بسم الله الرحمن الرحيم

BAG-OF-WORDS MODELING FOR HIDDEN WEB CRAWLER

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR IN COMPUTER SCIENCE AND ENGINEERING

Njingamndap Ousmanou Mohamed
(114450)

Supervised By
Md. Kamrul Hasan, PHD
Associate Professor

Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)
The Organization of the Islamic Cooperation (OIC)
Gazipur-1704, Dhaka, Bangladesh

Certification

This is to certify that this thesis entitled "BAG-OF-WORDS MODELING FOR HIDDEN WEB CRAWLER" is a true work of NJINGAMNDAP OUSMANOU MOHAMED (114450) who successfully carried out the thesis work under the supervision of **Dr. Md. Kamrul Hasan,** associate professor Islamic university of Technology. This thesis counts as my final year research work which puts an end to my four years program as bachelor student in computer science and engineering at the Islamic University of Technology (IUT) Dhaka, Bangladesh.

Author:

Signature:

Name: Njingamndap Ousmanou Md

Roll Number: 114450

Date:

Supervisor:

Signature:

Name: Dr. Md. Kamrul Hasan

Date:

Head of Department

Signature:

Name: Prof. Dr. M.A. Mottalib

Date:

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

ACKNOWLEDGMENT

Thanks to Allah (SWT), the most gracious and the most merciful. The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it Possible, whose constant guidance and encouragement crown all efforts with success.

I am grateful to my project supervisor Dr. Md. Kamrul Hasan for the guidance, inspiration and constructive suggestions that help me in the preparation of this thesis work. I am also thankful and grateful Systems and Software Lab (SSL) team for their great support throughout this research, particularly Hasan Mahmud and Mohiuddin Khan for all the tremendous useful criticisms and feedbacks.

Finally i also wish to take this opportunity to express my sincerest gratitude and heartiest thanks to the head of CSE department, Prof. Dr. M. A. Mottalib, as well as the Organization of the Islamic Cooperation for endless support it provides in the field of education.

Gazipur Dhaka, Njingamndap Ousmanou Mohamed (114450)

Novenmber 2015

DEDICATION

THIS THESIS IS DEDICATED TO MY PARENTS WHO BROUGHT ME INTO THIS WORLD, AND WHOSE LOVES REMAIN IN MY HEART FOREVER

ABSTRACT

The internet is an integral part of our daily life, many of our activities we do it online like gaming, education and so on. Before getting any information online we have to search for it, thus web search engine like Google which are web applications which search over the internet for websites, become very popular. Search engine was there at the early age of internet and become the tool for searching. Search engine has a component which is charged of collecting information that we see on the interface when searching, it is called “crawler”. The crawler collects the information by retrieving the content of the pages it has the link, then look inside the content and take all the links. Hence it can resume the process of retrieving the content of these new links and collect all the content. The process continues until there is no more links to retrieve the content.

However, on the internet information are not only on the pages we see, there are a tremendous amount of information that we are not seeing inside databases this is called hidden web.

To access this information a normal user has to fill up form to access it. Like in the ecommerce websites, we have to give some information on the research box to get information about a product we are looking.

The crawler however has to be able to put the correct words on the form to crawl the content inside databases. Many researches have been going out to solve this issue, to determine the optimal list of words to fill up the form with. The list of words is also known as the bag-of-words (BOW).

In this thesis, we propose a model of structuring words inside the *bow* to reduce access to a words and set of related words on the *bow*. We have successful implemented the structure and integrated it to a crawler we have also implemented and showed the

result of experiment. From that we are able to see the effect of the structure in a crawler.

TABLE CONTENTS

1. Introduction

Retrieving the information from the internet has always been a passionate and very interesting field for the information retrieval researchers. Search engine is at the heart of information retrieving. Search engine is made of three parts:

- The search box interface: which is the parts which is in charge of receiving the input request of users and format it to send to the database server to process the request and send the response back to the interface which will format it to show to the user.
- Index database: It is the portion of the search engine in which information are stored.
- Crawler: which is the program in charge of traversing the internet in order to retrieve the information stored in the web pages and stored it in the indexed databases which will be later sent as a response to a request to users.

Crawling is the process of exploring web applications automatically [22]. The crawler explores the internet by starting from a set of seed of links, then retrieves the content of links and collects all the links. Hence it start the process of retrieving the content of the new links it as found then retrieves the content again until there is no more new links to crawl. However the information of the internet is not only limited to the content of a web page. There is a tremendous amount of information stores inside databases that to access it we have to fill up a web forms. For a normal user we know which words to use. But the crawler has to know which words is appropriate to fill up the form. The content behind web form is called *hidden web* or *deep web.*

In order to fill up the web form and get more information from the backend databases, we have to fill up the form with data related to the website. Vertical web crawlers (topic focused crawlers), which are crawler that to look for a particular topic on the web, are

proposed to crawl only information related to a particular topic. These crawlers will have predefined keys words related to a particular topic and fill up the form with these key words and update its bag of words with the output of the form. However the vertical crawlers will be useless on the websites which are not related to the topic they are mining and there will be some waste of resources while interacting with the server since they will have to go through their whole set of words to crawl the website whereas it was only necessary to go through a small set of the list of words. In this chapter we will the motivations of the research in section 1.1. Finally in section 1.2 we summarize the contributions of this research work in addressing this challenge.

1.1Motivations

The web has started as a connection of multiple pages, and the crawlers had to go through the links connected the pages to find the content of the web. The web applications have evolved a lot, letting to introduction of forms on which user has to enter data in order to retrieve the information in the side the data bases. These categories gave born to a new type of web crawlers which do not merely follow links but have to fill up the web forms to retrieve data inside databases. These types of crawlers are called hidden web crawlers. To retrieve information hidden in the deep web, the web crawler would submit the HTML form many times, each time filled with a different dataset. Thus the problem of crawling the deep web got reduced to the problem of assigning proper values to the HTML form fields [4].

In 1998, Lawrence and Giles [5] estimated that 80 percent of web contents were hidden in 1998. Later in 2000, Bright-Planet suggested that the deep web contents is 500 times larger than what surfaces through following hyperlinks (referred to as shallow web) [6]. The size of the deep web is rapidly growing as more companies are moving their data to

databases and set up interfaces for the users to access them [6]. Only a small fraction of the deep web is indexed by search engines. In 2007, He et al [7] randomly sampled one million IPs and crawled these IPs looking for deep webs through HTML form elements. The study also defined a depth factor from the original seed IP address and constrained itself to depth of three. Among the sampled IPs, 126 deep web sites were found. These deep websites had 406 query gateways to 190 databases. Based on these results with 99 percent confidence interval, the study estimates that at the time of that writing, there existed 1,097,000 to 1,419,000 database query gateways on the web. The study further estimated that Google and Yahoo search engines each has visited only 32 percent of the deep web. To make the matters worst the study also estimated that 84 percent of the covered objects overlap between the two search engines, so combining the discovered objects by the two search engines does not increase the percentage of the visited deep web by much.

From here, there are still lot of data remaining in the public web databases that are not still covered by the web search engine. We will only search on the public databases because accessing a private databases is not legal. These policies make the deep web to grow day by day.

In order to tackle the issue of selecting words to fill up, we have propose in this research work a model a network representing the connection among words. A Connection between two words represent the likelihood for them to appear on the same wen pages. The model is update as the crawling is going with the pages obtained from crawling. Hence Network of words will be used to find the most likelihood which may appear on the same pages with the words want to fill up the form. By this way the time to locate relevant words to a web pages will reduce, and also the way words are

connected will give a kind of grouping of words in categories, which is very useful in case of topic crawling.

1.2 List of contributions

In this these works we have define a model for generating the bag of words as a network of connected words. In this model words are nodes of the network and edges connecting two words are the likelihood of for the words to appear in the same documents. That means if the weight of the connection between two words is big thus there is high probability to be in the same web pages or documents. But this way whenever two nodes of the networks are found in the same document their weight will be increasing.

The network will grow as the crawling is going on. By this way the network will be update with the current information it collects from pages.

2. Related works

As we started earlier in the introduction, crawler is not something new, thus many research works have been going on in this field. In this chapter, we will discuss about some of the earlier works done in the field of deep web crawler and information retrieval in general. We will start by looking in the earlier generation of web crawler in section 2.1. Then we will go for horizontal web crawler also known as general crawlerin section 2.2. Then we will move to topic based crawler in section 2.3. Finally we will talk about some methods to detect useful information on the web or semantic content of web in section 2.4.

2.1 Traditional web crawling

Web applications can be represent as a direct graph where nodes are pages and edges are connection between two pages [22]. Early web crawler were designed to exploit this structure of web applications in order to crawl the whole internet. In 1993 four web crawlers where written *World Wide Web Wanderer, Jump Station*, and *World Wide Web Worm* [2]. These crawlers crawl the whole internet by following the interconnected web applications. In [3], Brin and Page tried to address the issue of scalability by introducing a large scale web crawler called Google. The main concern here is to download the maximum number of pages by making the crawler to run on multiple computers. However web applications have evolved. The content of web applications in not accessible only by following links between. Web applications have introduced web forms which the users have to fill up in order to retrieve information behind the form [12]. Bright-Planet estimated that the deep web contents is 500 times larger than what surfaces through following hyperlinks. This information is located inside databases. Thus the traditional web crawlers fail to crawl web applications containing the web forms.

Thus there is a need of new type of crawler to tackle the issue of exploring data inside web forms.

2.2 Horizontal deep web crawler

As stated earlier, contents of web applications are not only through the links between web pages. We applications have databases where they store data, and most often accessing this information is through web forms. Deep crawler is a crawler which tackle the issue of filling the web forms in order to get access to databases. The main purpose of the deep web crawler is to find the right words to fill up the forms with [5] given a websites. Many researchers addressed this issue. In [5], Raghavan and Garcia-Molina proposed a semi-automatic method to fill up the form that mostly depends on human inputs. Thus this system is not full automatic and can be used in an industrial concept. Also Liddle and al [9] described a method to detect form elements and make a HTTP GET and POST requests using default values specified for each field. This system is not fully automated and only focus on default values of the system whereas there are many others values that may be used to fill up the web forms and not all the web applications will give default values for html forms. Barbosa and Freire[4] proposed a two phase algorithm to generate textual queries. The first stage collected a set of data from the website and used that to associate weights to keywords. The second phase used keys words to retrieve the data. The previous process is improved by Ntoulas and al. [8] by defining three policies for sending queries to the interface: a random policy, a policy based on the frequency of keywords in a reference document, and an adaptive policy that learns from the downloaded pages. Given four entry points, this study retrieved 90 percent of the deep web with only 100 requests. In [6], M.Soulemane, M. Rafiuzzaman and H. Mahmud proposed a method to detect single input text, then fill it up with generated data to retrieve the maximum data from the web page. YU [8] proposed a

novel method for extracting entity data from deep web precisely. These approaches aim to retrieve maximum content of web applications. These are called horizontal web crawlers because they do not specify which content of the web pages they want to retrieve rather the retrieve everything. However these researches were only interested to have a set of words and fill up the form with the words inside the set. Then update the set of words with new words found from the output result. But web forms in web applications will not require the same set of words to retrieve the maximum number of words. By using all words in the set of words we can retrieve the maximum content of the web application. But instead of that, we can detect the sub set of words in the set of words (also called bag-of-words) which can retrieve the maximum hidden content of the web application. This process will obviously reduce the amount of requests and the time needed to retrieve the hidden contents of the web application. However these researches do not take into account relationship between words and content of the websites.

2.3 Vertical deep web crawler

Vertical deep web crawler is a deep web crawler that retrieve only data from the deep web that are related to a specific topic. They are also called topic focused web crawler. In [26], Panagiotis Liakos, Alexandros Ntoulas,Alexandros Labrinidis and Alex Delis presented a topic focused web crawler, in which the crawler eliminates irrelevant links from the result of the web forms, and this is done based on the information about the links which is given on the result of web forms. Thus it allows to avoid crawling links which are not relevant to the topic of the crawler. The crawler starts with a set of words related to specific topic, it crawls the web application these given words. However a topic is made of sub topics and for a single website using a sub set of *bow* may be

enough to retrieve the content instead of taken the whole content of the *bow.* This method does not specify a way to select a sub set of bow to retrieve the whole content. In our research we tackle this issue.

2.4 Semantic content of web

The semantic content of web tries to address the issue of detecting the sections of a webpages that contents the main information of the web pages. Meaningless information found on a page may be advertisements sections. Researchers have proposed algorithms to extract the real semantic content from HTML. The wrapper-based algorithms [13], [14], [15], and [16] are easy to design and may have high performance. But the trained templates may lose effectiveness quickly when websites change their templates. Tag ratio-based algorithms [17] [18] may extract the real semantic content without the assumption of fixed template, here the assumption is that the main information of the website is found on the line where the ratio number of words divided by the number of tags is the lowest one. In [10], the author proposed a method to retrieve the semantic content of web via repeated structure, this method is a mixture of both wrapper-based algorithm and the tag ratio-based algorithm. In order to separate the web page into different portions we will be taken into account repeated structure [10]. In our proposed crawler, we divide a web page into different section with the assumption that this sections are independent, that means the data content in each section are not related. Further explanation is given in chapter3.

In[27] Schonhofen proposed a method to identify document topics using the wikipedia category network, and this method can be used in order to find related terms to a given words. Crawling can use this method to find related words, however this is different from our method in the sense that the crawler is independent of any third party website

and detecting a topic is based on interconnection of words in the *bow.* A section of interconnected words can be defined as a topic.

3. Proposed Methodology

The main problem of the deep crawler is to find the meaningful words to fill up the form to retrieve the most relevant result from databases. Given a website, a web crawler may have already its bow filled with predefined words [26] or may look for words inside the page and fill its bow [4], [6], [8]. Using the content inside their bow, these crawler will fill the form and get the result then update the bag of words with the result of the HTTP request send to the server. However these methods do not take into account that words are related. There are words that most often appear together in the same area or topic or subtopic. Our solution concept consists of modeling the bow as a graph of words where words are interconnected. The connection between words represent the likelihood to appear in the same documents and we define what we mean by documents in the upcoming sections. Before going into detail of the generation of the bow, let see the overall working process of the crawler we define. To fully fulfill this task, the crawler follows the following steps:

1. Selection of key words from the web pages
2. Update the pool of URLs
3. Selection of terms highly related to previous keys words
4. Update the network of words
5. Send requests to webservers and then get the result.
6. Go to step 1 until the end of crawling

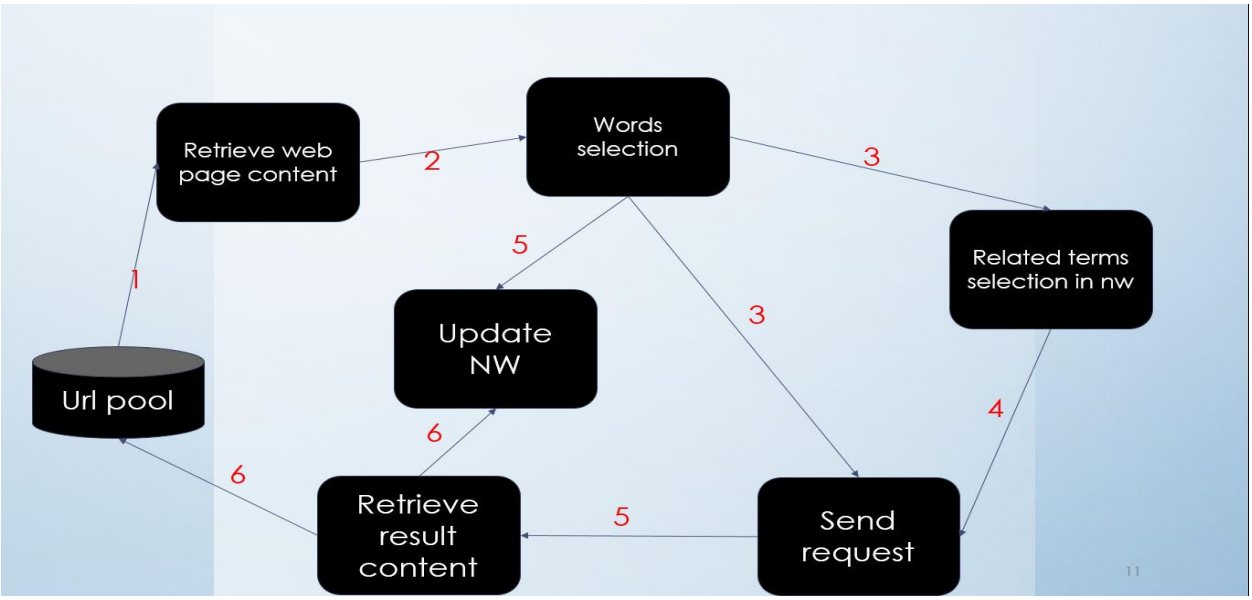


Fig.2 System Overview

In the following section we will go into details of these steps. In section 3.1, we describe the mechanism of section of keys words giving a web pages. Then in section 3.2 we describe the network of words, its structure, how the words are organized, the mechanisms of updating the network of words, and the mechanisms of selecting related terms in the network of words. In section 3.3 we have the sending request to the webserver. Finally, in section 3.4 we have the overall algorithms.

3.1. Selection of key words from the web pages

To find relevant keys words on a website, we will use TF (term frequency) and IDF (inverse document frequency) (without making any assumption about terms and stop-words, thus the crawler will support evolution of a language and multiple languages).

In order to apply IDF we need to a set of documents, in our case we define a document as parts of the web page defined by repeated and non-repeated structures. As we can

we on the pictures below the results of the forms are set of information into repeated structures. The information blocks follow the same appearance only the content of the information changes. Moreover the GUI of websites becomes more and more organizes into independent block thus we can take those blocks as the documents. Thus the most relevant key words will be those which repeated the least in components or partitions of the website.

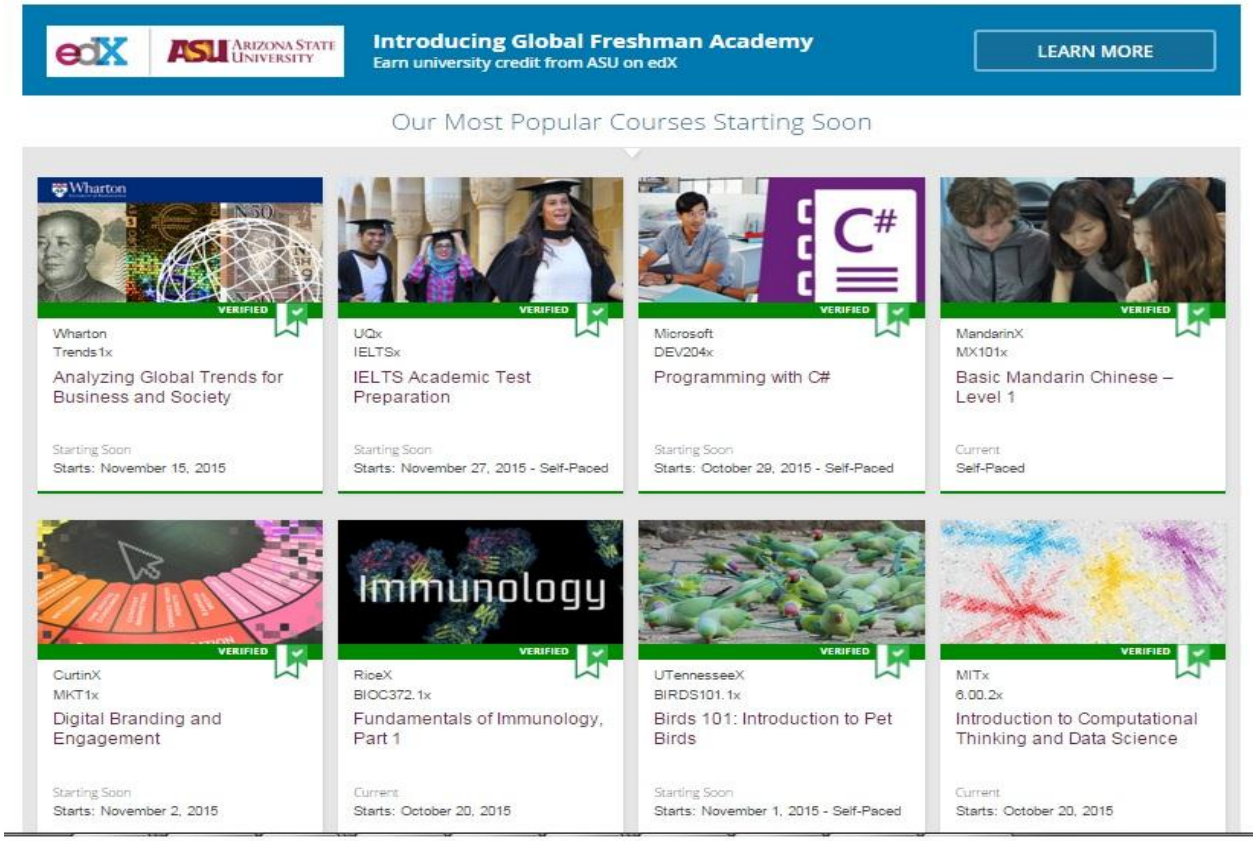


Fig.3. edx.org home page

Fig.4. portion of html code of the edx.org home page

As we can observe from the fig.2, there is a repetition of blocks of structure and the difference between each block is only the content. There is no relationship between content of block. From fig.3 we can see clearly that the structure of all the structures is the same in name of tag and attributes of tags (in fig.3 the attribute is “class”). We will locate these repeated structures based on DOM (Document object model) of the underlined web pages.

In order to find the different documents, we will:

- **Determine the repeating structure of the website** base on the attribute of the html elements like the “id” and “class” attribute. Siblings on the DOM(document object model) of a web page will be consider repeated strutures the have similar class and id attibute.

  Starting from the root of DOM, we will get all the chlidren on the root. Then find all the similar siblings based on their attributes and their names. Hence all the remaining children will be consider as new roots and we will traverse them to find similar children. This process will continue until the whole DOM of the page is traversed. This mechanism is depicted in the following algorithm.

The following algorithm is for determining repeated structures then cpmputing TF-IDF of words

Algorithm 1 find documents

INPUT: a node N of the DOM

Output: List D of documents

ND ← N

while ND is not empty

    C ← getchildren(ND_0)

    for all i ← 0 to |C|

        if C_i.attribut == C_{i+1}.attribut **then**

            D.insert(C_i)

        else

            ND.insert(C_i)

        end if

    end for

end while

e = CERT(N)

D.insert(e)

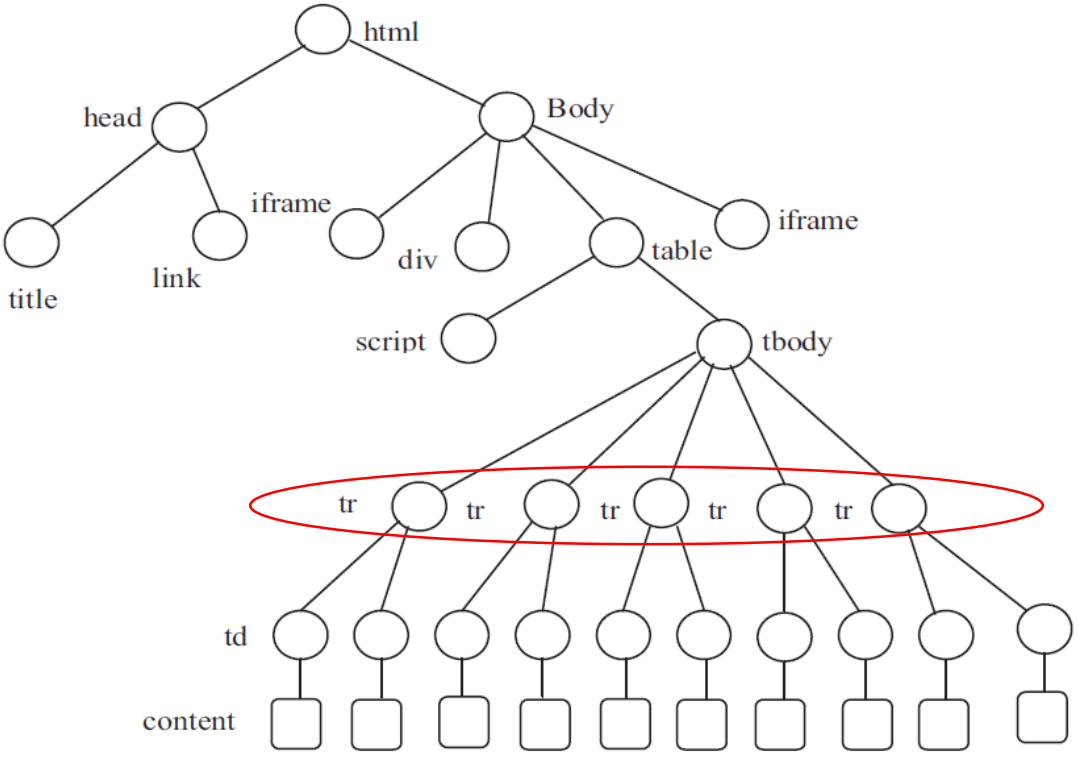


Fig5. DOM of a webpage

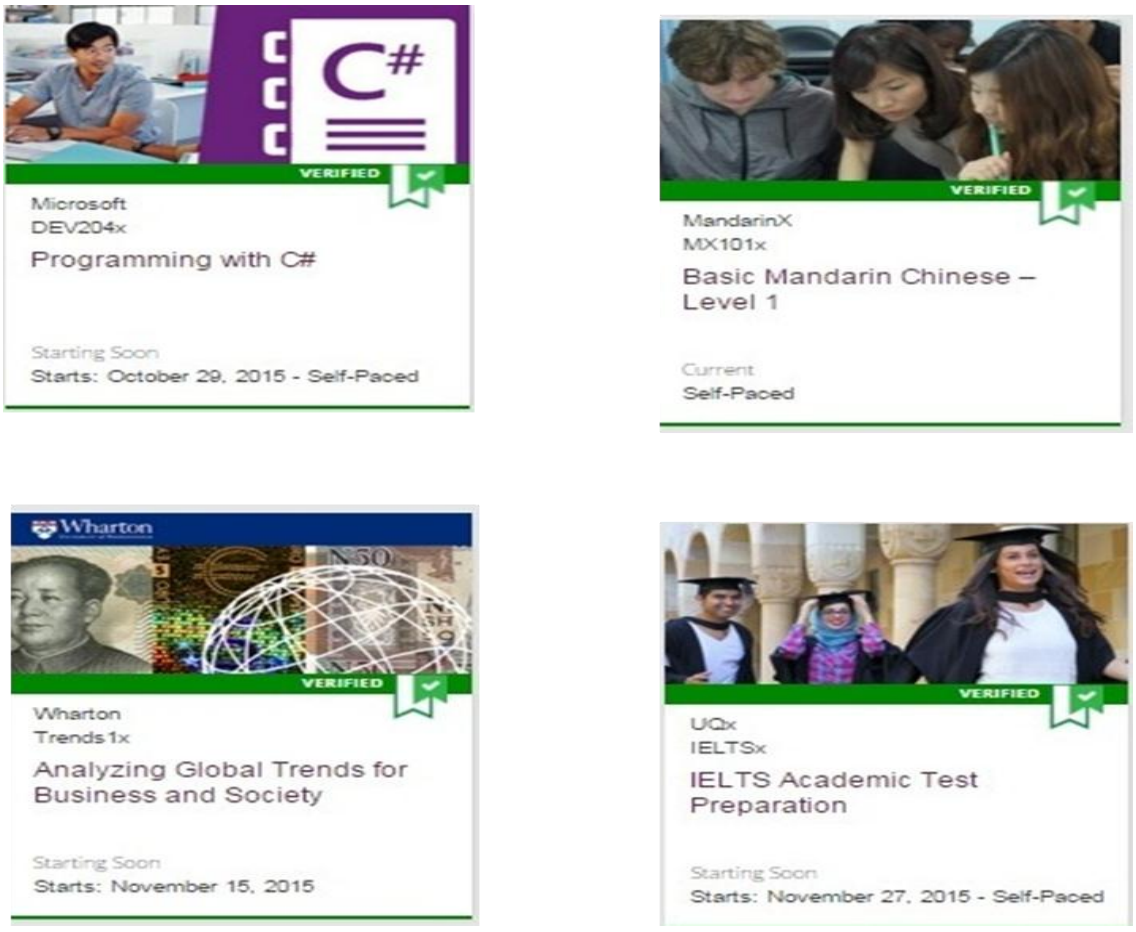


Fig6. Sections(documents) from edx.org

After finding the documents we can now find the keys(relevant) words from each documents by using term frequency and inverve document frequency(TF-IDF) concept to remove terms and stop words.

TF(term frequency) is used to compute the frequency of occurency of words in a particular documents. If we consider only the the TF mecanism then prepositions and articles will have highest frequency since they tend to appear the most in a single document. For that reason we will use also IDF(inverse document frequency), given a set of documents the IDF will give the highest mark to terms which are present in less documents and least mark to terms which appear in most of the documents.

Using TF-IDF, the crawler will support multiple languages, there will be no interaction of the human being to indicate the stop words.Having a list D of n document, the equation 1 shows how to compute the TF of a term t in a document d.

$$\mathrm{TF(d,t)}= \begin{cases} 0 & if\ n(d,t) = 0 \\ 1 + \log(1 + \log(n(d,t)) & otherwise \end{cases} \quad (1)$$

Where n(d,t) is the number of time term appears in the document d. And the IDF for a term t is compute following the equation 2 given a list D of n document.

$$IDF(t) = log\frac{1 + |D|}{|D\mathrm{t}|} \quad (2)$$

Where D_t is the number of documents in which the term t appears.

The following algorithm is used to computer the relevant keys words in each document found using the agorithm 1 then those words will be used to build the networds of words comsidered as the bag of words:

algorithm 2 compute words relevancy

INPUT: D documents

OUPUT: W list of words

N<------- |D|

for all i <----- 0 to |D|

T <----- getTerms (D_i)

for all j <----- 0 to |T|

W_{ij} (T_i)<----- IDF(T_j)*TF(D_i,T_j)

end for

sort(W_i)

end for

W_i will contain the list of words of document D_i sorted by relevancy order. Thus W contains words for all documents we have.

Having this list of relevant words, we can now build the networks of words in the next section.

3.2 Network of words (NW)

NW represents a directed weighted graph of words where the edge e_{ij} represents the number of documents in which the terms i and j appear simultaneously.

Let's n b the number of relevant keys terms we select from a document D_i.

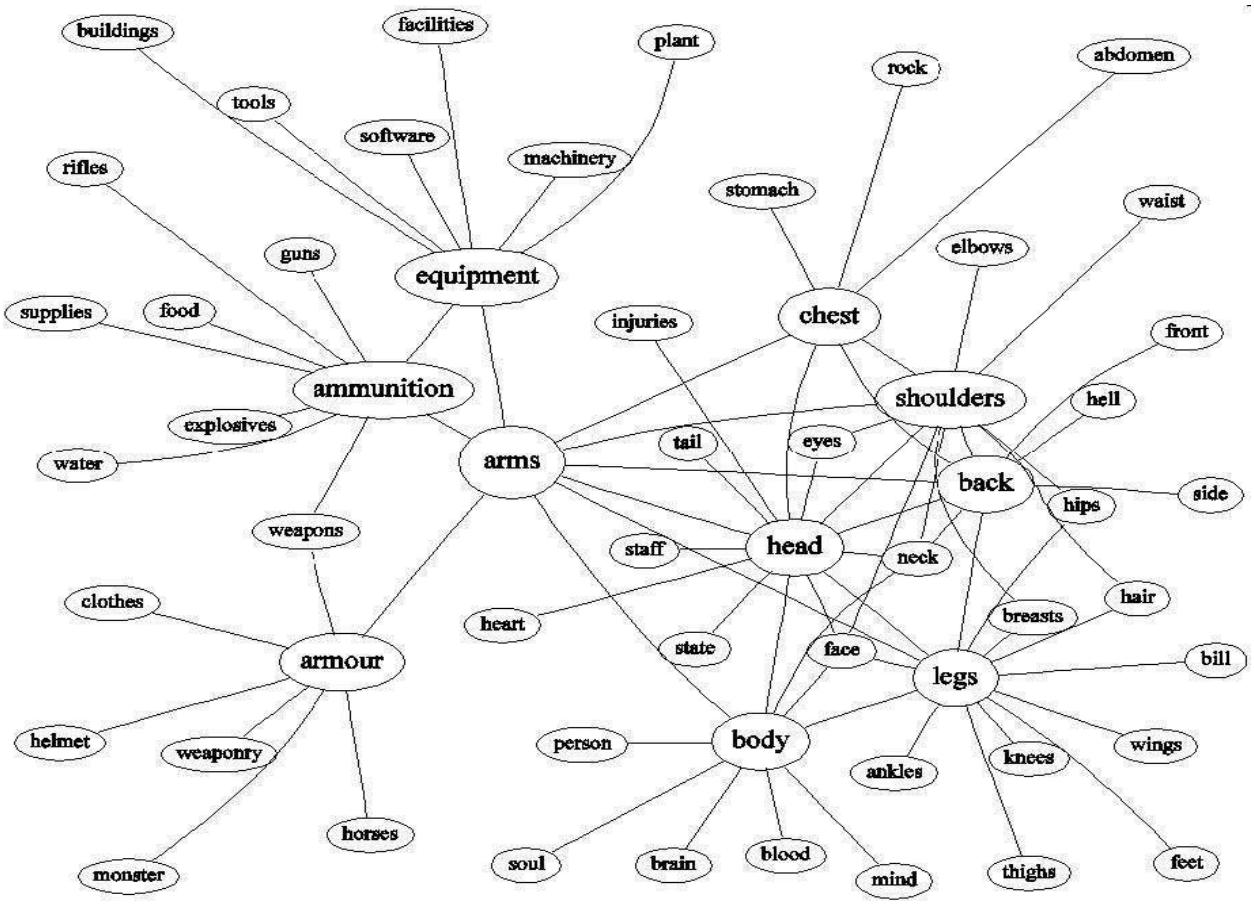


Fig.7 Network of words

In the network of words also called bag of words, we will determine d relevant terms highly connected to the n relevant terms of the document D_i, then use the n + d words to crawl the website. By this ways we can reduce the amount of results need to send to webserver in order to crawl the maximum amount of hidden web. Thus, with the networks of words, there is no longer need to all the keys words of the bag –of-words to crawl a particular website. By this way the amount of query to crawl a website will significantly reduce, hence the hidden crawling time will reduce significantly also.
From here we can see how this general hidden web crawler can adapt to a specific website, using only words related to the content of the websites to crawl the content of the websites. By update the NW by the n relevant words found, the system will be in constant learning process and the more we crawl the better the accuracy of the

connectivity between the words. The regular learning process of the network of words will make the selection of related terms more precise, more websites focus and more grouping of words. By this way the crawler will have only words related to the crawled website in order to get more data from the hidden web with less queries.

Also the network will allow to stay focus on the topics of the website by using only a small range of its data to crawl the given website.

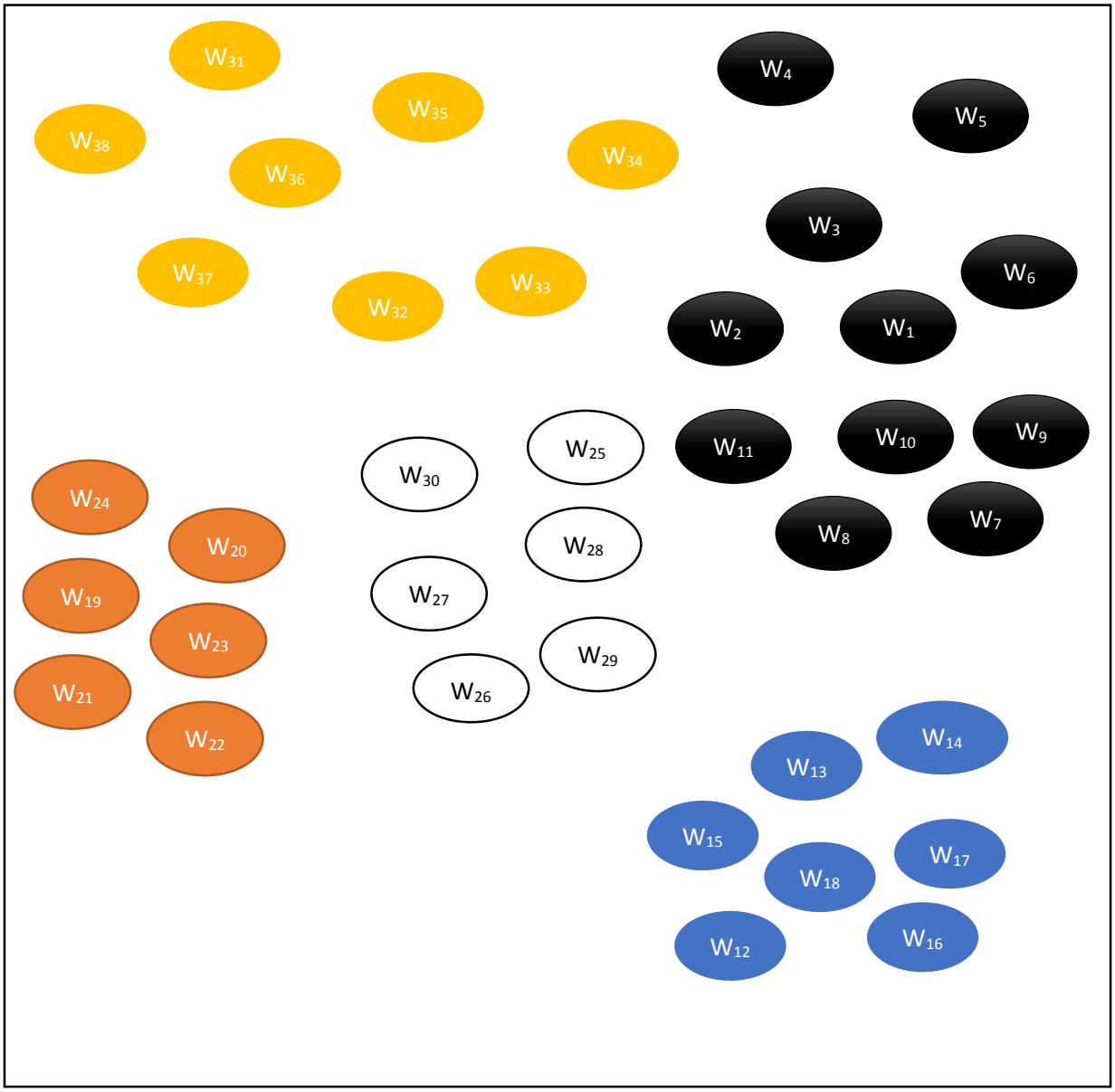


Fig.6 words grouping in the network of words

From the previous figure we can see that in the network of words, some words are more connected to others and we have mark this difference by colors. Thus more connected words have the same color, hence when crawling a website and I find a relevant words which are in the network of words I will just pick inside the latter the words having the same color. Here the web crawler will focus on data of the website and use only keys words related to it. In vertical hidden web crawler, the network of words will be very useful to connect words of the subcategories between them, thus reducing the number of words needed to crawl a website which contains only a subcategories of the focused topic.

Compare to others solutions which propose to take the whole content of the bag-of-words to only crawl a small subset of a topic, the network of words will reduce this by selecting only the words related to the specified words of the websites, thus avoiding irrelevant requests and reducing the crawling, thus improving the crawler performance. The following two algorithms will be on selection of related keys on the network of words and the update of the network.

Algorithm 3 select related words

INPUT: W (list of relevant words) and NW (network of words)

OUTPUT: WR (words related to W)

for all i <----- 0 to |W|

t <----- get_connected_terms(NW,W_i)

t <----- select_d_highly_connected(t)

WR <----- WR + t

End for

Algorithm 4 update NW

INPUT: W, NW, D

OUTPUT: NW

for all i <----- 0 to |W|-1

    for all j<----- i+1 to |W|

        e_{ij} <----- NW.get_weight(W_i, W_j)

        d <----- documents_where_appearing_simultaneously(D,W_i,W_j)

        e_{ij} <----- e_{ij} + d

        NW.setweight(W_i, W_j, e_{ij})

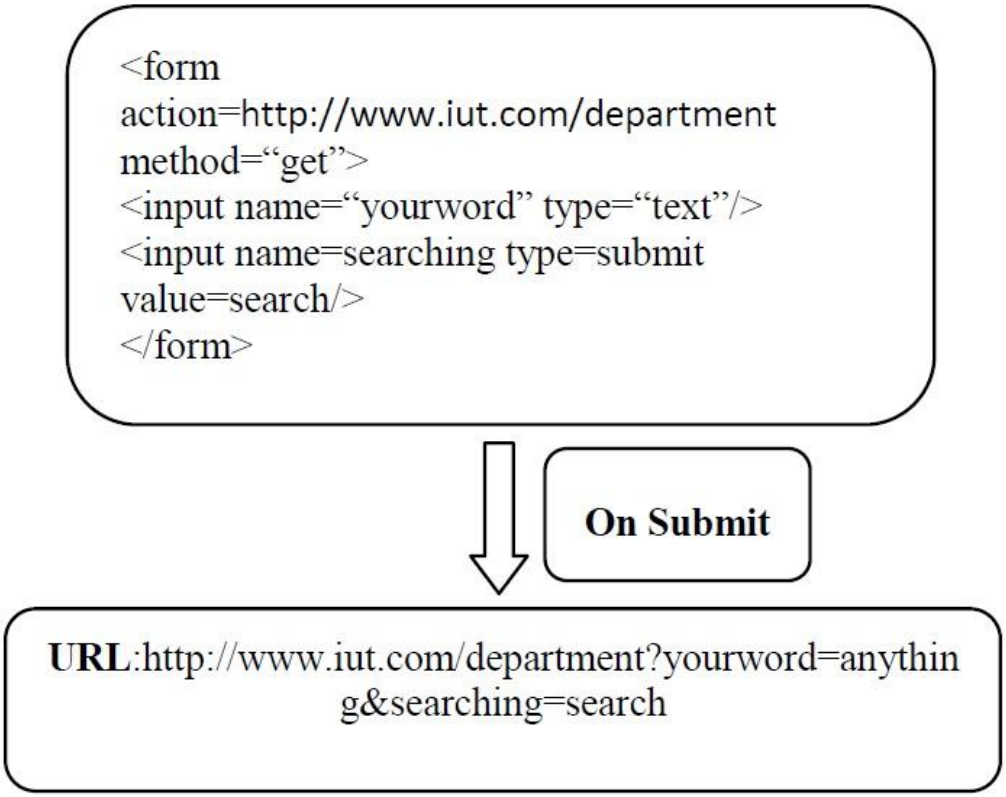
    End for

End for

The algorithm 4 (update) is the one which will keep the network of words in constant learning process. In fact whenever a new relevant word is found the word will be added to the network and will be connected with the words with which they appear in the same documents. If two words are found during crawling and are already in the network of words then their likelihood (weight of their edges) will be increase.

3.3 Sending request

In order to get the data from the database, we send a request to server. In this research work we only consider the get method. However the main purpose of the research work is not about using get or post method. In the bellow figure can see how the link will be formatted to be sent to the server.



3.4 Crawling algorithm

With the help of the previous algorithms, we can rewrite the overall working process of the crawler, and the following algorithm gives the process

```
Algorithm crawl
----------------------------------------------------------------------------------------------------------------------------
INPUT: seeds
OUTPUT: pool_url
pool_url <----- seeds
While not empty(pool_url)
        url <----- getNonVisitedUrl(pool_rul)
        content <----- getcontent(url)
        content <----- removeallscrpit(content)
        content<----- removestyle(content)
        pool_url <----- pool_url  + select_all_url(content)

        /*determine form and crawl the hidden content if it exists*/
        If web_form_in(content) then
                D <----- find_documents(contents)
                W <----- compute_word_relevancy(D)
                WR <-----select_related_words(NW,W)
                NW<----- updateNW(W,NW,D)
                W <----- sort(W,WR)
                i<-----0
                while not all_seen(W) and N > 0
                        i<-----i+1
                        term<----- get_highly_connected_term_notUsed(W)
                        content<-----sendRequest(term)
                        D<-----findDocuments(content)
                        CWi<----- compute_words_relevancy(D)
                        WR<----- select_related_words(NW,CWi)
```

NW<----- updateNW(CW_i,NW,D)

pool_url <----- pool_url + select_all_url(content)

CW_i<----- CW_i+WR

If nrequest==I **then**

I=0;

W<-----W+CW

end if

N<----- N-1

end while

end if

end while

Here N represent the maximum number of allowed requests to send to server to retrieve hidden content. And “n” represent the number of steps after which we will update the current words used to fill the forms. But this the will be a kind of priority between words. Thus the first words to be seen will have a relative priority related to the number of “n”.

The crawler starts by retrieving the content of seed links. Then it divides the content into section to form documents. Then the crawler computes the frequency of terms in each documents. Hence it computes the inverse document frequency. Then it computes the products of the inverse documents frequency times the term frequency. Then the crawler selects n most relevant terms in each documents based on TF*IDF. And it selected terms in the bag-of-words weightiest edge for each relevant data. These terms plus the relevant terms can be used to crawl the deep web. And this sub set of words will be update with new relevant words found during crawling plus the related words to the news keys words.

The Bag-of-words is update with new relevant terms found during crawling and for two relevant words found in the same document, their edge weight will be updated by an amount. By this way the bag-of-words will be learning regroup words of the same topic together, near each other in the network of words.

4. Experiment result and analysis

4.1 Development

The crawler which has been developed in our research work, starts from an initial URL and continuously crawl the internet. After downloading the pages, the crawler will segment the pages into different sections and we will consider these sections of the pages as documents as mentioned above in section 3. The html of each of the documents will be remove and tokens will be forms from words of the documents. Base on TF*IDF, the relevancy of each token will be find in the documents. The most relevant words in each document will be inserted inside the bag-of-words following the procedure stated in section 3.

We have used java programming as the programming language of development given numerous API available in this language to interact with internet. The development was done using Netbeans 7.4 and JDK 1.8. We have also use Jsoup, which a library to traverse DOM tree of web pages.

4.2 Experimental Setup

For the experiment we have considered the Open Directory Project (dmoz), a multilingual open content directory of World Wide Web links. The listings of the site are grouped into categories (which may include various sub-categories) depending on their topic. Dmoz indexes approximately 5 million links that cover a broad area of topics. Each link is accompanied with the site's title, a brief summary and its categorization. The links are searchable through a keyword search interface and dmoz enforces an upper limit on the number of returned results (10,000 results).

We have also used stackoverflow which is a multilingual website for questions and answers, for sharing information and group exchange. The web site is well known for its tremendous questions and answers content.
The experiment was conducted using a single personal computer with the following characteristics:

- Processor: Intel core2 duo 2.80 GHz
- 4 GB of RAM
- 1000 GB hard disk
- 500 Kbps bandwidth

In this experiment we have determined the number of the request needed to crawl the hidden web. We will measured the number of distinct links found in website base on the number of related words we selected from network of words. By this way we can see the effect of the network of words in reducing the number of requests needed to get a maximum amount of hidden web. We will also see the effect of the amount of output consider to see evaluate the number of distinct link we will get.

4.3 Result analysis

4.3.1 Related words effect evaluation

Figure.7 shows the result from our crawler using [28] as the initial links. By filling up the form found on [28] multiple times and keeping track of the distinct links in the output result, we were able to draw the below graphs depending on three parameters :

- d = number of related words
- m= number of relevant words from a document

- page = the numbers of documents

On figure, both graphs have the same m and page but different d. And the graph having the highest d(number of related words) detects more distinct links after the same iteration which is the intended output we were looking for. This result proves the usefulness of the network of words to select the most important words of the website.

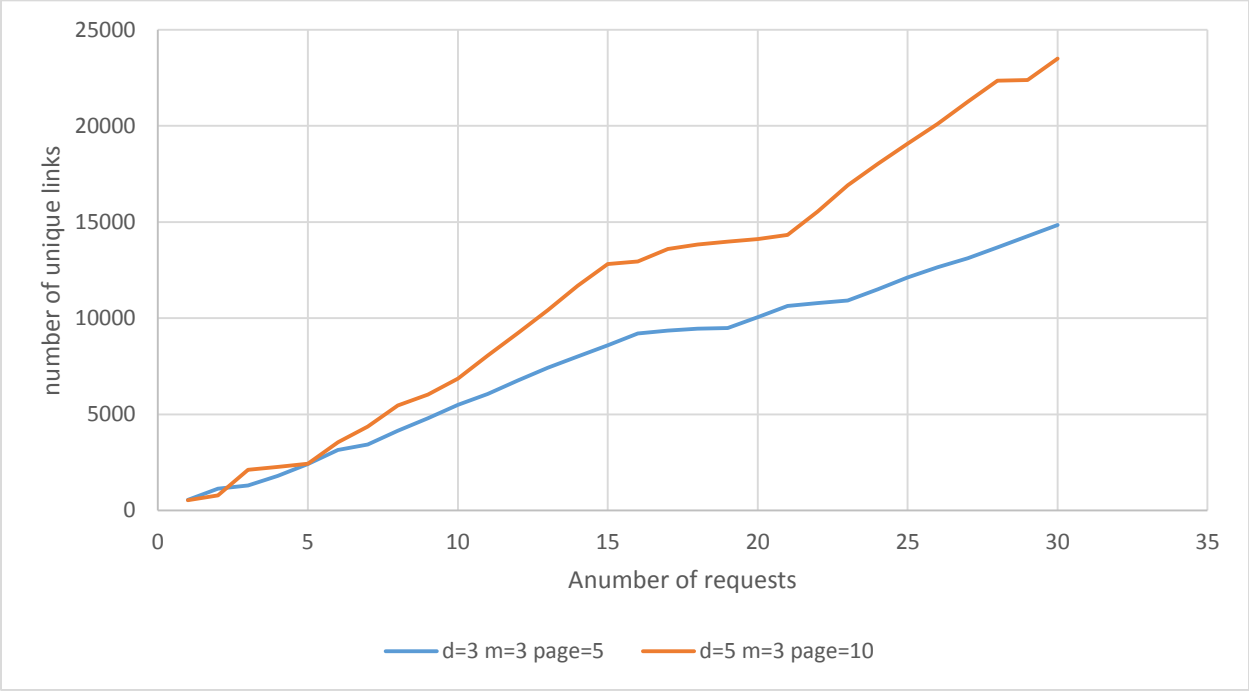


Fig.7 variation related words got from network of words

Let d = number of related words

m= number of relevant words from a document

page = 10*50= 500 documents

4.3.2 Amount of result: evaluation of effect

In figur2 8 we have the output of the crawlers based on the number of output documents taken into consideration. We can see that the graph having the higher numbers of documents tends to perform well at the beginning of the crawling better as we good we can see that both graphs may grow at same speed. Since the graph have the same d, we can say that the graphs shapes is only affected by the amount of documents page. However we have to notice that more documents means more times consuming for the crawler. Thus considering the a high amount of document to have a good result will only make the crawling slow.

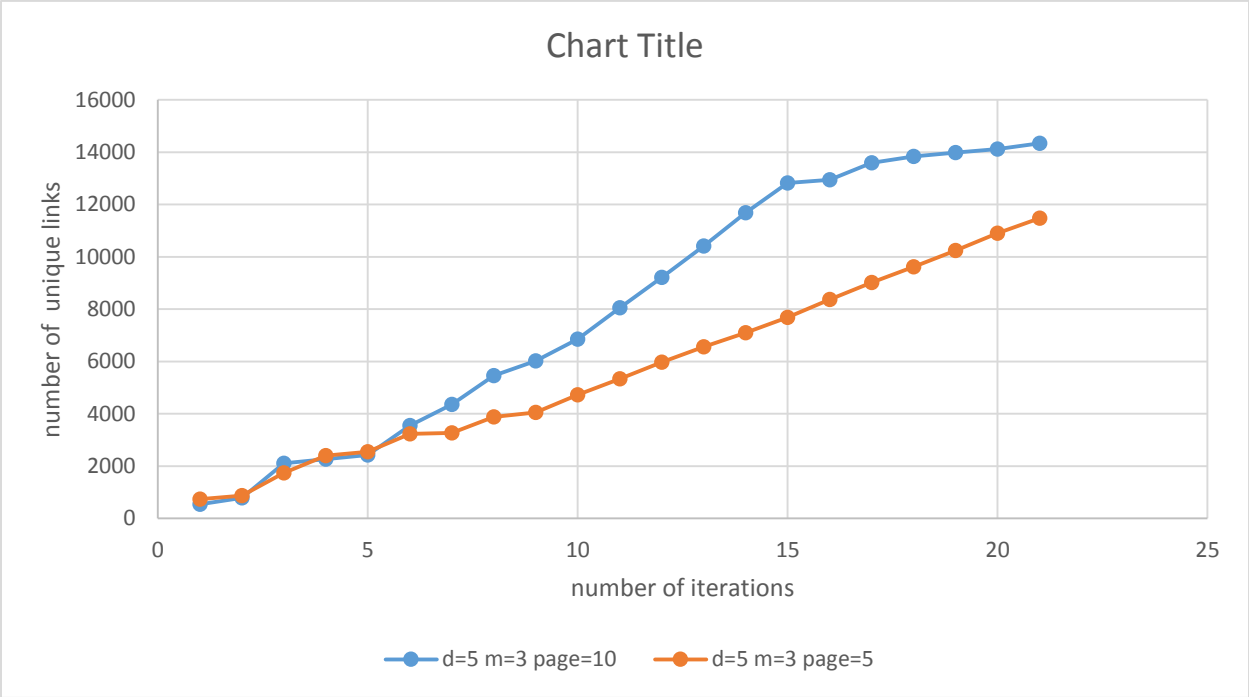


Fig.4 variation related words got from network of words

Let d = number of related words

m= number of relevant words from a document

number of documents = page *10

5. Conclusion and future work

Our research work was to retrieving that inside the databases with minimum resources need by modeling the bag-of-words into networks of words. We have successfully conducted the research and found satisfactory result from section 4, which show the usefulness of the bag-of-words organized as a network of words. From the result of section 4.3, we can say that if the crawler runs for a long time the networks of words will have a strong connection between words and words grouping will be more highlighted. This grouping of words will be very useful to control website only focusing on a specific content the bag-of-words instead of considering all the content when crawling or a method that does to take into account the connection among words.

Future experiment may be

- To integrate a databases inside the crawler to store the information learnt, this will allow the web crawlers to continue learning where it stops and it will not always start from the beginning.
- To evaluate the effect of bag-of-words inside topic focus crawling
- Evaluate the hidden web on the client side

References

[1] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang, "Accessing the deep web," *Commun. ACM*, vol. 50, no. 5, pp. 94–101, May 2007. [Online] Available: http://doi.acm.org/10.1145/1230819.1241670

[2] O. A. McBryan, "Genvl and wwww: Tools for taming the web," *In Proceedings of the First International World Wide Web Conference*, 1994, pp. 79–90

[3] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proceedings of the seventh international conference on World Wide Web 7*, ser. WWW7. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 1998, pp. 107–117. [Online]. Available: http://dl.acm.org/citation.cfm?id=297805.297827

[4] L. Barbosa and J. Freire, "Siphoning hidden-web data through keywordbased interfaces," in *In SBBD*, 2004, pp. 309–321

[5] S. Raghavan and H. Garcia-Molina, "Crawling the hidden web," in *Proceedings of the 27th International Conference on Very Large Data Bases*, ser. VLDB '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 129–138. [Online]. Available:http://dl.acm.org/citation.cfm?id=645927.672025

[6] M.Soulemane, M. Rafiuzzaman and H. Mahmud "Crawling the Hidden Web: An Approach to Dynamic Web Indexing" *International Journal of Computer Applications (0975 – 8887) Volume 55– No.1, October 2012*

[7] YU Hai-tao1,2, GUO Jian-yi, YU Zheng-tao, XIAN Yan-tuan, YAN Xin A Novel Method for Extracting Entity Data from Deep Web Precisely

[8] A. Ntoulas, "Downloading textual hidden web content through keyword queries," in *In JCDL*, 2005, pp. 100–109.

[9] S. W. Liddle, D. W. Embley, D. T. Scott, and S. H. Yau1, "Extracting Data behind Web Forms," *Lecture Notes in Computer Science*, vol. 2784, pp. 402–413, Jan. 2003.

[10] *Zheng He, Hangzai Luo, Jianping Fan, Xiao Liu* EXTRACTING THE SEMANTIC CONTENT OF WEB PAGES VIA REPEATED STRUCTURES

[11] S. Lawrence and C. L. Giles, "Searching the world wide web," *SCIENCE*, vol. 280, no. 5360, pp. 98–100, 1998.

[12] M. K. Bergman, "The deep web: Surfacing hidden value," September 2001. [Online]. Available: http://www.brightplanet.com/pdf/deepwebwhitepaper.pdf

[13] Ziv Bar-Yossef and Sridhar Rajagopala, "Template detection via data mining and its applications," in *International Conference on World Wide Web*, 2002.

[14] Shian-Hua Lin and Jan-Ming Ho, "Discovering informative content blocks from web documents," in *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[15] Lan Yi, Bing Liu, and Xiaoli Li, “Eliminating noisy information in web pages for data mining,” in *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.
[16] Sandip Debnath, Prasenjit Mitra, and C. Lee Giles, “Automatic extraction of informative blocks from webpages,” in *ACM SIGAPP SAC*, 2005.
[17] Thomas Gottron, “Content code blurring: A new approach to content extraction,” in *InternationalWorkshop on Text Information Retrieval*, 2008, pp. 29–33.
[18] T Weninger, WH Hsu, and J Han, “Cetr: Content extraction via tag ratios,” in *International Conference on World Wide Web*, 2010.

[19] S. Lawrence and C. L. Giles, “Searching the world wide web,” SCIENCE, vol. 280, no. 5360, pp. 98–100, 1998.
[20] M. K. Bergman, “The deep web: Surfacing hidden value,” September 2001. [Online]. Available: http://www.brightplanet.com/pdf/deepwebwhitepaper.pdf
[21] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang, “Accessing the deep web,” Commun. ACM, vol. 50, no. 5, pp. 94–101, May 2007. [Online]. Available: http://doi.acm.org/10.1145/1230819.1241670
[22] Seyed M. Mirtaheri, Mustafa Emre Dinc¸t¨urk, Salman Hooshmand, Gregor V. Bochmann, Guy-Vincent Jourdan, ”A Brief History of Web Crawlers”, arXiv:1405.0749v1 [cs.IR] 5 May 2014
[23] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, “State of the art: Automated black-box web application vulnerability testing,” in Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, 2010, pp. 332–345.
[24] A. Doup´e, M. Cova, and G. Vigna, “Why johnny cant pentest: An analysis of black-box web vulnerability scanners,” in Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2010, pp. 111–131.
[25] J. Marini, Document Object Model, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 2002.
[26] Panagiotis Liakos, Alexandros Ntoulas,Alexandros Labrinidis, Alex Delis,” Focused crawling for the hidden web” 16 April 2015 Springer Science+Business Media New York
[27] Schonhofen, P.: Identifying document topics using the wikipedia category network. In: Proceedings of the 2006 IEEE/WIC/ACMInternational Conference onWeb Intelligence (WI), pp. 456–462, Hong Kong (2006)
[28] http://stackoverflow.com/