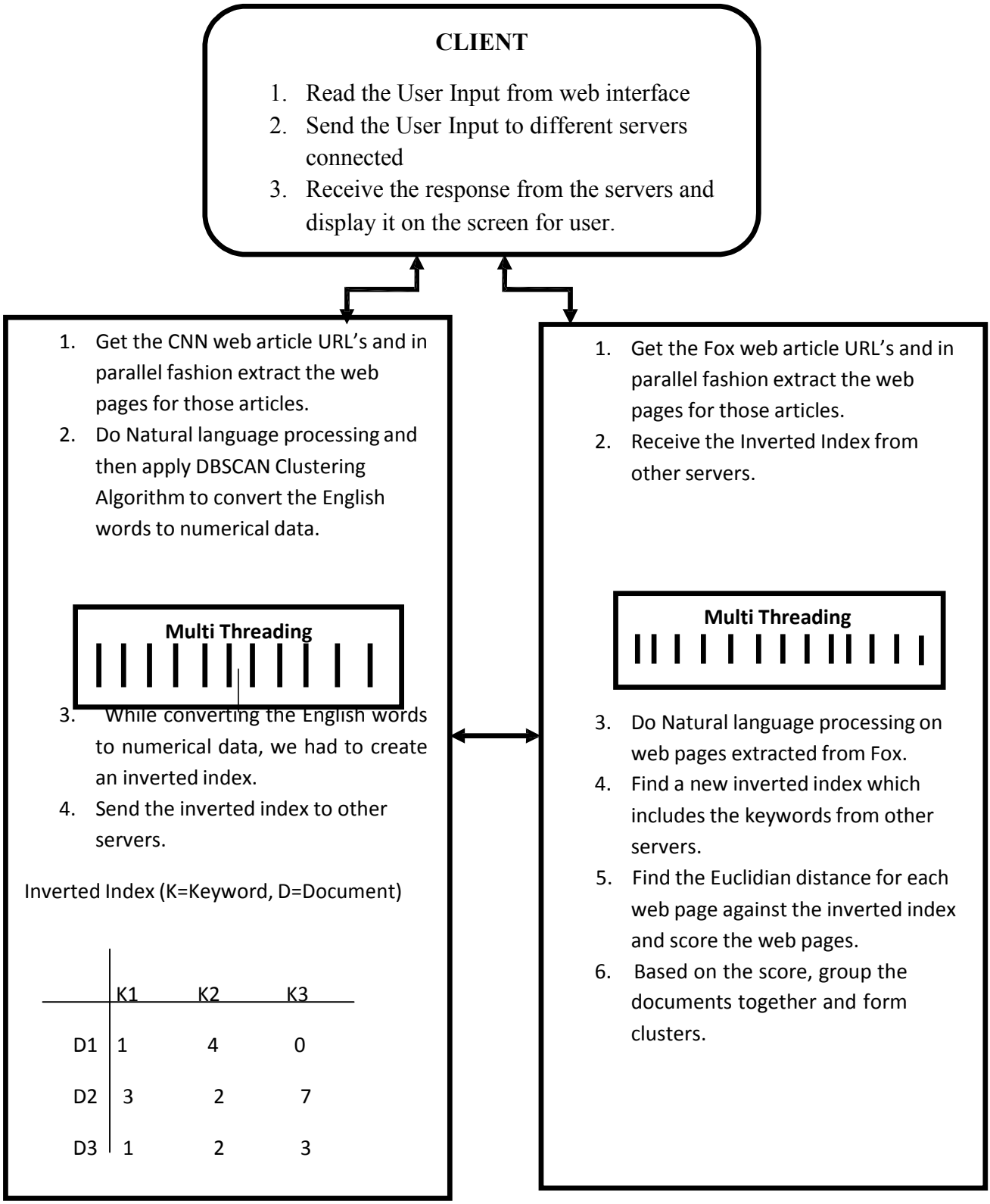


Figure 4: Architecture of Distributed Parallel Processing

6. Preliminary Work

6.1 Making Java Sockets to work

As mentioned before, Java sockets are used to connect from one system/server to other using sockets. Socket acts as an end point in a two way communication link between two systems running on the network. Every socket has to be bound to port number so that, the data send through it (TCP layer) can be identified by the application destined.

In this example, we are sending the data to an IP address called 10.185.16.95. To receive the data from Client, the server has to listen on a Particular port. So, the client has to specify the IP address and port number to which he wants to send the data and Server (System with that particular IP address) has to listen for data on that particular port number to receive the data. Once the Socket connection is established, both Client and Server can send and receive the data to each other.

Client creates socket on a port to connect to server using the statement below

Con = new Socket("IP address",portnumber);

Example :

```
requestSocket = new Socket("10.185.16.152", 2004);
```

Server creates a socket server to listen and establish the socket connection by using the statements below

serverCon = new SocketServer("portnumber","number of message at a time);

Example:

```
providerSocket = new ServerSocket(2004, 10);
connection = providerSocket.accept();
```

The above steps establish the socket connection. Client and Server can send/receive the data using statements below

To send data :

```
out = new ObjectOutputStream(connection.getOutputStream());
out.flush();
```

To Receive data :

```
in = new ObjectInputStream(connection.getInputStream());
```

6.2 Reading the data from User (web interface) and displaying the response to web interface

In our Web application, we should be able to read the keyword from user through web interface and process it in the backend and respond to request.

We provide a form as web interface, where the user can give the input keyword. We then read the input keyword given by user using Java Servlets. The Form provided can either send the user input using GET or Post method. In Get method, actual data sent is present in web query string (name value pairs) on the browser. In Post method, actual data sent is not transparent in web query string and is more secure. However, If Get method is used, Java Servlet can only read the user input in doGet method. If Post method is used, Java Servlet can only read the user input in doPost method.

6.2.1 To Read Data in Java Servlet

Java Servlet provides a predefined object "Request" which will get the parameters (data given by user) from form. Below statement is used to read parameter from the Form.

```
String keywords = request.getParameter("keyword");
```

The data send by user will be stored in the string called "keywords".

6.2.2 To Display data on web interface

Using a Java Servlet, we can create a Html page in it. This can be done by using a predefined object called "Response". This response object will allow the developer to display the data on web interface. Below statement is used to create a html page and display data in it.

```
response.setContentType("text/html"); PrintWriter out =
response.getWriter(); out.println("<H2>Data Extracted
and Zipped</H2>\n<br>");
```

6.3 Extraction of web pages based on user input

In this web application, I should be able to extract the web pages which are related to the user input. To achieve this, I query the popular search engines like CNN and Fox. These search engines have powerful algorithms to get the web pages based on the user input. So, I leverage these search engines to get wide range of web pages for that user given input. The below code snippet will query the search engine and download the web page with results.

```
url   = new URL("http://search.yahoo.com/search;_ylt=?p=" + keyword);
urlConn = url.openConnection();
inStream = new InputStreamReader( urlConn.getInputStream());
buff= new BufferedReader(inStream);
Line = buff.readLine()
```

In the above code snippet, the first line queries the search engine with user input. In the above, the user input is present in a variable called “keyword”. Lines after the first line will open the connection to web pages and download the web page.

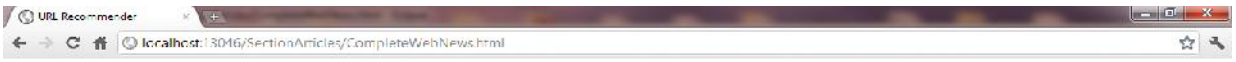
6.4 Parsing the downloaded web pages to get more URL’s relevant to user input

Once we download the web pages which have results from search engine, we need to parse through these downloaded pages to extract the URL’s (web results) present in them. To extract the url’s from downloaded web results, we need to find the pattern in web page to extract them and do further computation. The below code snippet will match the patterns given and extract the required URL’s.

```
Pattern p = Pattern.compile("<a class=\"yschttl spt\" href=\"(.*?)\" data");
// Create a matcher with an input string
Matcher m = p.matcher(nextLine);
boolean result = m.find();
news = m.group(1);
```

In the above code snippet, the first line will create a pattern and matchers are found out in the second line of code. Third line will check whether there is any match for the pattern given. Fourth line will save the URL matched into variable news.

In the figure 5, you can see that, the user chooses the website from which he want to extract. In the figure 6, you can see all the URL’s extracted for that user given input. However, in the background, it passes the user input to servlets and that is passed to server through java sockets and web pages are downloaded and URL’s extracted as discussed in the previous sections.



Finding Similar articles By Applying Distributed DataMining and Making Recommendation

Choose the options and the application extracts data at run time and recommendations are made on those extracted articles

CNN - Website :
Fox News - Website :
Extract

Please be patient..Extraction takes a minute or two

Figure 5: Screen Shot of Home Page of the URL Recommender

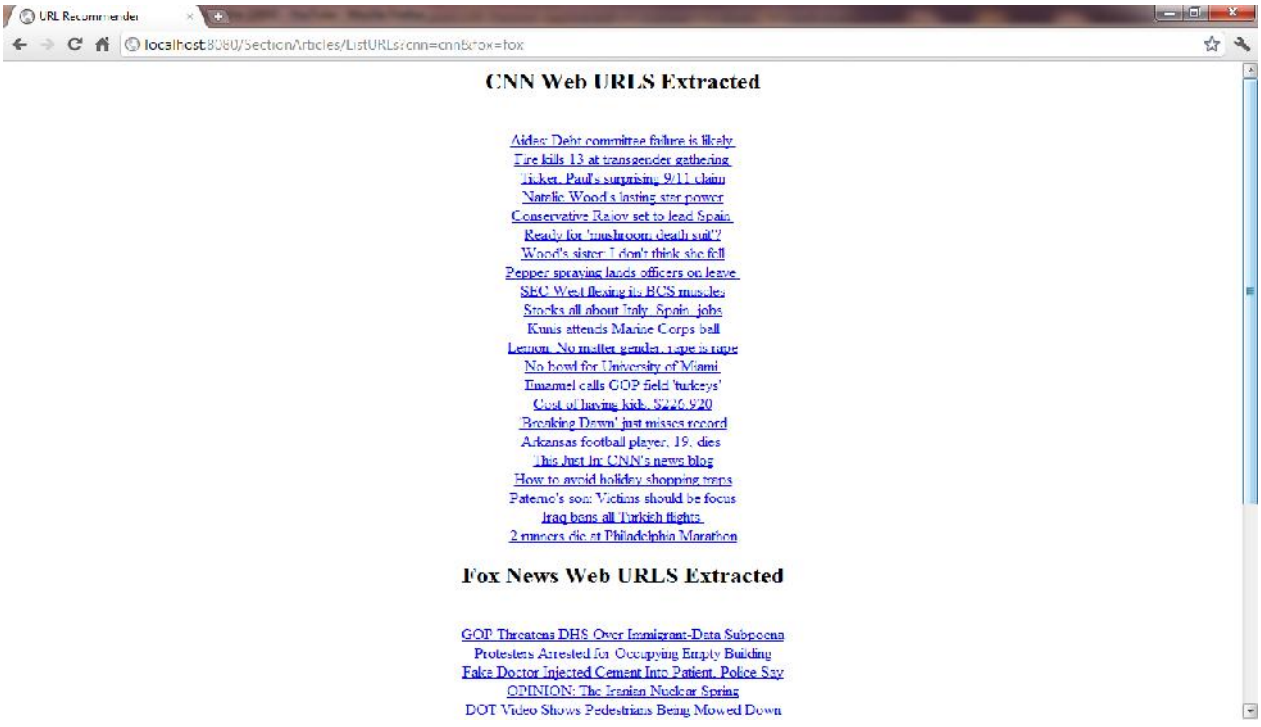


Figure 6: Screen Shot of List of URL's extracted at Run time for the given Keyword

7. Implementation

7.1 Natural Language Processing

As mentioned before, Natural Language Processing includes

1. Parse for the paragraph data in the web page.
2. Tokenizing the web page words.
3. Removing stop words.
4. Stemming words.
5. Finding the inverted index.

7.1.1 Parsing for paragraph data

When we download the document, we need only paragraph data and nothing else. To do that, we remove all the HTML tags and java script code from the downloaded document. We parse through the document for and and get the data in between those tags. But in some cases, there is possibility that, even the paragraph tag can contain HTML tags inside it like <img> etc. We need to take out html and java script code, so that we can tokenize only text content and words which are important for the document. Below is the code to parse through the document content and extract the required content.

```
paradata = paradata.replaceAll("<p>(.*?)</p>",".");
```

Variable “paradata” contains the content of the document and whenever it matches with pattern in the first argument, it replaces with the second argument in the whole document. Also, in the first argument, we have (.*?) which is a non greedy approach of matching. Once we match something of that pattern in the document. It stores the content within that paragraph tag.

7.1.2 Tokenizing the web page words

When we are done with parsing of the document, we will be left with only text content and no HTML or java script code. Then we tokenize the whole document against a delimiter or set of delimiters like space ( ) or colon (:).

```
String data = readFileAsString(filetoread);
StringTokenizer st = new StringTokenizer(data, "=;->\n-_<>:. 123456789,\"");
String key = st.nextToken();
```

First line code reads the entire document content into a string named “data”. In java, we have a predefined class for tokenizing called “StringTokenizer”. The first argument for this class’s constructor is the actual string of content (document content) and send argument is the set of delimiters against which document content is tokenized. Third line of code reads a one token (word) at a time.

7.1.3 Removing stop words

Once we are done with tokenizing the words in the document against a delimiter. We have the all the words from document in a list form. But, all words in the document are not important for indexing a document and also for finding the uniqueness of the document. The most common words are the least important ones for indexing and finding the uniqueness of the document. These most common words are called stop words. Some examples of stop words are “a, the, an, was, is” etc. To do this, I have downloaded the list of stop words from internet and removed all the stop words from all the documents.

7.1.4 Stemming the words

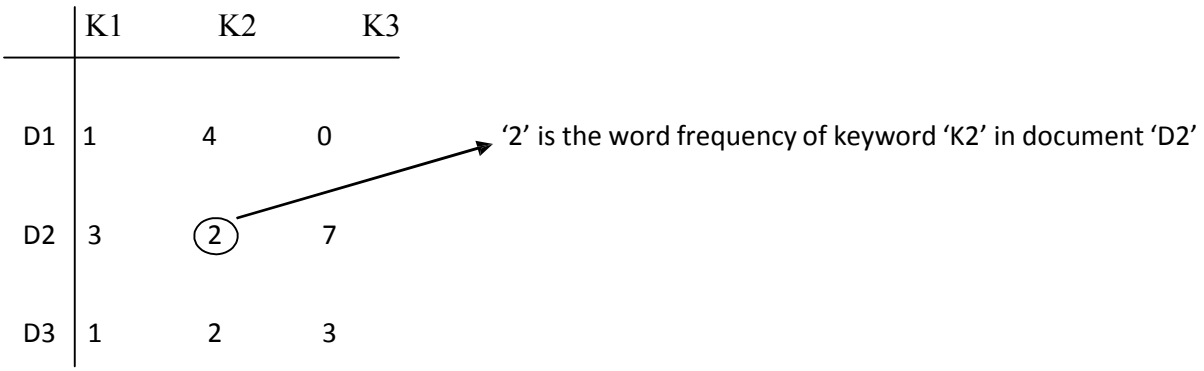
Once we are done with removing stop words, we will be having a shorter set of words for each document. But, there may be duplicates of a single word in different forms. For example, “considered” and “considering” are derived from root word “consider”. Considered and

considering are different forms of the same word consider. And only root words in the document needs to be considered for getting accuracy in the word frequency and word uniqueness. So, all the words in the documents are stemmed to its root form. If the word is already in root form, it is left alone. To do the stemming, I have used a popular algorithm created by "porter". The stemming program is usually called porter stemmer. So I have used that porter stemmer code to stem the words to its root form.

7.1.5 Find the inverted index

Inverted index by definition is finding the word frequency in that particular document and finding the frequency of the word against all the documents under consideration. Then, take the few key words which are unique and can represent the document.

In my project, I found out the word frequencies in their respective documents and then found the word frequencies against all other documents under consideration. But I got a huge set of key words which are matching in all keywords. So, to make it simple and more accurate, I kept a threshold for word frequencies against documents. In additions to that, to specify a section name in cluster, I took the keyword from URL, title and give them highest frequencies within that document. So that this keyword is present in inverted index and can used to specify as section name in later stages. So when the clusters are formed, the highest frequency word from those set of documents is taken and kept as name of the section. The word which appears in most of the documents and crosses the threshold is considered to be a keyword. This process is done for each word in each document. Then we can find a set of keywords that represent all documents. Below is a figure, which will show a sample inverted index table.



As you can see from the above table, D1, D2 and D3 represent the documents and K1, K2 and K3 represent the keywords. These keywords are the words which cross the minimum number of documents, they should appear. The values in the table are word frequency of the keyword in that particular document. For example, Keyword 1 (K1) has word frequency of 1 in Document 1 (D1). Similarly, Keyword 3 (K3) has word frequency of 0 in Document 1 (D1). The above table is called the inverted index table.

7.2 DBSCAN clustering algorithm

Density Based Spatial Clustering of Applications with Noise (DBSCAN) plays a major role in clustering non linear shapes (structures) based on the density of the data. DBSCAN is one of most popular density based clustering algorithm. As we know, detecting number of clusters for data points is very challenging as we are unaware of what kind of data we are dealing with. This issue becomes more significant when dealing with huge amounts of data which might include noise. DBSCAN algorithm solves the problem of fixing the number of clusters to be formed even before the algorithm is started. DBSCAN algorithm basically uses the concepts of density reachability and density connectivity.

Input parameters which are used in this algorithm are.

1. Size of epsilon neighborhood (e): The radius (e) is used to decide upon, whether the data point comes in that cluster or not.

2. Minimum points in a cluster (m): Data points that should present within the epsilon neighborhood (e) (radius).

A cluster is formed, when data points present within the radius (e) is more than or equal to the specified minimum number (m).

Before outlining the algorithm, it is important to get familiar with the terminology used in this algorithm for data points. Data points are classified in to three types

1. Core data points: These are the data points in a cluster which are present within the specified radius and also have required minimum number of points in it.

2. Border data points: These are the data points in a cluster which are present within the specified radius, but do not have the minimum required number of data points in it.

3. Noise point: A data point is considered to be a noise point, when it is neither a core point nor border point.

As mentioned before, the two concepts used by DBSCAN algorithm are

1. Density Reachability: A data point ‘A’ is said to be density reachable from data point ‘B’, when the data point ‘B’ has minimum number of points within the radius (e). And also data point ‘B’s is in the radius (e).

2. Density Connectivity: This is a chaining process. Two data points ‘A’ and ‘B’ are said to be density connected if and only if there exists a data point ‘C’ which has the required

minimum number of data points within the radius (e) and the points ‘A’ and ‘B’ are present within that radius. Let’s say, if ‘A’ is neighbor of ‘B’ and ‘B’ is a neighbor of ‘C’, then this implies ‘A’ is a neighbor of ‘C’.

7.2.1 Algorithmic steps for DBScan Clustering

In my project, we have set of documents and we have built an inverted index table for all documents under consideration. The process of generating the inverted index is also called converting document content to numerical data.

So let’s say we have 5 documents namely D1, D2, D3, D4 and D5. And all these documents are represented by 5 keywords (K1, K2, K3, K4, and K5). As shown in the table before, each document has a numerical vector of data corresponding to it. Let’s say D1 has numerical vector as D1 = {1, 5, 7, 2, 0}. In a similar way, each document will have a numerical vector related to it, which we get from inverted index table.

The other important term which is used in DBSCAN clustering algorithm is Euclidian distance. Euclidian distance by definition is finding the distance between two data points. Euclidian distance found out by square root of squared differences between the two data points. The formula is given below

$$\sqrt{(C1-X1)^2 + (C2-X2)^2+.....+(Cn-Xn)^2}$$

Where C1,C2…Cn are vector values in the center C and X1, X2 … Xn are values of data point X.

Now, let's say we have 5 data points D = {D1, D2, D3, D4, D5}. Below are the steps of K-means clustering algorithm

STEPS:

1. We start off by selecting a random unvisited data point.
2. We then extract the neighborhood data points for this data point. Neighborhood data points are those data points which satisfy the criteria of minimum data points within the radius (e) specified. The distance between two points is determined by Euclidean distance which in turn will decide whether the point is within radius or not.
3. If the selected data points meets the criteria of minimum data points within the radius (e), then the clustering process starts and this point is marked as visited. If the point doesn't meet the criteria, then it is marked as noise point for now. And Later on in different iteration I will be revisited.
4. If the selected point meets the required criteria, then all its neighborhood data points come into this cluster. And each point's neighborhood data points are found out and if they meet criteria, then the neighborhood data points are also included in the same cluster. Then step 2 is repeated until all the data points are determined in a cluster.
5. If there are any unvisited nodes, they are visited and processed leading to formation of new clusters or noise.
6. This process of clustering continues until all the data points are marked as visited.

7.2.2 Advantages

1. In DBSCAN algorithm, we don't have to specify the number of clusters to be formed as it is clusters are formed at runtime while clustering.

2. DBSCAN algorithm is able to identify the noise data points while the process of clustering is done.

3. DBSCAN clustering algorithm forms clusters which are in arbitrary shape and size as shown in the figure 7 below.

7.2.3 Disadvantages

1. DBSCAN algorithm has troubles finding clusters when considered data is of high dimensional.

2. DBSCAN algorithm also doesn't work well then clusters have varying densities.

7.3 Naming of Clusters

As said before, the keywords from the URL and title are given highest score (weight) in their respective documents while finding the term frequencies and forming the inverted index. By doing this there is higher possibility of having these keywords in inverted index. Once this inverted index is found, it is converted to numerical data as said before. This data is used in clustering process. When forming a cluster, we take the highest frequency word from all the data points and keep that name as cluster name. For example, a cluster is formed for documents d1, d2, d3 and their respective numerical data and term frequencies are also known. Once the cluster is formed, we take out highest value from the numerical data and take out that term from that

document and place it as the name of cluster. This process is done for each cluster formed.

Figure 7: Screen Shot of Clusters formed for extracted articles

8. Results

As discussed before, we are analyzing parallelism at two levels. In the first level, we are achieving parallelism when we are extracting from multiple sources at the same time using multiple servers in the backend. Second level of a parallelism is achieved when extracting web pages from each source for the user given keyword at runtime using Java threads.

In this section, we will compare the time taken to extract and cluster at different levels of parallelism.

1. Comparing between Sequential and different servers (sources) extracting and clustering web pages (but no multithreading while extracting the web pages):

Around twenty times, the test is run with different keywords as input and times taken are recorded. The average time taken to sequentially extract and cluster web pages took about 72367 milliseconds per test which is about 72 seconds for one single test. The average time taken to extract and cluster web pages in parallel fashion with multiple servers in place took about 55237 milliseconds per test which is about 55 seconds for one single test. You can see the results in bar chart in the below figure. X-axis represents the test runs and Y-axis represents seconds scale. Red bar (first bar) is the results of sequential time and Yellow bar is for time of multiple servers.

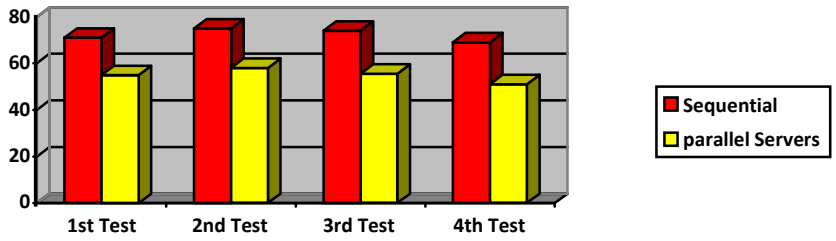


Table 1 : Graph showing the time comparisons of sequential processing and parallel processing through servers

2. Comparing the time taken when clustering is done sequentially and when clustering is done in multiple servers and also the web pages are extracted in parallel fashion in each server:

The same keywords are used as in the previous test which is about twenty test runs. The average time taken to sequentially extract and cluster web pages took about 72367 milliseconds per test which is about 72 seconds for one single test. The average time taken to extract and cluster web pages in parallel fashion when web pages are also extracted parallel in each server is about 28537 milliseconds per test which is about 28 seconds for one single test. You can see the results in bar chart in the below figure. X-axis represents the test runs and Y-axis represents seconds scale.

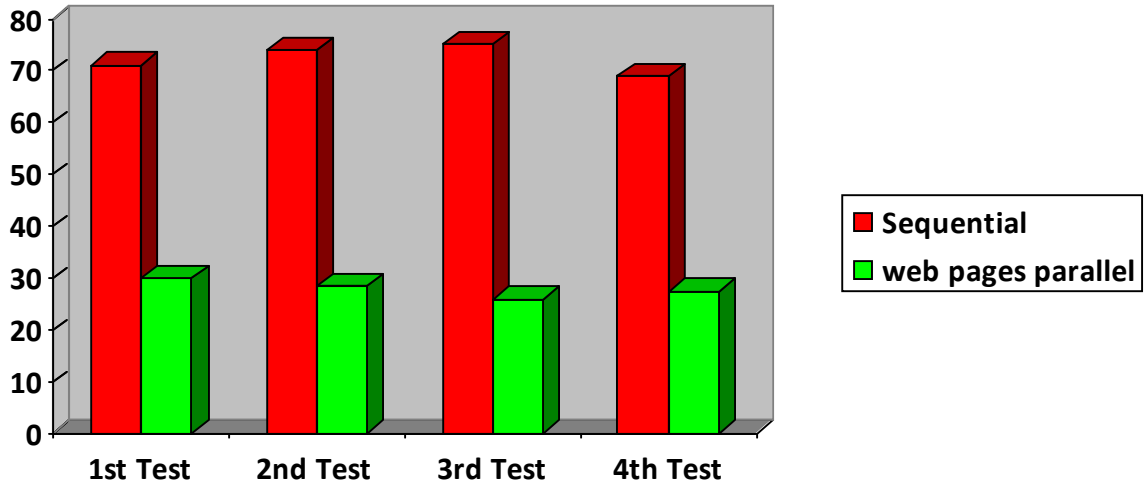


Table 2: Graph showing the time comparisons of sequential processing and parallel processing through servers and multithreading within

When comparing three tests at the same time, then we can see a drastic amount of time difference. When we did the test in parallel fashion at multiple levels, the time taken was less than half of the time taken by sequential test. Below is the Histogram with three kinds of test results together.

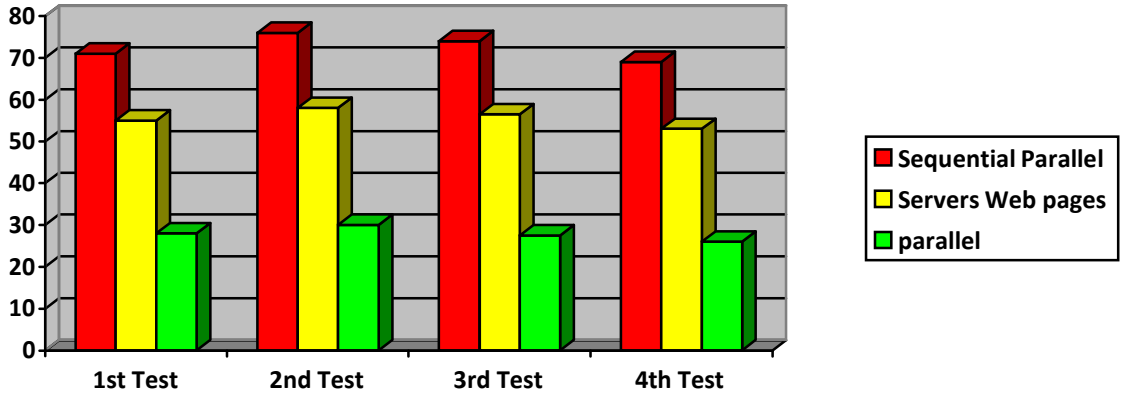


Table 3: Graph showing the time comparisons of sequential processing, parallel processing through servers and parallel processing through servers and multithreading(all together).

Factors to be considered:

1. One important factor to consider is network overhead. The time taken to send/receive user input and result is important.

2. Also, other important thing to consider is network congestion.

3. Computation power of server is important for performance.

9. Analysis

Now that clustering of web pages are done in both parallel and sequential ways. How do we know that the results of the clustering are good? How do we know, whether the web pages in each cluster are relevant to each other or not?

To answer the above questions, I have performed a subjective test by taking the opinions about the clusters from 15 people. These questions were asked to them

1. How good are the clustered results?

2. Do web pages in each cluster relate to each other?

After the survey, 60% of the people found the results to good and each cluster has web pages relevant to each other. In the rest 40%, some people felt that the results were okay and some people felt, the web pages in the cluster are not at all relevant to each other.

Reasons for results to be bad:

As discussed before, one of the main reasons for bad results is the bad choice of cluster centers. If cluster centers are close to each other, then the Euclidian distance used to find the score between data points and cluster centers will not able to group the web pages in the right cluster. A data point whose score is closest to zero with a particular cluster center, then it is taken into that cluster. If the cluster centers are near to each other, then score will not be definite and results will be bad. The only way to resolve this problem is to choose the cluster center as distant as possible so that we get a definite score for each data point.

10. Conclusion

In this project, a web application using java was developed and the report focused on the advantages of parallel processing over sequential processing of a web application (URL Recommender) and detailed implementation of how this web application is developed. This web application reduced the time taken to process the user's request and at the same time provides the user with good web page recommendations. The speed and productivity of this web application is much better than the usual sequential web application. Although there were few factors to be considered, mainly the hardware requirements were more in parallel processing which is not the case with sequential processing. However, we were able to see the performance difference significantly.

This web application can be extended to extract web URL's with in web pages which will improve the quality of recommendation.

11. References

[1]. Mariam Rehman and Syed Atif Mehdi : Comparison of Density Based Clustering Algorithms.

[2]. Slava Kisilevich, Florian Mansmann and Daniel Keim : P-DBSCAN: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos.

[3]. DBSCAN Clustering Algorithm. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/DBSCAN

[4]. DBSCAN Clustering Algorithm explained. Retrieved from Wikipedia:

https://sites.google.com/site/dataclusteringalgorithms/density-based-clustering-algorithm

[5]. Adriano Moreira, Maribel Y. Santos and Sofia Carneiro :

Density Based Clustering Algorithms – DBSCAN and SNN (Version 1.0)

[6]. Cluster Analysis – DBSCAN. Retrieved from Wikipedia:

http://www.hypertextbookshop.com/dataminingbook/public_version/contents/chapters/chapter004/section004/green/page001.html

[7]. Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu: A Density - Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", The Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA, 1996

[8]. Levent Ertoz, Michael Steinback, Vipin Kumar: "Finding Clusters of Different Sizes, Shapes, and Density in Noisy, High Dimensional Data",

Second SIAM International Conference on Data Mining, San Francisco, CA, USA, 2003

[9]. Bradley PS, Fayyad U, Reinna C, 1988 : Scaling Clustering Algorithms to Large Databases.

*[10]. R. Ng and J. Han, 1994, Efficient and Effective Clustering Methods for Data Mining. Proc. Of 1994 Int'l Conf. On Very Large Databases (VLDB'94), Santiago, Chile, pp 144-*155