ANALYSIS OF TURKEY'S VISIBILITY ON GLOBAL INTERNET

SERTAÇ ORALALP

MAY 2010

ANALYSIS OF TURKEY'S VISIBILITY ON GLOBAL INTERNET

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERTAÇ ORALALP

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MAY 2010

Approval of the thesis:

ANALYSIS OF TURKEY'S VISIBILITY ON GLOBAL INTERNET

submitted by **SERTAÇ ORALALP** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Dr. Attila Özgit _____
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Assoc. Prof. Dr. Kürşat Çağıltay _____
Computer Education and Instructional Technology Dept., METU

Dr. Attila Özgit _____
Computer Engineering Dept., METU

Dr. Ali Arifoğlu _____
Informatics Institute, METU

Dr. Cevat Şener _____
Computer Engineering Dept., METU

Dr. Onur Tolga Şehitoğlu _____
Computer Engineering Dept., METU

Date: 5th May, 2010

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Sertaç ORALALP

Signature:

ABSTRACT

ANALYSIS OF TURKEY'S VISIBILITY ON GLOBAL INTERNET

Oralalp, Sertaç

M.S., Department of Computer Engineering

Supervisor: Dr. Attila Özgit

May 2010, 166 pages

In this study, Turkey's Internet visibility will be analyzed based on data to be collected from multiple different resources (such as; Google, Yahoo, Altavista, Bing and AOL). Analysis work will involve inspection of DNS queries, Web crawling and some other similar techniques. Our goal is to investigate global Internet and find webs that has common pattern of representing Internet visibility of Turkey and compare their characteristics with other webs' on the world and discover their similarities and differences.

Keywords: Turkey's Internet visibility, web crawling, ranking, positioning

ÖZ

KÜRESEL INTERNET ÜZERİNDE TÜRKİYE'NİN GÖRÜNÜRLÜĞÜNÜN ARAŞTIRILMASI

Oralalp, Sertaç

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Dr. Attila Özgit

Mayıs 2010, 166 sayfa

Bu çalışmada, Türkiye'nin Internet üzerindeki görünürlüğü çok farklı kaynaklardan toplanan verilere göre analiz edilecektir. Analiz; DNS sorgulama, Web tarama ve diğer bazı benzer tekniklerin kullanımını içerir. Amacımız, Türkiye'nin Internet görünürlüğünü temsil eden, ortak özelliklere sahip web siteleri ile bunların dışında kalan küresel Internet'ten bulduğumuz diğer web sitelerinin benzerlikleri ve farklılıklarını keşfetmektir.

Anahtar Kelimeler: Türkiye’nin Internet’te görünürlüğü, web tarama, sıralama, konumlandırma

ACKNOWLEDGEMENTS

I would like to thank people who supported me during preparing of this thesis. Specifically, I would like to thank to my supervisor Dr. Attila Özgit, for giving me valuable support with his advice and criticism during my study.

I also want to thank all the people I had taken their support during the last year. I had fruitful support and collaboration with the System Administrators Can Eroğul and Gökdeniz Karadağ.

Maybe the most valuable part of the support comes from my love, Sahure who supported me to dedicate all my free time (nights and weekends) for my study; and also my son Mehmetcan, who made valuable contributions with his comments.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Visibility is a term, which is used in various fields, but generally used in meteorology. The term is defined as; "In meteorology, visibility is a measure of the distance at which an object or light can be clearly discerned. Visibility affects all forms of traffic: roads, sailing and aviation. Meteorological visibility refers to transparency of air: in dark, meteorological visibility is still the same as in daylight for the same air." [1]

In the World Wide Web, a website is called visible when it has pre-ranked position at the web page list of any search engine. The pre-ranked position of any website at the web page list creates preference for users to reach ease. Like visibility in meteorology, visibility in Internet is a measure of page rank at which a website can clearly be seen to be reached. For example, if the traffic density is used as an indicator of Internet Visibility of websites, then it is possible to conclude, when the traffic density of any website increases, Internet Visibility of that website gets higher. In other words, the website is visible if it has higher traffic density. So, many indicators may be used to determine the visibility of websites and countries where the websites registered to.

Two different resources were investigated related with Internet Visibility of countries and their results were compared:

First method of determining a country's Internet Visibility is to compare traffic rank scores of web sites (from Alexa.com) registered in the name of a country within a sample (for example, in the first million) of web

sites having highest scores, with the traffic rank scores of other countries. Positions of websites globally were used to determine countries' Internet visibility.

Webometrics.com ranking is a good indicator of impact and prestige of universities and was utilized to determine country's Internet Visibility as second method in our research.

The results obtained in this study, related with the internet visibility of countries, has been obtained by exploring rating data according to various indicators of websites registered countries in the ranking list.

1.1 Description of Research Subject

Turkey's Internet Visibility was analyzed based on the data to be collected from Webometrics.com and Alexa.com. Both resources was utilized as statistical sources of this research:

Traffic rank information of websites in top 1.000.000 list found at Alexa.com was combined together to allow country base comparison.

Web positioning of Colleges and Universities were used as an indicator to compare countries' visibility. Webometrics.com presents valuable data about ranking of those institutions and countries.

During the inspection of visibility of websites and countries, some valuable derivative results would be possible to drive crawling Turkish websites.

Three different approaches were used to create the keyword list to achieve blind search to the websites having Turkish content:

determining keywords with heuristic approach as explained in Chapter 2,

from ongoing researches (Turkish Corpus Project at METU and Mersin University),

published research of Dr. İlyas GÖZ [2].

These were used to inquire websites found at the web page list of specific search engines (such as; Google, Yahoo, Altavista, Bing and Aol) then compare results of inquiries based on set of selected keywords.

In addition to the above studies conducted, following information sources were also used.

The list of Turkish websites found from Alexa.com was observed to determine percentage of websites at the first ten thousand that supports in multiple languages.

Turkish websites found from Alexa.com were inquired to create word frequency list from Internet.

METU DNS server is in our focus and DNS log records were inspected to understand pattern of inquiries if exists.

1.2 Specific Aims of the Research

In this thesis, Turkey's Internet Visibility and general characteristics of Turkish websites were analyzed through multiple channels, which were defined at previous article.

Using the database of Alexa.com, an algorithm was experienced for determining the position of countries' Internet visibility at global Internet.

In addition to the specific goal of the thesis, web sites have been crawled to collect information related to their content and compare results found with three different search algorithms as explained at 3.2.3 in Chapter 3.

1.3 Size of Visible Web Space

In April 2007 survey 113,658,468 web sites were responded to the request of Netcraft.com. Later in April 2010, almost twice of previous figure, 205.368.103 web sites were responded to the request [3].

There were 70,392,567 (active & inactive) websites indexed by Yahoo as of August 2005, while Google announced on 7/25/2008 that it had indexed over 1 trillion unique URL's.

Every day, more and more Internet sites are being created. This means that number of active domain names and web contents will change fairly rapidly over short periods of time. However, global Internet has some common characteristics in terms of visibility, as explained in the following paragraphs.

For example, Google's ranking methodology is so called PageRank™, which uses 500 million variables and 2 billion terms. PageRank uses linked network structure as a regulatory tool. As usual, Google evaluates connections established from page A to page B, a "vote" as comments. According to the Google's ranking methodology, a web page's importance is determined by vote.

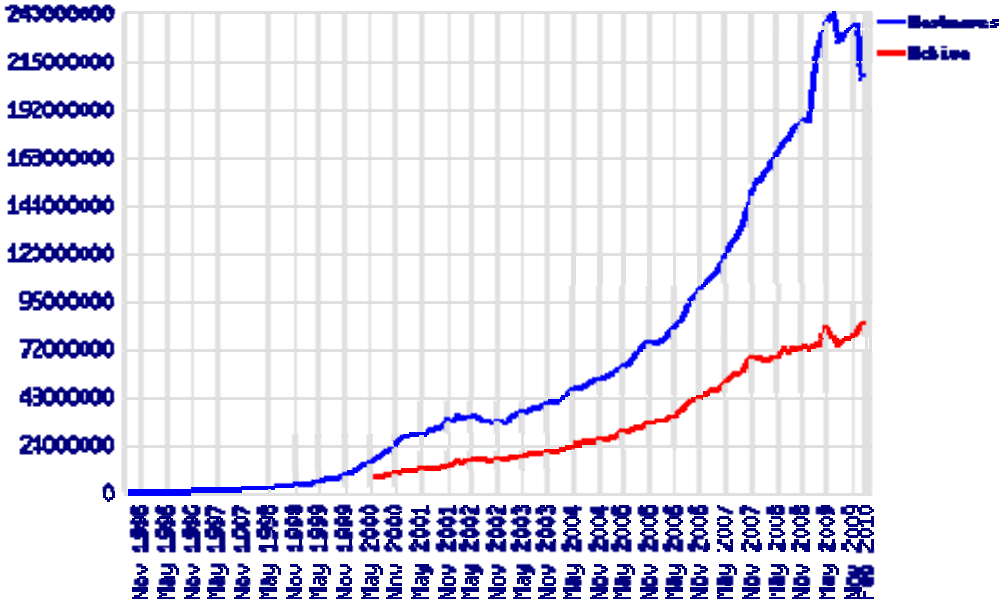

Figure 1: Total sites across all domains November 1995–February 2010

Web sites are classified according to their content and ranked in specific categories by different ranking methodologies used by specific search engines, such as; Google, Yahoo, Altavista, Aol, Bing etc.

1.4 Outline of the Thesis

The organization of the thesis is as follows:

In Chapter 2, a survey of the related work founded on literature search is given. The algorithms described in this chapter were utilized to create our algorithm.

Chapter 3 starts with the detailed problem definition then our "Alternative Algorithm" was introduced in detail being used to determine the visibility of countries based on the data driven from Alexa.com.

In Chapter 4, the logic our software presented and how the process done to reach the results.

Chapter 5, the results of the study are given in detail.

Finally, Chapter 6 summarizes the thesis study and provides the conclusions. Possible improvement ideas to the proposed algorithms are also given in the future work section.

CHAPTER 2

RELATED WORK

2.1 The Literature Search Related to WEB Visibility

To prove the originality and value of our contribution, we reviewed of the existing literature in the area of our research.

Visibility is a term, which is used in many areas [4], such as; quality or fact or degree of being visible; perceptible by the eye or obvious to the eye ("low visibility caused by fog"); degree of exposure to public notice ("that candidate does not have sufficient visibility to win an election"); capability of providing a clear unobstructed view,("a wind shield with good visibility").

Visibility is a mathematical abstraction of the real-life notion of visibility[5].

Visibility develops Enterprise Resource Planning (ERP) software, which targets small to mid-sized Engineer-to-Order (ETO), Make-to-Order (MTO), Mixed-Mode and Project-Oriented Manufacturing companies [6].

In meteorology, visibility is a measure of the distance at which an object or light can be clearly discerned. It is reported within surface weather observations and METAR code either in meters or statute miles, depending upon the country [1].

The condition of being visible; the degree to which things may be seen[7].

Visible - capable of being seen; or open to easy view; "a visible object"; "visible stars"; "mountains visible in the distance"; "a visible change of ... visible(a): present and easily available; "the cash on hand is adequate for current needs"; "emergency police were on hand in case of trouble"; "a visible supply"; "visible resources" [8].

Visible is billed as a not-for-profit, free, quarterly magazine dedicated to the history of the lesbian, gay, bisexual, and transgender community[9].

Visible was an album by Canadian progressive rock band CANO, released in 1985[10].

Visible - Able to be seen [11].

In the context of this ranking, the term refers to link visibility: The number of external inlinks received by an institutional domain. The most used syntax for this request in search engines is repositories [12].

The greatest distance in a given direction at which it is just possible to see and identify with the naked eye [13].

The distance a person can see with normal vision, and is measured in miles [14].

“Internet visibility” and “Web site visibility” have many links and are major consultancy subjects to ensure the visibility of web sites by the specific search engines. On the other hand, Internet Visibility was used as synonyms with web positioning and being noticed.

The keyword that we used specifically for literature search: "Turkey's Internet visibility", did not give any specific result and this subject has not been studied previously either for Turkey or for other countries. But, statistical data related with ranking and positioning of websites and countries exist on the Internet. The data obtained from Alexa.com and Webometrics.com has this nature and helped our argument to determine the visibility of the countries.

Another literature survey has been achieved to determine Turkish Keywords those would be used to recognize web sites having Turkish content. Three researches were determined;

Turkish Corpus Project at the Middle East Technical University,

Turkish Corpus Project at the Mersin University,

The publication of the Turkish Language Association; "Turkish dictionary of the written word frequency" was done by Dr. İlyas Göz [2].

The list of Turkish words with frequently used was requested from first two projects' leaders (Assist. Prof. Dr. Bilge Say, ODTÜ and Prof. Dr. Mustafa Aksan, Mersin University). Although their encouraging approaches, their projects were ongoing and the time was early to share the list of frequently used words on the Internet.

The publication of the Turkish Language Society's Review of Dr. Göz's "Written word frequency of Turkish"[2] has synonym of words and additional-root separation process have been passed through and more healthy to be used, but it has some drawbacks; first, the list itself does not pass through some morphological operations, second, compilation of texts of the early 2000s to participate - so do not reflect present-day. But, anyhow we used the frequency list of Dr. İlyas Göz[2], aware of the disadvantages of his work.

However, research of Dr. İlyas Göz[2] was determined to utilize for the thesis then most frequently used 50 Turkish words were selected from his

book. On the other hand, inspecting a series of Turkish websites, a list of words (Appendix B) generated with educated guess and then 19 most frequent words were selected from that list to use determining Turkish websites from search engines before crawling process.

With these results, imagination of our contribution to the literature with our study has been exciting us even more.

2.2 The Data Collected from DNS Server

The Domain Name System (DNS) is a hierarchical naming system for computers, services, or any resource connected to the Internet or a private network. The Domain Name System makes it possible to assign domain names to groups of Internet users in a meaningful way, independent of each user's physical location[15].

Computers send inquiry request to the local DNS server. The local DNS server finds this address and responds to the request of the computers. If the local DNS server cannot find the requested address in its own data, it transfers request to a top-level official root DNS machine. Root DNS server forwards the remote address of the DNS, which is the lookup server that is responsible from the desired address to the local DNS.

In this context, inquiry requests sent to METU DNS were analyzed to discover the nature of the requests.

2.3 State of The Art in Web Crawling Algorithms

Web crawling techniques have different algorithms depending on where they are used as well as their performance and complexity.

Universal search engines distribute the crawling process across the users, queries or even client computers to explore context and use them commercially.

Web searching is a difficult task, no matter which Web crawling algorithm is used and how sophisticated the ranking algorithm is, the inquiry will not be successful if static page has not been indexed by the search engine. "Given the current size of the Web, even large search engines cover only a portion of the publicly-available Internet; a study by Lawrence and Giles showed that no search engine indexes more than 16% of the Web in 1999."[16]. There are approximately 20 to 40 billion "static" pages [based on the estimation of Cyveillance's research at 2000] of visible Webs (means the Web in search of pages have been indexed). That volume needs indexing more than 10 billion pages for the largest search engine[17].

Search engines could not be successful to keep up the growth of newly announced and updated frequency of Webs, even the capacity and capability of hardware-software and network bandwidth resources. Cho and Garcia-Molina's research in 2000, based on a capacity of 720,000pages being daily crawled; they found that 50% of webs that were crawled within 50 days period, were changed. They also reported that the rate of change varies according to domains, e.g. webs having .com domain were changed within 11 days while sites of .gov domain were changed within a period of four months [18].

The main web crawling algorithms use Web's hyperlinked structure to trace new pages. When a page is captured, their inward and outward links may be added to lists of unvisited page addresses. Adaptive crawlers use machine-learning techniques to guide their search.

The so-called **link analysis** means that a page's content and linkage to other pages could be used as an indicator to estimate its relevance with

respect to user queries, has been adapted to machine learning techniques [Kleinberg and Lawrence 2001][19].

The Best Known Crawling Algorithms [19] are briefly introduced in the following paragraphs.

Breadth-First is probably the first algorithm at the crawling history. It has a simple strategy and was announced at 1994. During the crawling, it keeps track of links and does not use any knowledge about the topic.

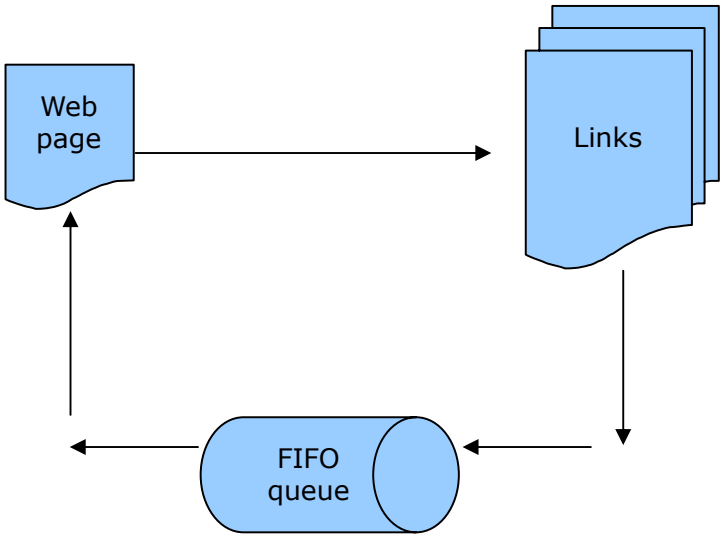

Figure 2: Algorithm of Breadth-First crawler

Best-First algorithm is for selecting one of the best link that satisfies some of the estimation criteria from given frontier of links. Similarity between the source page (p) and the topic (keywords) is computed to find the relevance of the pages pointed by p that would be the best estimate for crawling.

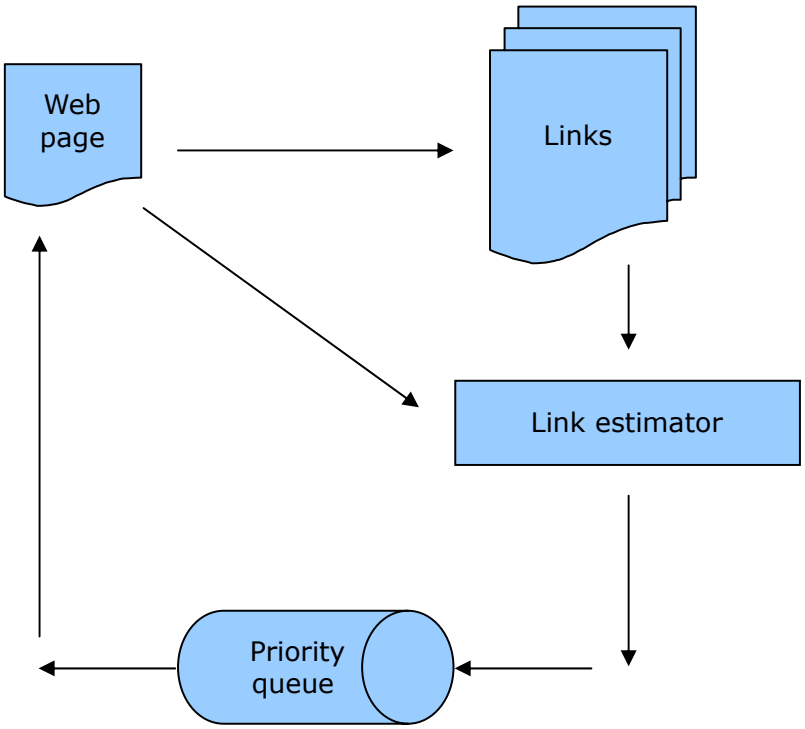

Figure 3: The Best-First crawler algorithm

Shark-Search is similar to Fish-Search, however it uses a continuous valued function for measuring relevance and having potential scores for the links, which is influenced by anchor text, text surrounding the links, maximum depth and inherited score from pages pointing to the page containing the links. Shark-Search is a sophisticated version of Best-First algorithm.

InfoSpiders is an evolutionary approach, which leads its adaptive agents to search for pages relevant to topic (contains list of keywords) and neural nets to decide on the link to be followed. Each agent counts the frequency of the keywords occurring around/near the link to be visited.

Experimental results have proved that Best-First algorithm was an effective crawler, especially in the short run.

InfoSpiders algorithm is more efficient for large frontier sizes thus InfoSpiders achieves the best performance/cost ratio. We used an adaptive model of InfoSpiders algorithms in our crawling algorithm.

CHAPTER 3

DATA COLLECTION

3.1 Data Collection Goals

We aimed at finding answers to the following basic questions through our research:

Will alternative methods of research provide us similar ranking values related with Turkey's Internet visibility?

Web crawling with different sets of keywords will bring out diverse results related with the general characteristics of Turkish websites?

What are the common features of Turkish websites founded from three different algorithms?

In what extend Internet word frequency list founded in our research will overlap to the written word frequency list of other studies.

3.2 Methodology for Data Collection

3.2.1 Data from Alexa.com

The list of top 1.000.000 websites at Alexa.com calculates global rank of websites tracking monthly base. Alexa.com tools keep track of web sites where they are installed, then calculates scores with the highest combination of indicators; visitors and page views. Country short code, rank in country base and sites linking in data were captured from Alexa.com to generate our statistical knowledgebase combining with global rank and name of websites downloaded from Alexa.com. New database was ready to generate countries ranking list that helped us to

determine the visibility list of countries. [Appendix E - List of Alexa_get_so.py]

3.2.2 Search Webometrics.com to Utilize Statistical Figures

The "Webometrics Ranking of World Universities" is an initiative of the Cybermetrics Lab, a research group ranking global educational institutions using web presence measures of their activity and visibility in which summarizes the global performance of the University. Cybermetrics Lab. ranks the institutions using; "size (Number of pages recovered from Google, Yahoo, Live Search and Exalead), visibility (The total number of unique external links received –inlinks- by a site can be only confidently obtained from Yahoo Search.), rich files (number of academic and publication activities and considering the volume of the different file formats, the following were selected: Adobe Acrobat - .pdf, Adobe PostScript - .ps, Microsoft Word - .doc and Microsoft Powerpoint - .ppt) and scholars (Google Scholar provides the number of papers and citations for each academic domain) as design and weighting of indicators." [20]. Each of the indicator has different weight calculating of the total ranking value, such as; size (web pages) %20, visibility (external links) %50, rich files %15 and scholar %15 as shown at Table 30 in Appendix F.

In connection with the ranking of Cybermetrics Lab., QS SAFE National System has been designed a model (the Country Scoreboard) based on the evaluation of the countries' higher education system according to the presence of their Universities in the Top 500 of the Ranking Web. "Four normalized indicators are used with equal weighting as follows, system (Number of universities in the Top 500 in the given country, divided by the mean position of those institutions), access (A score built according to ranks (5 points for a university in the top 100, 4 points for 101-200, 3 points for 201-300, 2 for 301-400 and 1 for 401-500) divided by the population size (root of the population in thousands) of the country (World Bank, 2007)), flagship (A normalized score (100 for positions 1-20, 96 for 21-40, and so on) based on the leading university rank for

countries with institutions among the Top 500) and Economic (Same score as the access defined before but divided by the GDP (PPP) per capita for the country in question (World Bank, 2007))."[21] Finally, Country's Scoreboard value was found by taking the mean of those four indicators as shown at Table 30 in Appendix F.

3.2.3 Web Crawling

The World Wide Web is typically browsed by scripts named as web crawlers in a methodological, and an automated manner (many call the art of finding as specific information from Internet). Web crawlers are named as; ants, automatic indexers, bots, worms or web spider and web robot [22].

Mission of our script is to visit the selected web pages, which are supposed to have Turkish content, like a web spider using web crawling technique. Our web spider has to visit only web pages, which has ranked by the particular search engines according to our three alternative criteria that have already been defined. These are:

50 Turkish keywords used most frequently in written Turkish (determined by Dr. İlyas Göz),

19 Turkish words were selected by intuitively (in Chapter 2),

The Turkish websites driven from Alexa.com were visited and searched for the criteria (i). Results are given in Chapter 5.

Our web spider scripts visit the web pages which had been indexed and categorized by the specific search engines with web crawling technique and it is designed to check and collect information related with the structural characteristics of the web pages, such as; width of website, number of internal and external addresses, HREF links, images and forms, size of web pages and the nation where web site is registered.

The web spider inspects content of body of websites and addresses internal html pages to record Turkish keywords, which have highest statistical probability of finding on the World Wide Web.

The web spider keeps track of information related to site visit, such as; time-stamp of visit, host name where the web spider started crawling, error code if encountered, depth level of search, how long our web spider stays in the web page and the search engine is used for data collection. While web spider visits the web pages, whose error responses were recorded to understand the behavior of those webs if needed.

After crawling web pages having Turkish content and compare them with others having Turkish content and try to find similarities (common features) and differences of those which are categorized in generic top-level domain, e.g. .com, .edu, .net, .org, etc.).

Dissimilarities of Turkish and non-Turkish web sites were compared based on strengths and weaknesses of the parties.

Another output is the word frequency list driven from Turkish websites found from Alexa.com. The first 50 most frequent words of the output were compared how overlapping with the list of Dr. İlyas Göz.

3.2.4 The Data Collected from DNS Server at METU

DNS server at METU (ns1.metu.edu.tr is known as the secondary DNS server) is accepted as an authorized information source for ".tr" addresses at domestically and internationally. METU DNS server collects requests of local DNS servers. METU DNS log records were inspected to determine direction of the inquiries and any significant data if exists.

3.2.5 Word Frequency List for WEB Sites at Alexa.com

In our work, a word frequency list was experienced and driven from Turkish websites found from Alexa.com.

CHAPTER 4

DEVELOPMENT SOFTWARE

4.1 Combining Related Data for Countries' Visibility

Firstly all ranking data that were used for our new algorithm is combined. This process is shown in Figure 4.

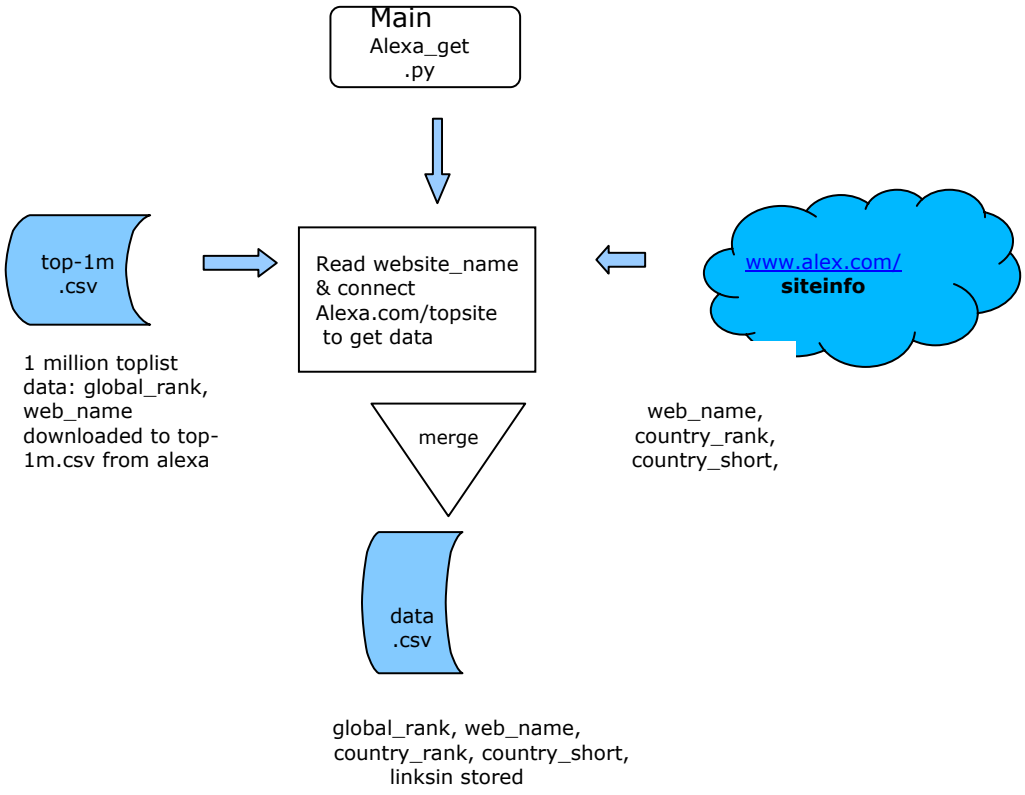

Figure 4: Combine related information in a database

Websites ranked and positioned in "1 million top list" is available to be downloaded at www.alexa.com/toplistthat contains only global rank and the name of the website. "1 million top list" is updated in monthly bases

and "1 million top list" used in our work is for January 2010. Rank in the country, country code and links in data reside in other page views which were inquired with the name of website. All data that was used to determine countries' visibility is combined in data.csv.

4.2 Creating Database to Report Countries' Visibility

Combined website specific ranking data is then appended to a database to manipulate whole data with keys. The new database is processed in country basis to create a summary database ready to be interpreted (Figure 5).

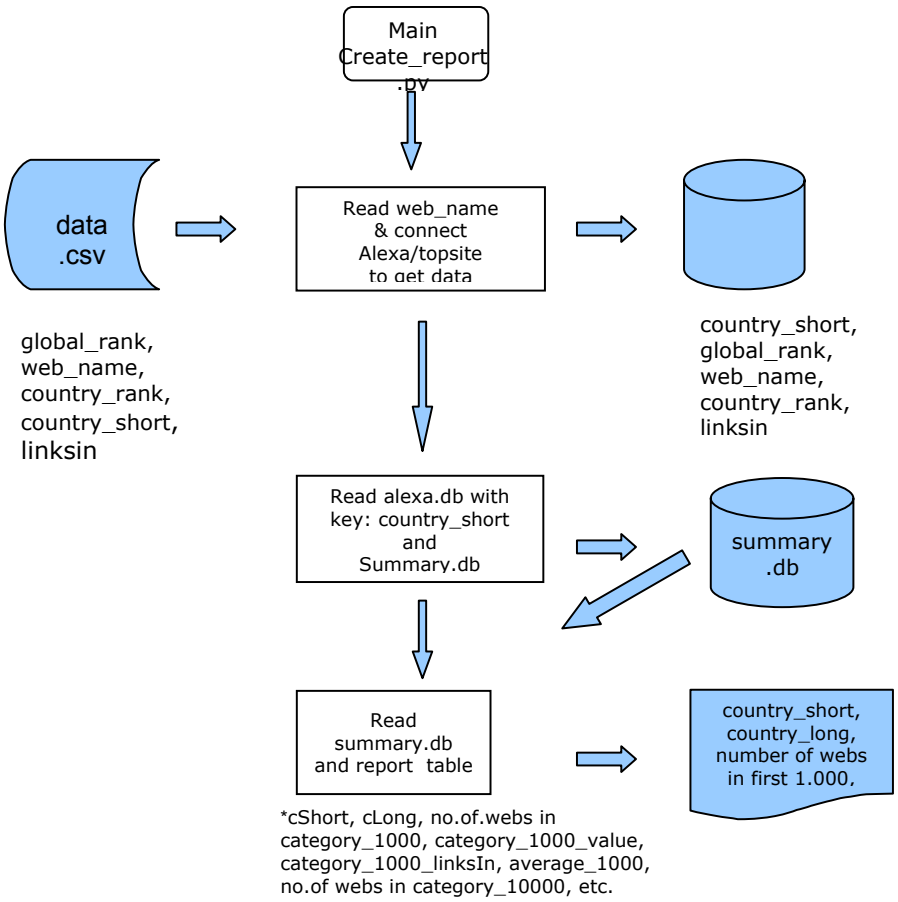

Figure 5: Extracting and transforming Alexa traffic ranking data

Summary.db is the basis for interpreting countries' Internet visibility, as it will be discussed in the next chapters.

4.3 Determining the Websites to be Searched

Two different sets of keywords (as defined in Chapter 3) are used to get page rank list of specific search engines/resources such as; Google, Yahoo, Altavista, Bing and Aol to determine websites to be crawled.

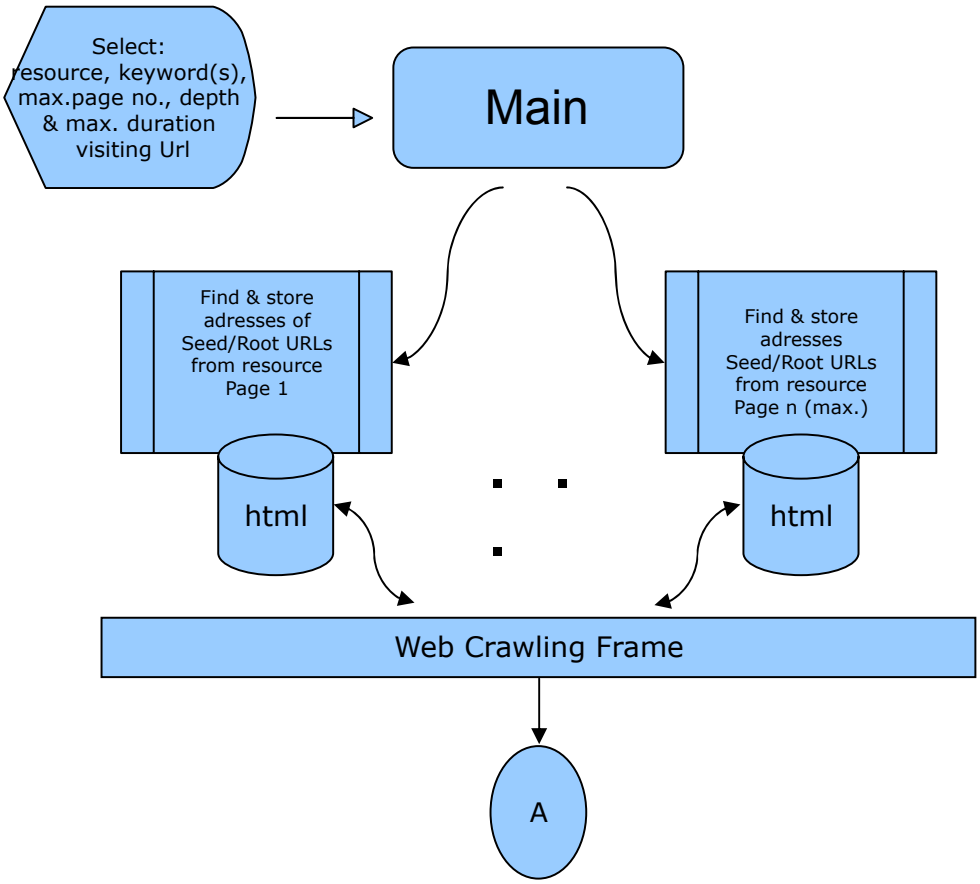

Figure 6: Frame of web crawling

This method, depicted in Figure 6, assures finding web pages having similar context before crawling the web pages. Matching the keywords with the context of the web search engines ensures finding the correct target to be searched.

Our main script crawls the web addresses alternatively related to the variant input scenarios (as shown in Appendix D) for each run.

Four keyword scenarios were prepared to run main script for 19 keywords determined heuristically. Those input scenarios are given in Appendix D and were named as;

keywords_forGoogle.txt
keywords_forYahoo.txt
keywords_forAltavista.txt
keywords_forAltavista.txt
keywords_forAol.txt

Two keyword scenarios were prepared to run main script for 50 keywords determined from study of Dr. Göz [2]. Those input scenarios were named as; keywords_forGoogle.txt, keywords_forAltavista.txt.

Content of those input files (of scenarios) were the same except first column of the data file whose recourse name was different.

There were five input values at each line of a scenario. First input value was the name of the resource to search keyword(s). The second input was the keyword(s) to be searched at the defined resource. List at Appendix B is the reference of selected keyword(s). Third input was the maximum number of resource page to be inquired. Fourth input was the maximum number of depth to be searched at the target webs. Fifth and the last input was the maximum time spent to search in the target webs.

The main script demands a page of web addresses from selected resource at each time. Those resource supplied web addresses are (related with the keywords and) written to HTML file to be parsed. Then, the parsed addresses are kept as root addresses of URLs to a todo_list.

4.4 Crawling of Target Websites to Collect Data

Script started visiting each root URL serially in the todo_list. When it visits the first URL address it finds first level addresses and appends those addresses to the todo_list then starts crawling down to the maximum depth or maximum time defined to the one was satisfies first. Our script crawls HREF links within the same tree to find HTML pages. When HTML page are found, it starts parsing to extract addresses of the internal and external links at the HTML page. Our script counts the number of images and/or forms when it encounters. On the other hand, the script counts all keywords in the list and inserts all needed information to the web db (as shown in Appendix C) before starting crawling new Ur address at the todo_list.

Our script only visits internal addresses, which have the same initial root address. It keeps counts of internal and external addresses for statistical usage.

When the crawling process was come to the end, URL address is appended to the doneLst and deletes the URL address, which was visited from todo_list. Before it starts crawling new URL, it checks the existence of new URL address at the doneLst, it starts crawling new URL address if it does not exist in the doneLst.

When main script encounters an exception it records those exception (errors) and other necessary information (as shown in Appendix C) to the log_error db.

Another output file was the Exitstat db, which contains summary records that were created at the end of each run (end of process of each data line). Each summary record is created after execution of each line in the input scenario file. In other words, each line in the scenario file corresponded to each summary record, which had general information of that finished run (as shown in Appendix C).

4.5 Preparing Data for the Analysis

Three files (web.db, log_error.db and Exitstat.db) are created after running main script using alternative scenarios (Figure 7).

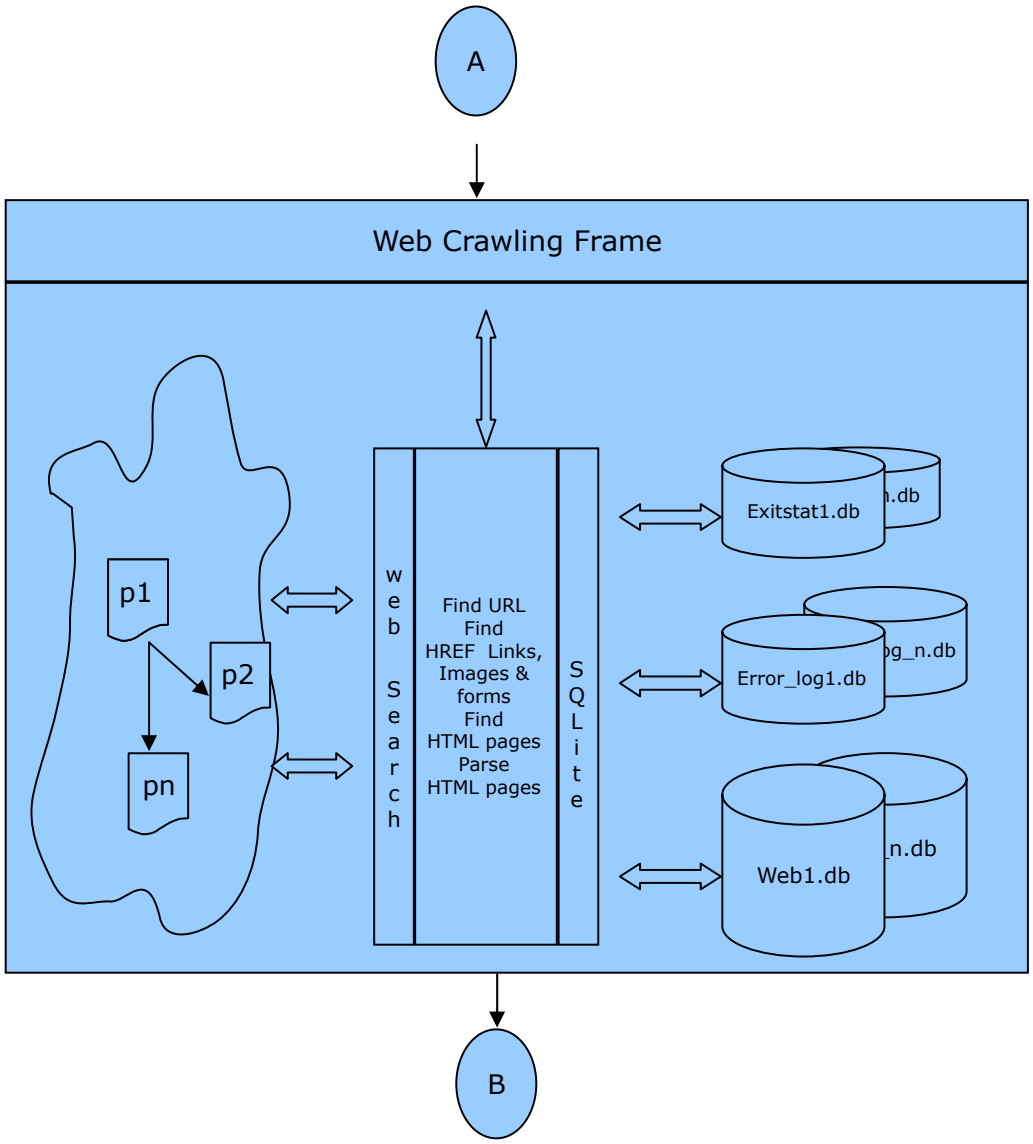

Figure 7: Frame of web crawling

Then sets of web.db and Exitstat.db files are filtered to direct Turkish and non-Turkish web page records to related top level domain files. There were many top level domains, such as; aero, asia, biz, cat, com, coop, edu, gov, info, int, jobs, mil, mobi, museum, name, net, org, pro, tel, travel, however, we concentrated only on com, edu, gov and org types.

We directed only web page records having com, edu, gov and org extensions on their names meaning that we had checked our hypothesis only on those files which were named specific top level domains that contained related Turkish and non-Turkish records.

Generalized description of filtering data related with the top level domain named file is shown in Figure 8 below.

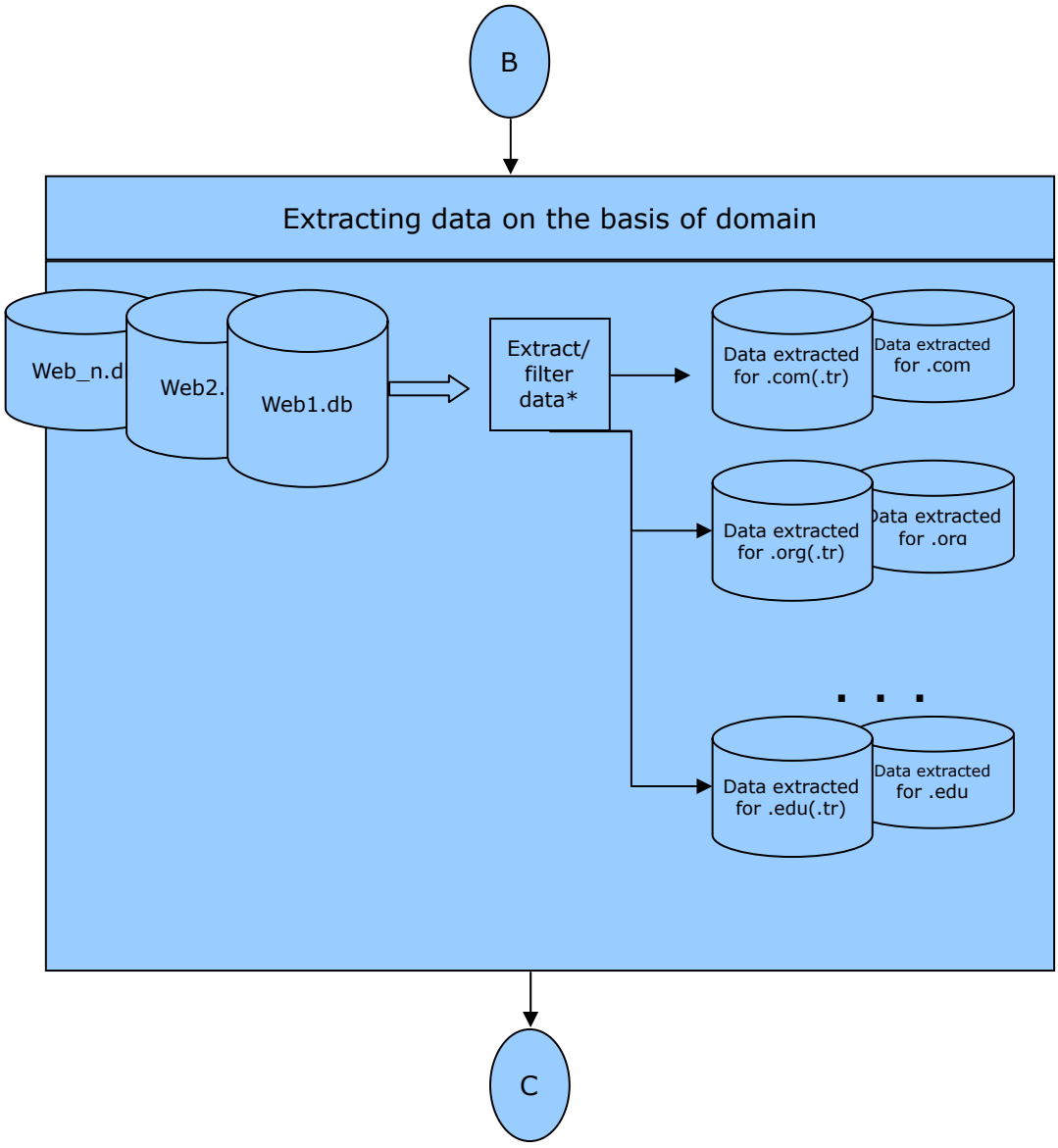

Figure 8: Extracting URL data on domain base

Detailed visualization of extracting URL records to related domains can be seen in Appendix F.

Our main script may visit the same web addresses multiple times, in other words, every web db and/or set of web dbs may contain multiple records of the same web addresses. In Appendix F, data records in web dbs were filtered and directed to the relevant top-level domain dbs.

If the record was inserted previously to the relevant domain db, the latest created record found at web dbs was updated instead of the record, which was previously inserted to domain db.

We have five pair of (.com, .edu, .gov, .info and .org) Turkish and non-Turkish domain files that were ready to be analyzed in the next chapters.

4.6 Analysis of Collected Data

We inspected so many valuable data about the web page when we visited and recorded those information (such as; total size of URL including size of internal addresses, total number links at the first level, total number of internal links addresses used in URL, total number of external links in the URL, total number ref documents referenced within URL, total number images referenced within URL, total number forms referenced within URL, maximum depth which was inspected at the rightmost native address, etc.,) at Web dbs. Using three alternative criteria that have already been defined in Chapter 3, similarity of Turkish domains were searched and findings are shared in Chapter 5's Results section.

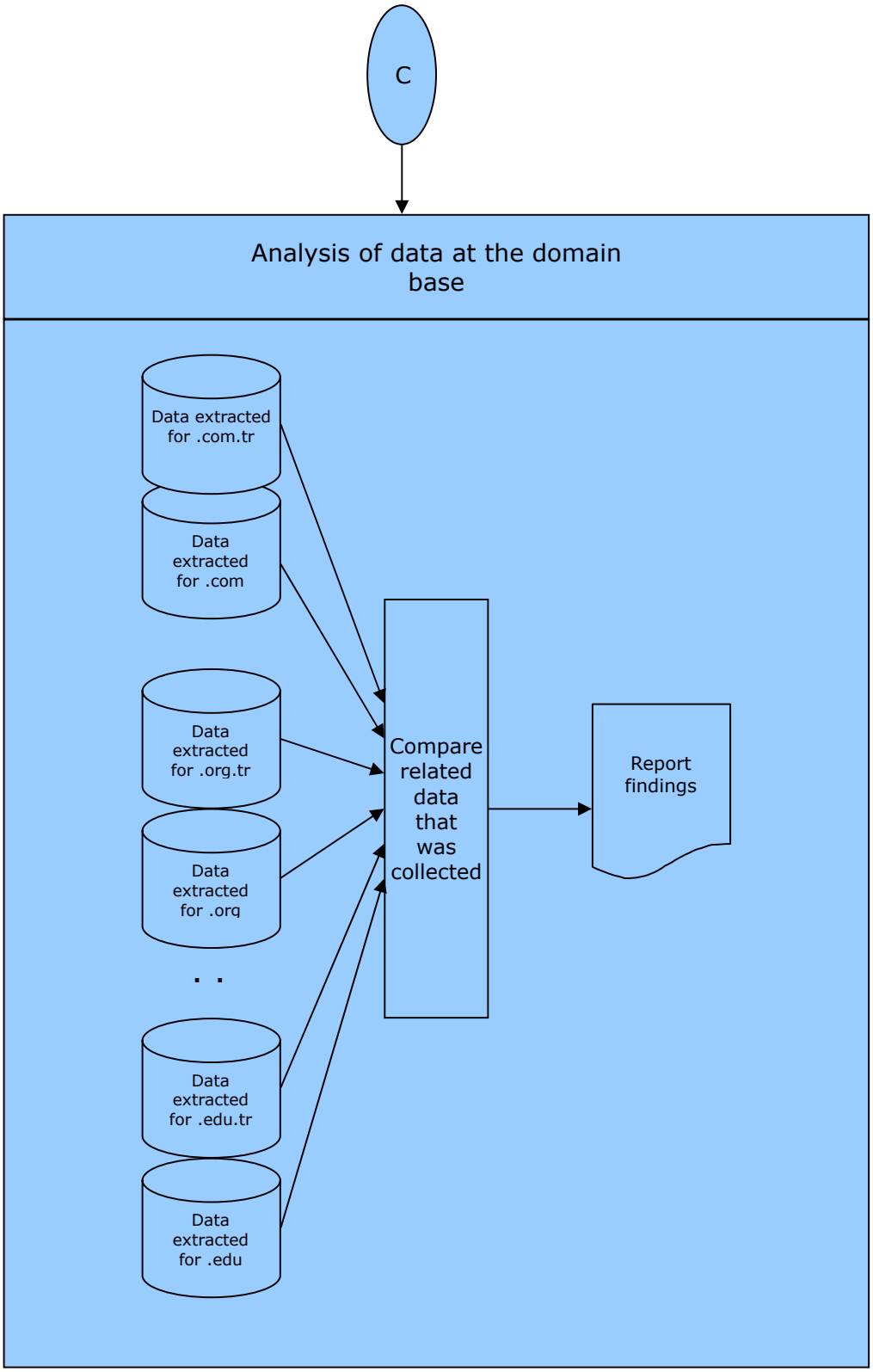

Figure 9: Data analysis on the base of domain

Selected visual URL records were decomposed to their belonging domains to compare related domain pairs.

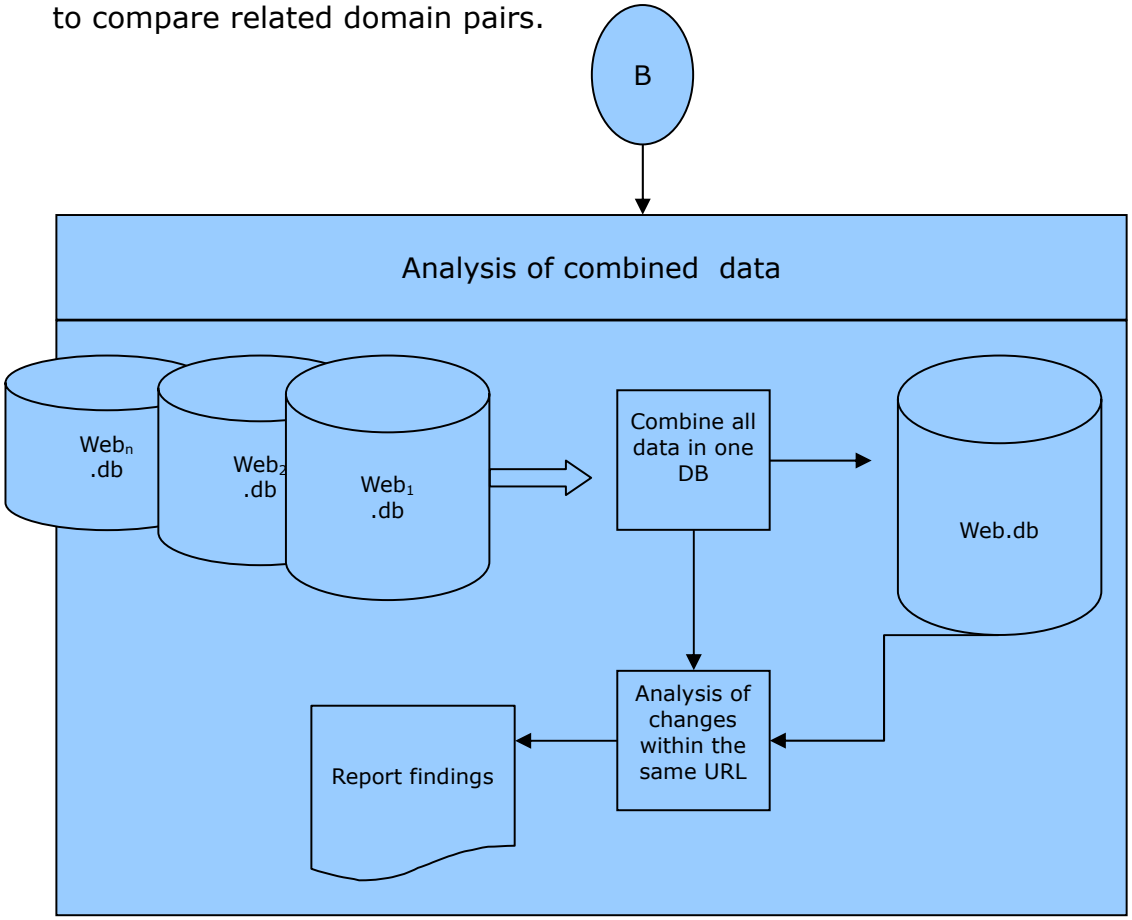

Figure 10: Data analysis on the whole data

Another analysis was performed on combined data of all Webs. We were interested in how often a web page had been changed within our searching period. Mean of time interval in between visits to the same web page was calculated (in Appendix F).

4.7 Determining Word Frequency List from Alexa.db

One of the findings of our work is an experience to produce frequency word list from addresses of Turkish websites found at Alexa.db assuring the crawled pages generally having Turkish content.

The outputs of our script (search_for_words_at_trURLs_from_Alexa.py) print out and create a text file of dictionary that keeps words. Outputs show us Turkish and as well as English keywords may exist in the

dictionary. The frequency word list was shared and discussed at the next chapters.

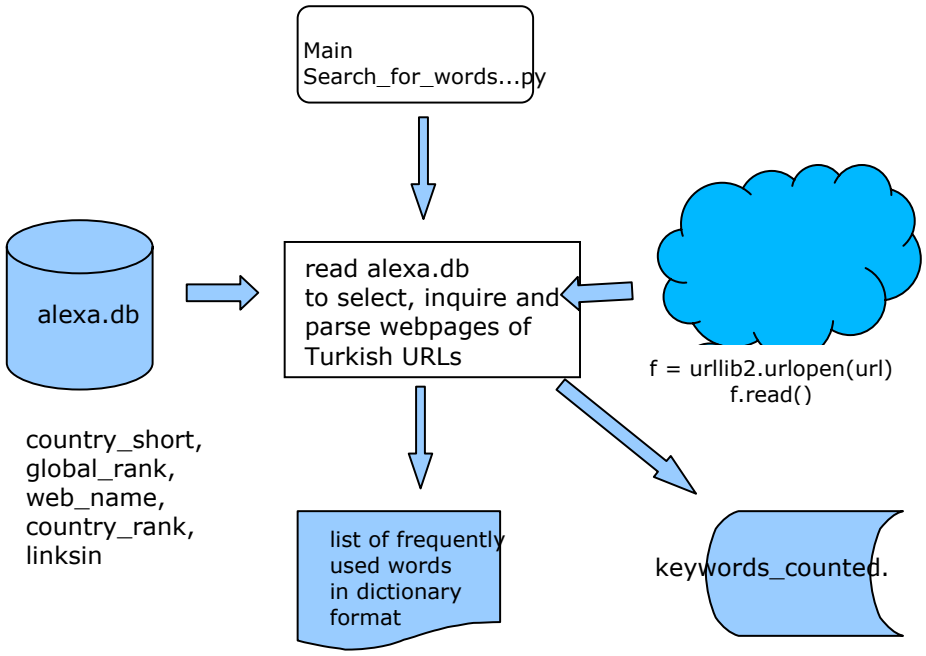

Figure 11: Data analysis on the whole data

CHAPTER 5

DISCUSSION OF THE RESULTS

5.1 Positioning of Countries

5.1.1 Ranking of Countries Based on Traffic Density Data from Alexa.com

The list in Appendix F is the converted form of top 1.000.000 globally ranked websites to the list of ranking countries.

The data records of top 1.000.000 websites have been stored to the Alexa.db database (as shown at Table 15 in Appendix A) which organised with composite prime keys of CShort, Wrank and Name making ease to reach and manipulate records in country base.

Four main categories (first: 1-1.000, second: 1.001-10.000, third: 10.001-100.000, fourth: 100.001- 1.000.000) were defined and 1.000.000 websites were placed in those categories related to their positions in the ranking list. Then websites at the top 1.000.000 list were distributed to those categories based on their positions (Table 1).

All four categories were used to display visibility of 230 countries.

Table 1: WEB Sites placed in the first category (1-1.000)

	Country	**total no. of webs at category 1 - 1.000**	Σ(1.000.000 - positions at category 1 - 1.000)	sum of links_in values at category 1 - 1.000	total no. of webs at category 1.001 - 10.000	Σ(1.000.000 - positions at category 1.001 - 10.000)	sum of links_in values at category 1.001 - 10.000
1	United States	431	430784910	1,4E+07	3360	3341899703	1,4E+07
2	People's Republic of China	127	126939719	1353704	847	842432403	1195827
3	Japan	61	60969995	693855	468	465464648	782823
4	India	55	54967645	117183	578	574874529	1012215
5	Germany	41	40977672	297137	454	451495286	820898
6	France	33	32982042	432897	264	262493316	477760
7	Russia	29	28988692	415577	320	318140208	405786
8	Brazil	17	16994328	188111	156	155052936	175186
9	United Kingdom	16	15992147	380930	195	193931941	483021
10	Italy	14	13993854	73503	147	146222134	324375
11	Spain	13	12993605	152708	180	179030596	377135
12	UnDefined	12	11992066	0	321	319108315	0
13	Mexico	12	11995215	58437	130	129266338	133131
14	**Turkey**	**11**	**10992909**	**28803**	**105**	**1,04E+08**	**124582**
15	Poland	9	8996299	76159	88	87488637	191464
16	Taiwan	9	8994231	47076	59	58680893	54741
17	Indonesia	8	7996518	111013	74	73620005	108831
18	Iran	7	6996287	43096	126	125266934	117390
19	Saudi Arabia	7	6996883	22582	120	119355594	128514
20	Egypt	5	4996822	5426	87	86560090	94537

Only first category was satisfying to display the visibility of 67 countries those have websites ranking in the first category. 47 countries were placed in the second category having place in between 68-114, 64 countries exist in the third category having place in between 115-178 and the fourth category contains 52 countries placed in between 178-230.

The cumulative positional value of websites (Σ(1.000.000 - position of website)) were calculated and used as an indicator of positional weight of websites as shown in Table 2. Ranking countries with this indicator causes small changes in the placement of countries when we compare the ranking list at Table 1.

Table 2: Positional weight factor used to display visibility of countries

	Country	total no. of webs at category 1 - 1.000	**Σ(1.000.000 - positions at category 1 - 1.000)**	sum of links_in values at category 1 - 1.000	total no. of webs at category 1.001 - 10.000	Σ(1.000.000 - positions at category 1.001 - 10.000)	sum of links_in values at category 1.001 - 10.000
1	United States	431	430784910	1,4E+07	3360	3341899703	1,4E+07
2	People's Republic of China	127	126939719	1353704	847	842432403	1195827
3	Japan	61	60969995	693855	468	465464648	782823
4	India	55	54967645	117183	578	574874529	1012215
5	Germany	41	40977672	297137	454	451495286	820898
6	France	33	32982042	432897	264	262493316	477760
7	Russia	29	28988692	415577	320	318140208	405786
8	Brazil	17	16994328	188111	156	155052936	175186
9	United Kingdom	16	15992147	380930	195	193931941	483021
10	Italy	14	13993854	73503	147	146222134	324375
11	Spain	13	12993605	152708	180	179030596	377135
12	Mexico	12	11995215	58437	130	129266338	133131
13	UnDefined	12	11992066	0	321	319108315	0
14	**Turkey**	**11**	**10992909**	**28803**	**105**	**1,04E+08**	**124582**
15	Poland	9	8996299	76159	88	87488637	191464
16	Taiwan	9	8994231	47076	59	58680893	54741
17	Indonesia	8	7996518	111013	74	73620005	108831
18	Saudi Arabia	7	6996883	22582	120	119355594	128514
19	Iran	7	6996287	43096	126	125266934	117390
20	Australia	5	4997745	54273	47	46728693	101322
21	Canada	5	4997371	16179	55	54730522	198790
22	South Korea	5	4997083	11362	38	37778515	32667
23	Egypt	5	4996822	5426	87	86560090	94537
24	Vietnam	4	3998730	12027	53	52687411	39390
25	Argentina	4	3998233	60549	19	18881650	31349
26	Netherlands	4	3998100	19798	50	49721840	112127
27	Thailand	4	3998050	13335	55	54671053	55792
28	Czech Republic	4	3997150	16554	26	25870614	139680
29	Hong Kong	3	2997625	4722	24	23866011	52575

Sum of links in (number of references to the related website) values is the summation of all links in values of websites of the same country and is used as an indicator to rank the countries. This is shown in Table 3.

Table 3: Sum of links in value used as an indicator to determine visibility of countries

	Country	total no. of webs at category 1 - 1.000	Σ(1.000.000 - positions at category 1 - 1.000)	**sum of links_in values at category 1 - 1.000**
1	United States	431	430784910	14340410
2	People's Republic of China	127	126939719	1353704
3	Japan	61	60969995	693855
4	France	33	32982042	432897
5	Russia	29	28988692	415577
6	United Kingdom	16	15992147	380930
7	Germany	41	40977672	297137
8	Brazil	17	16994328	188111
9	Spain	13	12993605	152708
10	India	55	54967645	117183
11	Indonesia	8	7996518	111013
12	Poland	9	8996299	76159
13	Italy	14	13993854	73503
14	Argentina	4	3998233	60549
15	Mexico	12	11995215	58437
16	Australia	5	4997745	54273
17	Taiwan	9	8994231	47076
18	Iran	7	6996287	43096
19	**Turkey**	**11**	**10992909**	**28803**
20	Philippines	2	1999396	25380
21	Saudi Arabia	7	6996883	22582
22	Netherlands	4	3998100	19798
23	Czech Republic	4	3997150	16554
24	Canada	5	4997371	16179
25	Thailand	4	3998050	13335
26	Hungary	2	1999071	12501
27	Vietnam	4	3998730	12027
28	South Korea	5	4997083	11362
29	Sweden	2	1999025	10868
30	Austria	2	1998993	7200

A score built multiplying varied coefficients with the total number of webs for all categories (5 points for 1-1.000, 4 points for 1.001-10.000, 3 points for 10.001-100.000, 2 points for 100.001-1.000.000).

The score found is determined with the weight of all ranking data in four categories, in another words, reflects effect of all values at all categories (Table 4).

Table 4: Visibility of countries with a score found multiplying predefined coefficients with the website counts for all categories

	Country Long	**Traffic density for all categories: 5 points for 1-1.000, 4 points for 1.001-10.000, 3 points for 10.001-100.000, 2 points for 100.001-1.000.000**	total no. of webs at category 1 - 1.000	total no. of webs at category 1.001 - 10.000	total no. of webs at category 10.001 - 100000	total no. of webs at category 100.001 - 1.000.000
1	United States	491912	431	3360	27969	196205
2	UnDefined	365050	12	321	7014	171332
3	People's Repubic of China	146965	127	847	8014	59450
4	India	143834	55	578	7037	60068
5	Germany	124981	41	454	4290	55045
6	Russia	88411	29	320	3396	38399
7	Japan	81768	61	468	3981	33824
8	United Kingdom	52546	16	195	1890	23008
9	Italy	41576	14	147	1590	18074
10	France	38746	33	264	2177	15497
11	Spain	32956	13	180	1689	13552
12	Brazil	32005	17	156	1674	13137
13	**Turkey**	**29246**	**11**	**105**	**1223**	**12551**
14	Indonesia	23965	8	74	905	10457
15	Iran	22638	7	126	1263	9155
16	Netherlands	22396	4	50	706	10029
17	Poland	21447	9	88	866	9226
18	Mexico	20428	12	130	1230	8079
19	Australia	17059	5	47	484	7697
20	Saudi Arabia	16294	7	120	1075	6277
21	Canada	15547	5	55	484	6925
22	Thailand	11838	4	55	560	4959
23	Pakistan	11534	1	15	433	5085
24	United Kingdom	11183	0	6	129	5386
25	South Africa	11137	1	20	360	4986
26	Sweden	10746	2	30	384	4732
27	Ukraine	9614	2	23	324	4270
28	Greece	9319	1	28	348	4079
29	Romania	9307	1	19	340	4103
30	Taiwan	9077	9	59	542	3585
31	Egypt	8274	5	87	627	3010
32	Austria	8039	2	16	219	3654
33	Argentina	8015	4	19	309	3496

5.1.2 Ranking of Countries Based on the Data Related with Top 500 in Webometrics.com

The Country Scoreboard (as shown in Appendix F) has been designed based on the model of the QS SAFE National System which is based on the "Webometrics Ranking of World Universities" (which is an initiative of the Cybermetrics Lab), evaluation of the countries' higher education system according to the presence of their Universities in the Top 500 of the Ranking Web.

5.1.3 Word Frequency List Created Crawling Websites at Alexa.com

A word frequency list was created from Turkish websites found from Alexa.com (Appendix F).

5.2 Results of Web Crawling

Three alternative criteria are used for web;

50 Turkish keywords used most frequently in written Turkish (determined by Dr. İlyas Göz), [2]

19 Turkish words were selected by intuitively.

The Turkish websites driven from Alexa.com are visited and searched for the criteria (i).

Results of these are shown in Table 5.

Table 5: Comparison of outcomes found at alternative searches

	result of inquiry with 19 keywords selected intuitively	result of inquiry with 50 keywords from Dr. İlyas Göz research	inquiry result of websites selected from Alexa.com search
Count of all URLs having size > 0	99.217	18.975	12.780
Average URL size	5.901.021	12.494.392	6.031.198
Average URL width	36	70	60
Average internal links	97	181	122
Average external links	162	482	92
Average HREF links info	6.590	13.469	9.191
Average IMG links	217	622	230
Average FORM links	2	5	0
Average search duration (in seconds)	96	247	164

5.2.1 Graphical Presentation of Statistical Data's

Data related with URL sizes in Turkish Domains are shown in Table 6 and Figure 12.

Table 6: Average size of domains found at alternative search algorithms

	com.tr	edu.tr	gov.tr	net.tr	org.tr	info.tr	other tr domains
inquiry with 19 keywords selected intuitively	5.248.407	7.113.806	7.738.267	7.529.096	8.600.930	6.528.715	4.294.653
inquiry with 50 keywords from Dr. İlyas Göz research	11.763.920	18.950.240	16.473.851	13.731.743	13.472.979	10.455.197	14.914.136
result of websites selected from Alexa search	5.441.718	7.584.441	7.658.792	7.549.384	7.347.372	6.029.904	6.607.318

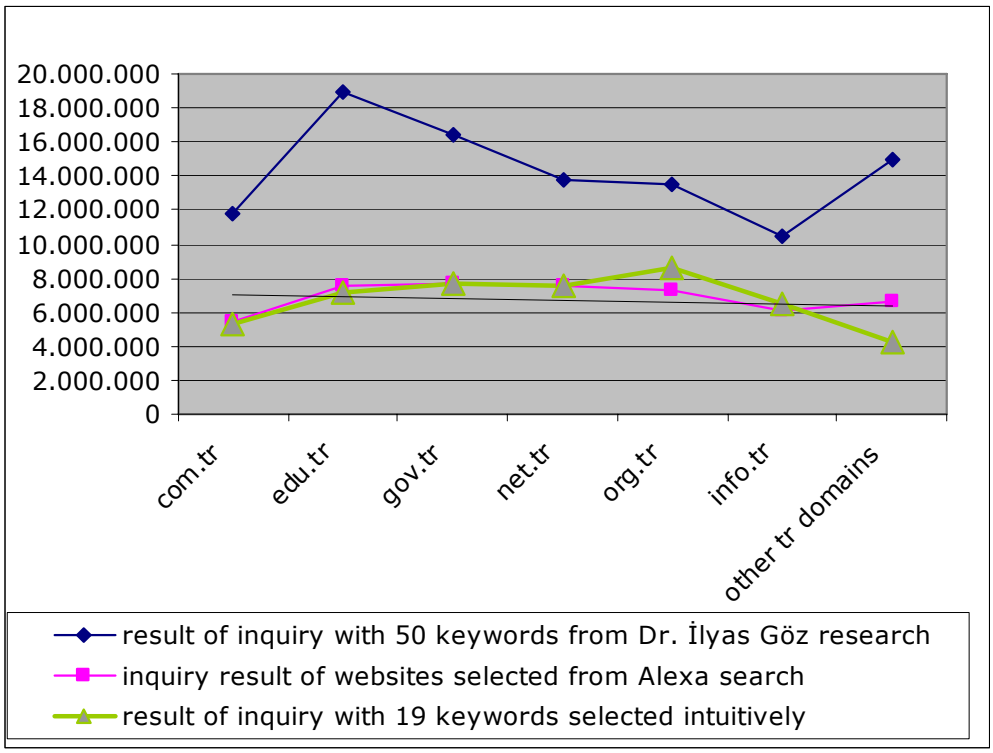

Figure 12: Graphical representations of domains' average sizes

Data related with URL width in Turkish Domains are shown in Table 7 and Figure 13.

Table 7: Average width of domains found at alternative search algorithms

Comparison of Turkish domains' average width found with three different search algorithms							
Turkish domains	**com.tr**	**edu.tr**	**gov.tr**	**net.tr**	**org.tr**	**info.tr**	**other tr**
result of inquiry with 19 keywords selected intuitively	38	14	13	55	28	77	25
result of inquiry with 50 keywords from Dr. İlyas Göz research	71	26	25	99	57	78	62
inquiry result of websites selected from Alexa search	54	23	14	82	71	95	67

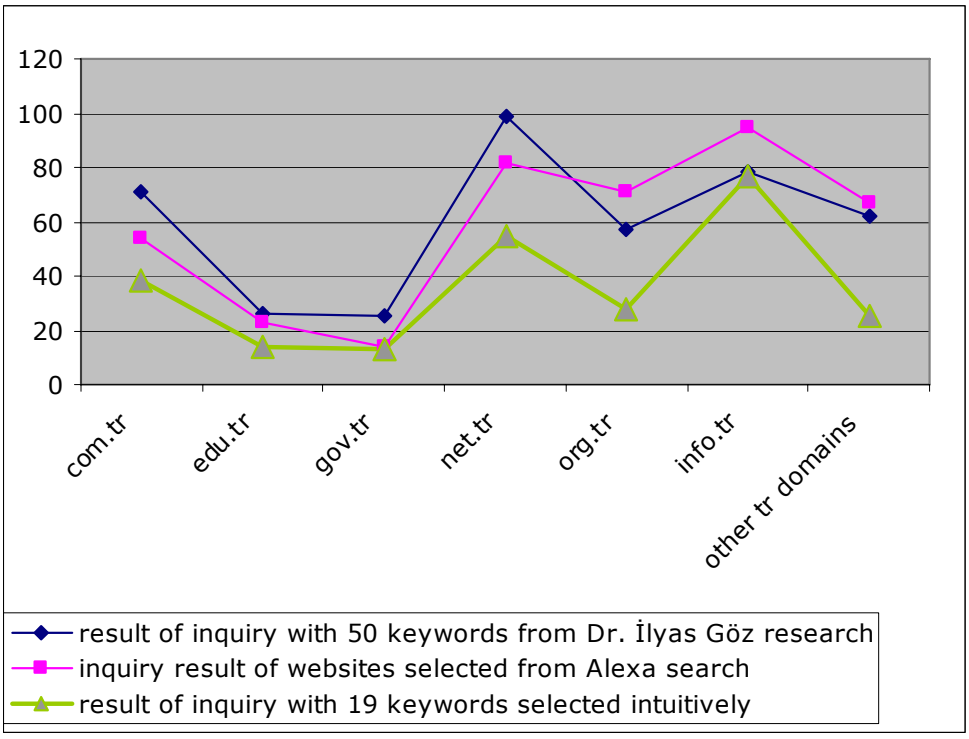

Figure 13: Graphical representation of domains' average width

Data related with Internal Links in Turkish Domains are shown in Table 8 and Figure 14.

Table 8: Average number of internal links of domains found at alternative search algorithms

Compare average number of Internal Links found in Turkish domains with three different search algorithms							
Turkish domains	**com.tr**	**edu.tr**	**gov.tr**	**net.tr**	**org.tr**	**info.tr**	**other tr domains**
result of inquiry with 19 keywords selected intuitively	98	32	14	174	82	147	66
result of inquiry with 50 keywords from Dr. İlyas Göz research	178	98	67	271	175	174	137
inquiry result of websites selected from Alexa search	112	19	15	167	144	155	129

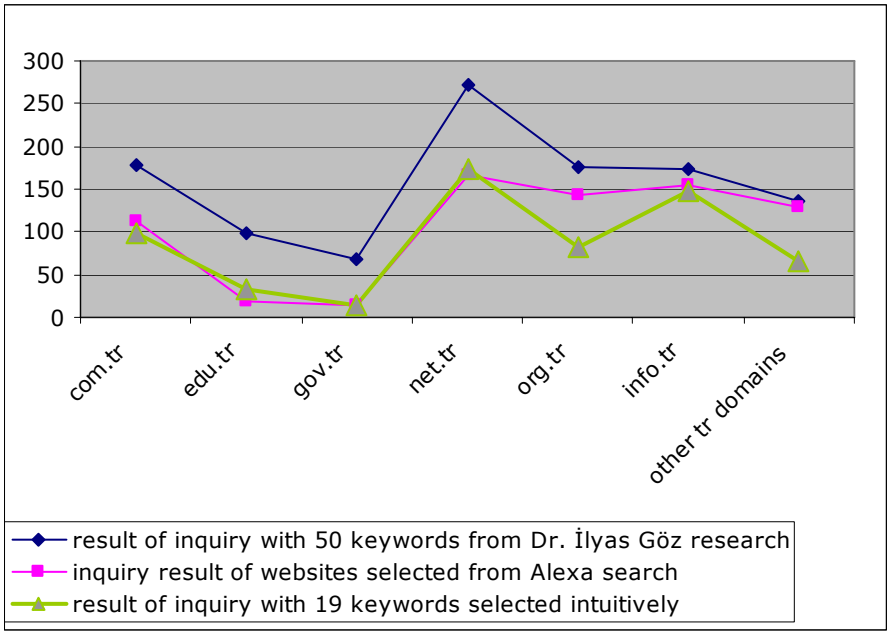

Figure 14: Graphical representation of average number of internal links

Data related with External Links in Turkish Domains are shown in Table 9 and Figure 15.

Table 9: Average number of external links of domains found at alternative search algorithms

Compare average number of External Links found in Turkish domains with three different search algorithms							
Turkish domains	**com.tr**	**edu.tr**	**gov.tr**	**net.tr**	**org.tr**	**info.tr**	**other tr domains**
result of inquiry with 19 keywords selected intuitively	186	68	19	102	202	272	55
result of inquiry with 50 keywords from Dr. İlyas Göz research	547	262	170	302	429	150	141
inquiry result of websites selected from Alexa search	88	36	20	106	124	90	85

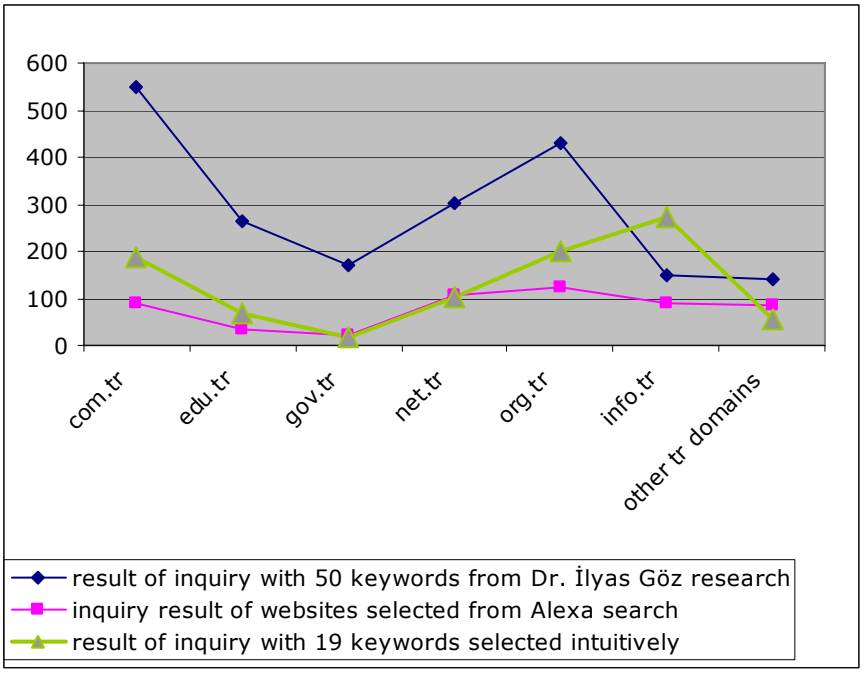

Figure 15: Graphical representation of external links as domain base

Data related with HREF Links in Turkish Domains are shown in Table 10 and Figure 16.

Table 10: Average number of Href links of domains found at alternative search algorithms

Compare average number of HREF Links found in Turkish domains with three different search algorithms							
Turkish domains	**com.tr**	**edu.tr**	**gov.tr**	**net.tr**	**org.tr**	**info.tr**	**other tr domains**
result of inquiry with 19 keywords selected intuitively	6.946	504	548	10.833	5.902	12.624	3.629
result of inquiry with 50 keywords from Dr. İlyas Göz research	13.885	2.302	2.166	18.747	11.830	12.239	10.472
inquiry result of websites selected from Alexa search	8.487	391	673	12.579	9.994	12.692	10.263

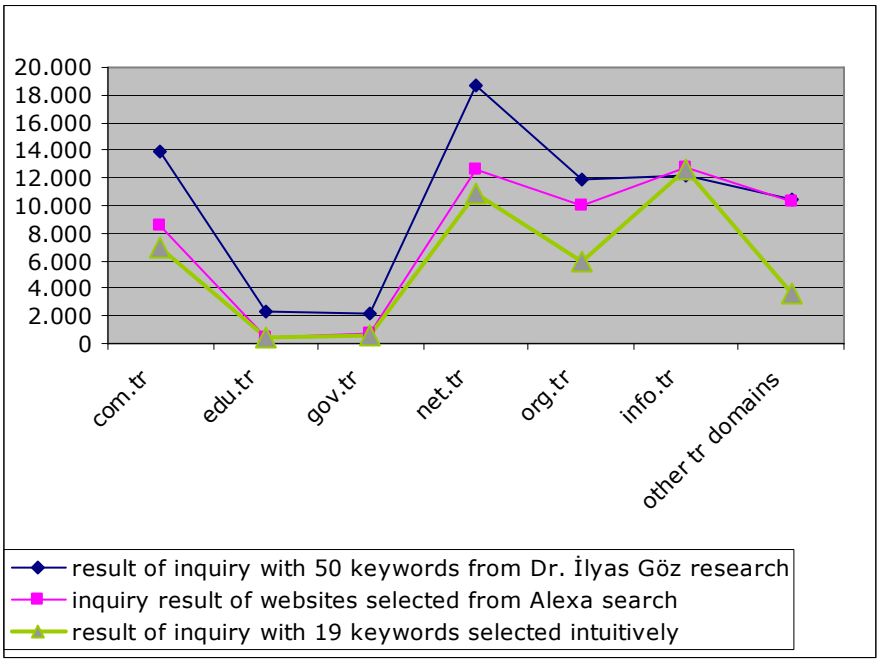

Figure 16: Graphical representation of HREF links as domain base

Data related with Image Links in Turkish Domains are shown in Table 11 and Figure 17.

Table 11: Average number of Image links of domains found at alternative search algorithms

Compare average number of Images Links found in Turkish domains with three different search algorithms							
Turkish domains	**com.tr**	**edu.tr**	**gov.tr**	**net.tr**	**org.tr**	**info.tr**	**other tr domains**
result of inquiry with 19 keywords selected intuitively	240	111	43	275	186	158	96
result of inquiry with 50 keywords from Dr. İlyas Göz research	679	293	173	672	410	331	387
inquiry result of websites selected from Alexa search	224	86	45	272	247	230	242

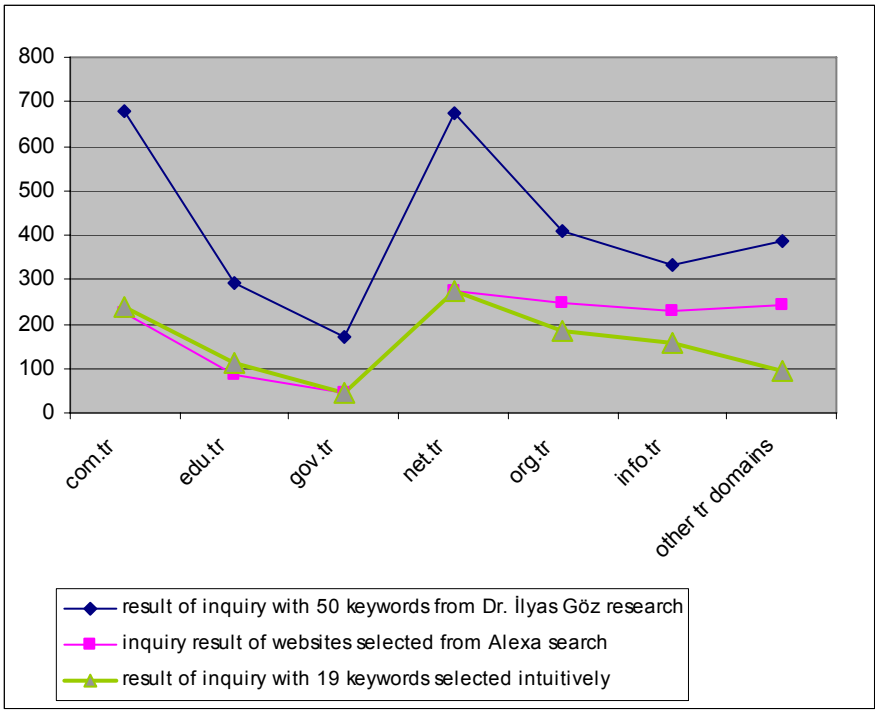

Figure 17: Graphical representation of average no. of Image links as domain base

Data related with search duration in Turkish Domains are shown in Table 12 and Figure 18.

Table 12: Average search duration (in seconds) as domain base

Average of search duration (in seconds) found in Turkish domains with three different search algorithms							
Turkish domains	**com.tr**	**edu.tr**	**gov.tr**	**net.tr**	**org.tr**	**info.tr**	**other tr domains**
result of inquiry with 19 keywords selected intuitively	95	49	26	153	97	198	68
result of inquiry with 50 keywords from Dr. İlyas Göz research	251	186	114	297	227	271	206
inquiry result of websites selected from Alexa search	150	30	66	218	199	212	182

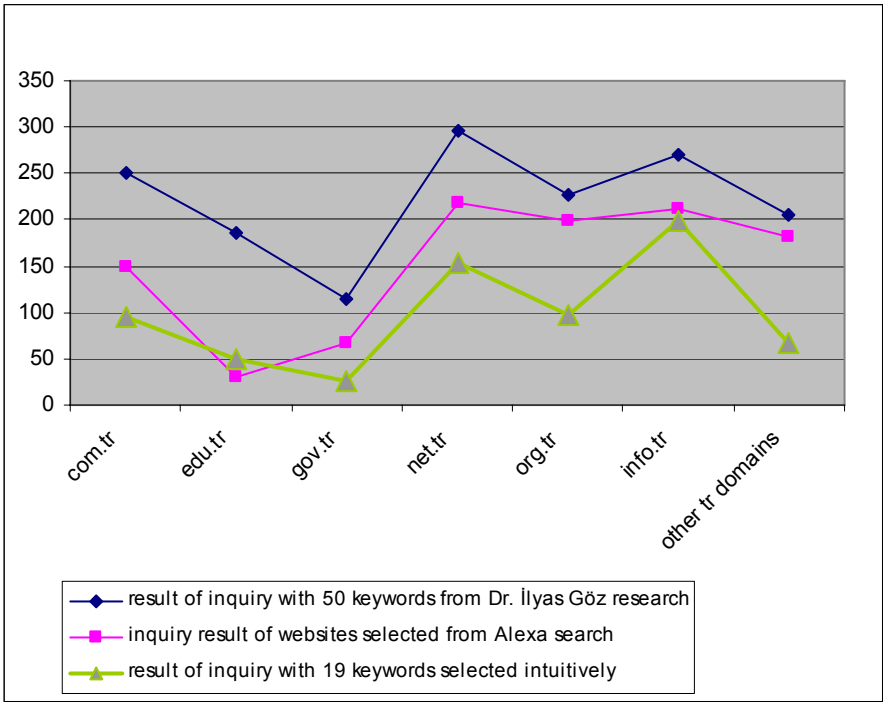

Figure 18: Graphical representation of average search duration as domain base

5.2.2 CPU Statistics Obtained from Summary Records

The overall CPU statistics related with web crawling within the scope of this work was presented on Table 13 to 15.

Total CPU time spent for crawling 467 combinations of 19 keywords took 97 days, 18 hours, 48 minutes and 43 seconds (Table 13). Details of all CPU timings are given in Appendix D.

Table 13: Total CPU time spent crawling 19 keywords.

		D U R A T I O N			
RESOURCE	**No. of runs**	**Days**	**Hours**	**Minutes**	**Sec.**
Altavista	35	22	7	19	37
Aol	113	17	3	26	22
Bing	190	7	22	10	10
Google	42	22	21	56	44
Yahoo	87	27	11	55	50
TOTAL	467	**97**	**18**	**48**	**43**

Total CPU time spent for crawling 50 keywords took 44 days, 14 hours, 49 minutes and 15 seconds (Table 14).

Table 14: Total CPU time spent crawling 50 keywords.

		D U R A T I O N			
RESOURCE	**No. of runs**	**Days**	**Hours**	**Minutes**	**Sec.**
Altavista	18	19	2	12	2
Google	27	25	12	41	53
TOTAL	45	**44**	**14**	**49**	**15**

6 separate runs executed for crawling Turkish websites at (1-100.000, 101.000-200.000, 200.001-400.000, 400.001-600.000, 600.001-800.000, 800.001-1.000.000 top list groups of) Alexa.com and 10 days, 13 hours, 7 minutes and 59 seconds were spent as CPU basis (Table 15).

Table 15: Total CPU time spent for all searches at Alexa.com.

		D U R A T I O N			
	No. of runs	**Days**	**Hours**	**Minutes**	**Sec.**
	6	10	13	7	59
TOTAL	6	**10**	**13**	**7**	**59**

5.3 Query Results of DNS Inspection

620.715 DNS log records were processed. 44333 records were selected which meet defined query type criteria's:

Query type: {'/MX/IN': 7343, '/SOA/IN': 12, '/A6/IN': 748, '/AAAA/IN': 6214, '/PTR/IN': 7, '/A/IN': 29205, '/NS/IN': 120, '/TXT/IN': 194, '/SRV/IN': 29, '/ANY/IN': 189, '/SPF/IN': 23, '/CNAME/IN': 249}

Total no. of query from Turkey: 96

Total no. of query from out of Turkey: 44237

5.4 Obtained Word Frequency List

A word frequency list was created from Turkish websites list found from list of top 1.000.000 websites at Alexa.com. More than 100.000.000 Turkish and non-Turkish words were determined and Turkish words were selected and grouped by manually. Then the words were sorted by

number of repetition and most frequent 51 words were listed as shown in Appendix F.

5.5 Turkish Websites Supporting Other Languages

The web audience is truly global, so a search has been achieved to determine the percentage of Turkish web sites supporting other languages. 116 Turkish websites are found (in top 10.000) to be supporting foreign users.

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

Results as shown in Chapter 5, have been evaluated in three sections;
visibility of Turkey (and countries),
findings obtained from the actual study and
replies of the questions as shared at 3.1 in Chapter 3.

6.1.1 Visibility of Countries

A significant finding of our work is that one could possibly reach different ranking of countries according to the parameters and/or criterion chosen.

No relation was determined between the results driven from Alexa.com and Webometrics.com. Knowledge base generated from these resources should be evaluated by two different aspects:
Alexa.com ranks websites based on their traffic density data.
Webometrics.com ranks global educational institutions using their web presence, measures of their activity to determine their visibility.
In Alexa.com Turkey’s position was evaluated within 231 countries. The traffic density data of websites at top 1.000.000 list of Alexa.com were decomposed in four categories on country basis. Turkey's position in other countries is found to be 14 according to the first category (Table 1).

Turkey's position in other countries is also found to be 14 according to the accumulated positional values of the websites [Σ(1.000.000 - positions at category 1 - 1.000)] on Table 2.

Turkey's position in other countries is found to be 19 according to the accumulated positional values of the websites (sum of links_in values at category 1 - 1.000) on Table 3.

Turkey's position in other countries is found to be 13, if the weight of all values in four categories were used for ranking on Table 4. If the weight of intervals were changed relatively: 5 points for 1-1.000; 0,5 points for 1.001-10.000; 0,05 points for 10.001-100.000; 0,005 points for 100.001-1.000.000, the position of Turkey would not change but other countries' position differs, as shown in Table 16.

Table 16: Different Weight of Intervals for Alexa.com Results.

	Country	**Traffic density for all categories: 5 points for 1-1.000; 0,5 points for 1.001-10.000; 0,05 points for 10.001-100.000; 0,005 points for 100.001-1.000.000**	total no. of webs at category 1 - 1.000	total no. of webs at category 1.001 - 10.000	total no. of webs at category 10.001 - 100000	total no. of webs at category 100.001 - 1.000.000
1	United States	6,214,475	431	3360	27969	196205
2	People's Republic of China	1756,45	127	847	8014	59450
3	UnDefined	1427,86	12	321	7014	171332
4	India	1216,19	55	578	7037	60068
5	Germany	921,725	41	454	4290	55045
6	Japan	907,17	61	468	3981	33824
7	Russia	666,795	29	320	3396	38399
8	France	483,335	33	264	2177	15497
9	United Kingdom	387,04	16	195	1890	23008
10	Italy	313,37	14	147	1590	18074
11	Brazil	312,385	17	156	1674	13137
12	Spain	307,21	13	180	1689	13552
13	**Turkey**	231,405	**11**	**105**	**1223**	**12551**
14	Mexico	226,895	12	130	1230	8079
15	Iran	206,925	7	126	1263	9155
16	Saudi Arabia	180,135	7	120	1075	6277
17	Poland	178,43	9	88	866	9226

All of the results derived from Alexa.com's "traffic rank" of a country may vary depending on where and how the values are evaluated. On the other hand, it should be noted that Alexa.com's traffic ranking data is dynamic and changes monthly.

The data related with "Webometrics Ranking of World Universities" determines the evaluation of the countries' higher education system according to the presence of their Universities in the Top 500 of the Ranking Web.

The Country Strength Rankings has been designed based on the model of the QS SAFE National System. Turkey's position in other countries is found to be 40, concerning the ranking as shown in Appendix F.

The Country Scoreboard has been designed based on the model of the QS SAFE National System. Turkey's position in other countries is found to be 45, with regards to the ranking as shown in Appendix F.

Turkey's position in other countries is found to be 36 within 240 countries with regards to the ranking to the "number of national domains" shown in Appendix F. It was a surprising result that USA is ranked at 17th.

Table 17: First Ten of "Number of National Domains" List.

RANK	COUNTRY	NUMBER OF NATIONAL DOMAINS
1	Germany (de)	80,100,000
2	United Kingdom (uk)	31,900,000
3	Japan (jp)	23,400,000
4	Netherlands (nl)	17,400,000
5	China (cn)	15,000,000
6	Russian Federation (ru)	14,900,000
7	Poland (pl)	12,300,000
8	Italy (it)	12,200,000
9	France (fr)	8,170,000
10	Brazil (br)	7,920,000

That figure does not imply ranking of countries according to their native language and the websites supports German is more than one of the other languages used worldwide (for example, English).

It should be noted that Webometrics.com ranking data changes twice in a year.

Rank of Turkey ranges between 13th and 45th positions in our search resources. 13th position was the best for Turkey within approximately 230-240 countries. Turkey's 45th position was reported at QS SAFE National System Country Scoreboard and determines and compares the position of Turkey's higher education institutions and universities within other countries. An organization responsible from higher educational institutions (such as YOK) has to deploy a working group to take and implement strategic decisions to improve the ranking of Turkish universities and the position of Turkey.

6.1.2 Findings Obtained from Crawling Data

Three alternative criterions have been used to search crawling websites:

50 Turkish keywords used most frequently in written Turkish (determined by Dr. İlyas Göz [2]),

19 Turkish words were selected by intuitively and

Turkish websites driven from Alexa.com are visited and searched to obtain data related with content of websites below.

Total CPU time spent crawling 518 combinations of keywords and Alexa.com is calculated 152 days, 22 hours, 45 minutes and 57 seconds. Those durations were recorded to compare time measurements to be obtained in future studies.

Obtained results have been presented in Tables 7-12 showing:

Average width of domains (as shown in Table 7)

Average number of internal links (as shown in Table 8)

Average number of external links (as shown in Table 9)

Average number of Href links (as shown in Table 10)

Average number of Image links (as shown in Table 11)

Average search duration (as shown in Table 12)

These results have shown the existence of correlation between three sets of data found from three alternative methods. In other words, data from different search strategies have similar characteristics and showing similar trends in each graph given in Chapter 5.

A word frequency list has been created from Turkish websites found from list of top 1.000.000 websites at Alexa.com. More than 100.000.000 Turkish and non-Turkish words (tokens) were determined and Turkish words were selected and grouped by manually. Then the words have been sorted by number of repetition and most frequent 51 words were listed (shown in Appendix F).

Comparison of word frequency list found in our work with the list of Dr. İlyas Göz, only 12 words were determined in the intersection area. The list found in our work has new words generally used in Turkish Internet terminology of present day. This shows an indication of dissimilarity in usage of Turkish at different domains.

Frequency word list generated in this Thesis is just an experimental work and occurrence of words found to be high. These numbers could not be directly compared with Dr. İlyas Göz's results, because the latter is based on written and limited number of documents, while former is using easy access to resources on Internet.

Query results of DNS log records inspection had shown us that 99% of inquiries were done from out of the Turkey.

A survey has been done to assess what percentage of Turkish websites is having multilingual features for foreign users. Result of this survey has shown us that %40 of Turkish websites in top 10.000 list found in Alexa.com supports more than one language and is truly global. Searching all Turkish websites within top 1.000.000 list to determine their openness to global access is considered as one of the future works.

6.1.3 Possible Answers to the Basic Questions

There is sufficient body of knowledge to answer questions that we raised at Chapter 3:

Q1. Will alternative methods of research provide us similar ranking values related with Turkey's Internet visibility?

A1. Results found from Alexa.com and Webometrics.com/QS SAFE National System has proved us that ranking of countries differ according to the issues being inspected. Turkey's position changes from 13th to 45th within 230-240 countries.

Q2. Will web crawling with different sets of keywords bring out diverse results related with the general characteristics of Turkish websites?

A2. Three alternative criteria were used to crawl websites; i. 50 Turkish keywords used most frequently in written Turkish (determined by Dr. İlyas Göz), [2] ii. 19 Turkish words were selected by intuitively and iii. The Turkish websites driven from Alexa.com are visited and searched for the criteria i

URL size, URL width, Internal Links, External Links, HREF Links and Image Links inspected in domain (com.tr, edu.tr, gov.tr, net.tr, org.tr, info.tr and others) basis crawling websites with three alternative criteria and different sets of keywords as explained in previous section.

URL size has not shown similarity in domain basis.

Q3. What are the common features of Turkish websites founded from three different algorithms?

A3. Some features of domains, such as; URL width, Internal Links, External Links, HREF Links and Image Links have similar trends in themselves. Data related with URL width, Internal Links, External Links, HREF Links and Image Links obtained from domains, has similar characteristics, at the result of crawling with three alternative criteria.

Q4. In what extend word frequency list found in our research will overlap to the written word frequency list of Dr. İlyas Göz?

A4. Dr. İlyas Göz's study has some drawbacks; first, the list itself does not pass through some morphological operations, second, compilation of texts of the early 2000s to participate - so do not reflect present-day. 12 words were determined overlapping in both lists. The reason why so few common words are in the intersection, is that the most of the words found in our search has been used lately (since last 10 years) and especially within web users.

6.2 Future Works

It is recommended that a new survey to determine the percentage of Turkish websites is open accesses to foreign users can be extended by applying for the top 1.000.000 list.

It would also be a good follow-up to our study, to automate all the scripts presented at Appendix D and facilitate to be used in the time-axis and compared the results obtained in.

REFERENCES

[1] Wikimedia Foundation Inc., http://en.wikipedia.org/wiki/Visibility, 01/05/2010.

[2] İlyas Göz, "Yazılı Türkçenin Kelime Sıklığı", 2003, XV + 576 s.,TDK

[3] Netcraft Limited, http://www.netcraft.com/survey/ [February 2010 Web Server survey] , 01/05/2010.

[4] Princeton University, http://wordnetweb.princeton.edu/perl/webwn, 01/05/2010.

[5] Wikimedia Foundation Inc., http://en.wikipedia.org/wiki/Visibility_(geometry) , 01/05/2010.

[6] Wikimedia Foundation Inc., http://en.wikipedia.org/wiki/Visibility_(corporation) , 01/05/2010.

[7] Wikimedia Foundation Inc., http://en.wiktionary.org/wiki/visibility, 01/05/2010.

[8] Princeton University, http://wordnetweb.princeton.edu/perl/webwn, 01/05/2010.

[9] Wikimedia Foundation Inc., http://en.wikipedia.org/wiki/Visible_(magazine) , 01/05/2010.

[10] Wikimedia Foundation Inc., http://en.wikipedia.org/wiki/Visible_(album) , 01/05/2010.

[11] Wikimedia Foundation Inc., http://en.wiktionary.org/wiki/visible, 01/05/2010.

[12] Consejo Superior de Investiqaciones, http://webometrics.info/glossary.html, 01/05/2010.

[13] National Academic Network and Information Center, http://www.kypost.com/content/weather/calculators/wxterms.aspx, 01/05/2010.

[14] Iowa Communications Network, http://www.weatherview.dot.state.ia.us/awosglossary.htm, 01/05/2010.

[15] Wikimedia Foundation Inc., http://en.wikipedia.org/wiki/Resource_record#DNS_servers [Domain Name Servers] , 01/05/2010.

[16] Performance System International Inc., http://www.cyveillance.com/web/news/press_rel/2009/2009-08-18.asp, 01/05/2010.

[17] Jon Kleinberg and Steve Lawrence, "NETWORK ANALYSIS: The Structure of the Web", Science, Vol. 294, No. 5548, pp. 1849-1850, 2001.

[18] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke and Sriram Raghavan, "Searching the Web", ACM Trans. Internet Techn., Vol 1, No 1, pp. 2-43, 2001.

[19] Filippo Menczer, Gautam Pant, Padmini Srinivasan, "Topical Web Crawlers", ACM Transactions Internet Technologies, Vol. 4, No. 4, pp. 378-419, November 2004.

[20] Consejo Superior de Investiqaciones, http://www.webometrics.info/methodology.html, 01/05/2010.

[21] QS Quacquarelli Symonds Ltd., www.topuniversities.com, 01/05/2010.

[22] Wikimedia Foundation Inc., http://en.wikipedia.org/wiki/Web_crawler, 01/05/2010.

[23] QS Quacquarelli Symonds Ltd., http://www.webometrics.info/Number_National_Domains_World.asp, 01/05/2010.

APPENDIX A

DATA FORMAT OF SQLITE TABLES

First three columns of Table 18 (Root, Date_inserted, Hostname_current) are designed to be "composite prime key". Each row contains 95 columns including fields of composite prime key.

Table 18: web.db (used for 19 keywords)

Field	***Attribute***	***Description***
Root	TEXT	name of the root URL,
Date_inserted	TEXT	date value (yy:mm:dd:hh:mm:ss) shows the time the row was inserted,
Hostname_current	TEXT	the name of the host on which script run,
Hostip_address	TEXT	the ip address of the host,
Root_ext	TEXT	extension of the Url which shows generic top-level domain,
Nation	TEXT	extension of Url which shows country code top-level domain,
Urlsize	INTEGER	total size of Url including size of internal addresses,
Urlwidth	INTEGER	total no. of links at the first level,
No_of_internal_links	INTEGER	no. of internal links addresses used in Url,
No_of_external_links	INTEGER	no. of external links in the Url,
Count_href	INTEGER	total no. of href documents referenced within Url,
Count_img	INTEGER	total no. of images referenced within Url,
Count_form	INTEGER	total no. of forms referenced within Url,
Tr_char0	TEXT	column is representing char ý,
Tr_charno0	INTEGER	total no. of ý was found in the Url,
Tr_char1	TEXT	column is representing char ı,
Tr_charno1	INTEGER	total no. of ı was found in the Url,
Tr_char2	TEXT	column is representing char ğ,
Tr_charno2	INTEGER	total no. of Ğ was found in the Url,
Tr_char3	TEXT	column is representing char 'ü',
Tr_charno3	INTEGER	total no. of 'ü' was found in the Url,
Tr_char4	TEXT	column is representing char 'ş',
Tr_charno4	INTEGER	total no. of 'ş' was found in the Url,
Tr_char5	TEXT	column is representing char 'ö',
Tr_charno5	INTEGER	total no. of 'ö' was found in the Url,
Tr_char6	TEXT	column is representing char 'ç',
Tr_charno6	INTEGER	total no. of 'ç' was found in the Url,
Tr_char7	TEXT	column is representing char Ğ,
Tr_charno7	INTEGER	total no. of Ğ was found in the Url,
Tr_char8	TEXT	column is representing char Ü,
Tr_charno8	INTEGER	total no. of Ü was found in the Url,
Tr_char9	TEXT	column is representing char Ş,
Tr_charno9	INTEGER	total no. of Ş was found in the Url,
Tr_char10	TEXT	column is representing char İ,
Tr_charno10	INTEGER	total no. of İ was found in the Url,

Table 18 continued.

Field	*Attribute*	*Description*
Tr_char11	TEXT	column is representing char Ö,
Tr_charno11	INTEGER	Total no. of Ö was found in the Url,
Tr_char12	TEXT	column is representing char Ç,
Tr_charno12	INTEGER	total no. of Ç was found in the Url,
Tr_char13	TEXT	column is representing char '&uuml' (ü),
Tr_charno13	INTEGER	total no. of '&uuml' was found in the Url,
Tr_char14	TEXT	column is representing char '&ouml' (ö),
Tr_charno14	INTEGER	total no. of '&ouml' was found in the Url,
Tr_char15	TEXT	column is representing char 'ccedil' (ç),
Tr_charno15	INTEGER	total no. of 'ccedil' was found in the Url,
Tr_char16	TEXT	column is representing char '&Ccedil' (Ç),
Tr_charno16	INTEGER	total no. of '&Ccedil' was found in the Url,
Tr_word0	TEXT	column is representing char 'Anadolu',
Tr_wordno0	INTEGER	total no. of 'Anadolu' was found in the Url,
Tr_word1	TEXT	column is representing char 'anasayfa',
Tr_wordno1	INTEGER	total no. of 'anasayfa' was found in the Url,
Tr_word2	TEXT	column is representing char 'ana sayfa',
Tr_wordno2	INTEGER	total no. of 'ana sayfa ' was found in the Url,
Tr_word3	TEXT	column is representing char 'arama',
Tr_wordno3	INTEGER	total no. of 'arama' was found in the Url,
Tr_word4	TEXT	column is representing char 'arkadaş',
Tr_wordno4	INTEGER	total no. of 'arkadaş' was found in the Url,
Tr_word5	TEXT	column is representing char 'dış',
Tr_wordno5	INTEGER	total no. of 'dış' was found in the Url,
Tr_word6	TEXT	column is representing char 'eğitim',
Tr_wordno6	INTEGER	total no. of 'eğitim' was found in the Url,
Tr_word7	TEXT	column is representing char 'firma',
Tr_wordno7	INTEGER	total no. of 'firma ' was found in the Url,
Tr_word8	TEXT	column is representing char 'hizmet',
Tr_wordno8	INTEGER	total no. of 'hizmet ' was found in the Url,
Tr_word9	TEXT	column is representing char 'geri',
Tr_wordno9	INTEGER	total no. of 'geri' was found in the Url,
Tr_word10	TEXT	column is representing char 'İç',
Tr_wordno10	INTEGER	total no. of 'İç ' was found in the Url,
Tr_word11	TEXT	column is representing char 'İleri',
Tr_wordno11	INTEGER	total no. of 'İleri ' was found in the Url,
Tr_word12	TEXT	column is representing char 'iletişim',
Tr_wordno12	INTEGER	total no. of 'iletişim ' was found in the Url,
Tr_word13	TEXT	column is representing char 'Şti',
Tr_wordno13	INTEGER	total no. of 'Şti' was found in the Url,
Tr_word14	TEXT	column is representing char 'ticaret',
Tr_wordno14	INTEGER	total no. of 'ticaret' was found in the Url,
Tr_word15	TEXT	column is representing char 'türk',
Tr_wordno15	INTEGER	total no. of 'türk' was found in the Url,
Tr_word16	TEXT	column is representing char 'turizm',
Tr_wordno16	INTEGER	total no. of 'turizm ' was found in the Url,
Tr_word17	TEXT	column is representing char 'türkiye',
Tr_wordno17	INTEGER	total no. of 'türkiye ' was found in the Url,

Table 18 Continued.

Field	*Attribute*	*Description*
Tr_word18	TEXT	column is representing char 'Üye',
Tr_wordno18	INTEGER	total no. of 'Üye' was found in the Url,
Tr_word19	TEXT	column is representing char 'yardım'
Tr_wordno19	INTEGER	total no. of 'yardım' was found in the Url,
Msg_status	INTEGER	error code of response message which was taken from Url/ server,
Msg_reason	TEXT	error message which was taken from Url/ server,
Query_duration	INTEGER	total time spent at the addresses within the same Url.,
Depth_level	INTEGER	maximum depth which was inspected at the rightmost native address,
Resourse	TEXT	shows resource name where queries has been done,
Search_key	TEXT	shows search key which is queried from Recourse,
Text_editor	TEXT	name of text editor used for inquiring Recourse,
Lastpageno	INTEGER	last page of Resource reached.

First three columns of table (Root, Date_inserted, Hostname_current) are designed to be "composite prime key". Each row contains 95 columns including fields of composite prime key.

Table 19: web.db (used for 50 keywords)

Field	***Attribute***	***Description***
Root	TEXT	name of the root URL,
Date_inserted	TEXT	date value (yy:mm:dd:hh:mm:ss) shows the time the row was inserted,
Hostname_current	TEXT	the name of the host on which script run,
Hostip_address	TEXT	the ip address of the host,
Root_ext	TEXT	extension of the Url which shows generic top-level domain,
Nation	TEXT	extension of Url which shows country code top-level domain,
Urlsize	INTEGER	total size of Url including size of internal addresses,
Urlwidth	INTEGER	total no. of links at the first level,
No_of_internal_links	INTEGER	no. of internal links addresses used in Url,
No_of_external_links	INTEGER	no. of external links in the Url,
Count_href	INTEGER	total no. of href documents referenced within Url,
Count_img	INTEGER	total no. of images referenced within Url,
Count_form	INTEGER	total no. of forms referenced within Url,
Tr_word0	TEXT	column is representing char 'bir',
Tr_wordno0	INTEGER	total no. of 'bir' was found in the Url,
Tr_word1	TEXT	column is representing char 'olmak',
Tr_wordno1	INTEGER	total no. of 'olmak' was found in the Url,
Tr_word2	TEXT	column is representing char 'için',
Tr_wordno2	INTEGER	total no. of 'için ' was found in the Url,
Tr_word3	TEXT	column is representing char 'ben',
Tr_wordno3	INTEGER	total no. of 'ben' was found in the Url,
Tr_word4	TEXT	column is representing char 'demek',
Tr_wordno4	INTEGER	total no. of 'demek' was found in the Url,
Tr_word5	TEXT	column is representing char 'çok',
Tr_wordno5	INTEGER	total no. of 'çok' was found in the Url,
Tr_word6	TEXT	column is representing char 'yapmak',
Tr_wordno6	INTEGER	total no. of 'yapmak' was found in the Url,
Tr_word7	TEXT	column is representing char 'gibi',
Tr_wordno7	INTEGER	total no. of 'gibi' was found in the Url,
Tr_word8	TEXT	column is representing char 'daha',
Tr_wordno8	INTEGER	total no. of 'daha' was found in the Url,
Tr_word9	TEXT	column is representing char 'almak',
Tr_wordno9	INTEGER	total no. of 'almak' was found in the Url,
Tr_word10	TEXT	column is representing char 'var',
Tr_wordno10	INTEGER	total no. of 'var' was found in the Url,

Table 19 Continued.

Field	*Attribute*	*Description*
Tr_word11	TEXT	column is representing char 'kendi',
Tr_wordno11	INTEGER	total no. of 'kendi ' was found in the Url,
Tr_word12	TEXT	column is representing char 'gelmek'
Tr_wordno12	INTEGER	total no. of 'gelmek ' was found in the Url,
Tr_word13	TEXT	column is representing char 'ile',
Tr_wordno13	INTEGER	total no. of 'ile' was found in the Url,
Tr_word14	TEXT	column is representing char 'vermek',
Tr_wordno14	INTEGER	total no. of 'vermek' was found in the Url,
Tr_word15	TEXT	column is representing char 'ama',
Tr_wordno15	INTEGER	total no. of 'ama' was found in the Url,
Tr_word16	TEXT	column is representing char 'sonra',
Tr_wordno16	INTEGER	total no. of 'sonra ' was found in the Url,
Tr_word17	TEXT	column is representing char 'kadar',
Tr_wordno17	INTEGER	total no. of 'kadar' was found in the Url,
Tr_word18	TEXT	column is representing char 'yer',
Tr_wordno18	INTEGER	total no. of 'yer' was found in the Url,
Tr_word19	TEXT	column is representing char 'insan'
Tr_wordno19	INTEGER	total no. of 'insan' was found in the Url,
Tr_word20	TEXT	column is representing char 'değil'
Tr_wordno20	INTEGER	total no. of 'değil' was found in the Url,
Tr_word21	TEXT	column is representing char 'her',
Tr_wordno21	INTEGER	total no. of 'her' was found in the Url,
Tr_word22	TEXT	column is representing char 'istemek',
Tr_wordno22	INTEGER	total no. of 'istemek' was found in the Url,
Tr_word23	TEXT	column is representing char 'yıl',
Tr_wordno23	INTEGER	total no. of 'yıl' was found in the Url,
Tr_word24	TEXT	column is representing char 'çıkmak',
Tr_wordno24	INTEGER	total no. of 'çıkmak' was found in the Url,
Tr_word25	TEXT	column is representing char 'görmek',
Tr_wordno25	INTEGER	total no. of 'görmek ' was found in the Url,
Tr_word26	TEXT	column is representing char 'gün',
Tr_wordno26	INTEGER	total no. of 'gün' was found in the Url,
Tr_word27	TEXT	column is representing char 'biz',
Tr_wordno27	INTEGER	total no. of 'biz ' was found in the Url,
Tr_word28	TEXT	column is representing char 'gitmek',
Tr_wordno28	INTEGER	total no. of 'gitmek' was found in the Url,
Tr_word29	TEXT	column is representing char 'iş',
Tr_wordno29	INTEGER	total no. of 'iş' was found in the Url,
Tr_word30	TEXT	column is representing char 'şey',
Tr_wordno30	INTEGER	total no. of 'şey' was found in the Url,
Tr_word31	TEXT	column is representing char 'ara',
Tr_wordno31	INTEGER	total no. of 'ara' was found in the Url,
Tr_word32	TEXT	column is representing char 'bilmek',
Tr_wordno32	INTEGER	total no. of 'bilmek' was found in the Url,
Tr_word33	TEXT	column is representing char 'zaman',
Tr_wordno33	INTEGER	total no. of 'zaman' was found in the Url,
Tr_word34	TEXT	column is representing char 'çocuk',
Tr_wordno34	INTEGER	total no. of 'çocuk' was found in the Url,

Table 19 Continued.

Field	*Attribute*	*Description*
Tr_word35	TEXT	column is representing char 'iki',
Tr_wordno35	INTEGER	total no. of 'iki' was found in the Url,
Tr_word36	TEXT	column is representing char 'bakmak',
Tr_wordno36	INTEGER	total no. of 'bakmak' was found in the Url,
Tr_word37	TEXT	column is representing char 'çalışmak',
Tr_wordno37	INTEGER	total no. of 'çalışmak' was found in the Url,
Tr_word38	TEXT	column is representing char 'içinde',
Tr_wordno38	INTEGER	total no. of 'içinde' was found in the Url,
Tr_word39	TEXT	column is representing char 'büyük',
Tr_wordno39	INTEGER	total no. of 'büyük' was found in the Url,
Tr_word40	TEXT	column is representing char 'yok',
Tr_wordno40	INTEGER	total no. of 'yok' was found in the Url,
Tr_word41	TEXT	column is representing char 'başlamak',
Tr_wordno41	INTEGER	total no. of 'başlamak' was found in the Url,
Tr_word42	TEXT	column is representing char 'yol',
Tr_wordno42	INTEGER	total no. of 'yol' was found in the Url,
Tr_word43	TEXT	column is representing char 'kalmak',
Tr_wordno43	INTEGER	total no. of 'kalmak' was found in the Url,
Tr_word44	TEXT	column is representing char 'neden',
Tr_wordno44	INTEGER	total no. of 'neden' was found in the Url,
Tr_word45	TEXT	column is representing char 'siz',
Tr_wordno45	INTEGER	total no. of 'siz' was found in the Url,
Tr_word46	TEXT	column is representing char 'konu',
Tr_wordno46	INTEGER	total no. of 'konu' was found in the Url,
Tr_word47	TEXT	column is representing char 'yapılmak',
Tr_wordno47	INTEGER	total no. of 'yapılmak' was found in the Url,
Tr_word48	TEXT	column is representing char 'iyi',
Tr_wordno48	INTEGER	total no. of 'iyi ' was found in the Url,
Tr_word49	TEXT	column is representing char 'kadın',
Tr_wordno49	INTEGER	total no. of 'kadın ' was found in the Url,
Msg_status	INTEGER	error code of response message which was taken from Url/ server,
Msg_reason	TEXT	error message which was taken from Url/ server,
Query_duration	INTEGER	total time spent at the addresses within the same Url.,
Depth_level	INTEGER	maximum depth which was inspected at the rightmost native address,
Resourse	TEXT	shows resource name where queries has been done,
Search_key	TEXT	shows search key which is queried from Recourse,
Text_editor	TEXT	name of text editor used for inquiring Recourse,
Lastpageno	INTEGER	last page of Resource reached.

First four columns of Table 19 (Root, Date_inserted, Hostname_current, Error_count) are designed to be "composite prime key". Each row contains 9 columns including fields of composite prime key.

Table 20: log_error.db

Field	***Attribute***	***Description***
Root	TEXT	name of the root URL,
Date_inserted	TEXT	date value (yy:mm:dd:hh:mm:ss) shows the time the row was inserted,
Hostname_current	TEXT	the name of the host on which script run,
Hostip_address	TEXT	the ip address of the host
Recourse	TEXT	shows resource name where queries has been done,
Search_key	TEXT	shows search key which is queried from Recourse,
Error_count	INTEGER	defines an error number which is incremented where an exception is encountered when inquiring an Url.,
Msg_status	INTEGER	status of exception which is encountered,
Msg_reason	TEXT	status information of exception which is encountered.

First four columns of Table 20 (Resourse, Search_key, Date_inserted, Hostname_current) are designed to be "composite prime key". Each row contains 19 columns including fields of composite prime key.

Table 21: Exitstat.db

Field	***Attribute***	***Description***
Resourse	TEXT	shows resource name where queries has been done,
Search_key	TEXT	shows search key which is queried from resourse,
Date_inserted	TEXT	date value (year:month:day:hour:minute:second) shows the time the row was inserted,
Hostname_current	TEXT	the name of the host on which script run,
Hostip_address	TEXT	the ip address of the host,
Lastpageno	INTEGER	is the maximum number of resource page to be inquired.
Entery_year	INTEGER	year of last run started,
Entery_month	INTEGER	month of last run started,
Entery_day	INTEGER	day value of last run started,
Exit_day	INTEGER	day value of last run ended,
Day_differ	INTEGER	difference between Exit_day and Entery_day,
Entery_hour	INTEGER	hour value of last run started,
Exit_hour	INTEGER	hour value of last run ended,
Hour_differ	INTEGER	difference between Exit_hour and Entery_hour,
Entery_minute	INTEGER	minute value of last run started,
Exit_minute	INTEGER	minute value of last run ended,
Minute_differ	INTEGER	difference between Exit_minute and Entery_minute,
Entery_second	INTEGER	second value of last run started,
Exit_second	INTEGER	second value of last run ended,
Second_differ	INTEGER	difference between Exit_second and Entery_second.

First three columns of Table 21 (CShort, WRank, Name) are designed to be "composite prime key". Each row contains 5 columns including fields of composite prime key.

Table 22: Alexa.db

Field	***Attribute***	***Description***
CShort	TEXT	The country code of the website is registered,
WRank	INTEGER	Global rank at the top list,
Name	TEXT	Name of the website,
CRank	INTEGER	Rank of the website in the country where it is registered,
Linkin	INTEGER	Number of reference links to the website.

First two columns of Table 22 (CShort, CLong) are designed to be "composite prime key". Each row contains 5 columns including fields of composite prime key.

Table 23: Summary.db

Field	*Attribute*	*Description*
CShort	TEXT	The country code of the website is registered,
CLong	TEXT	The country name of the website is registered,
Category_1000	INTEGER	Sum of website at first 1000 top list,
Category_1000_value	INTEGER	Sum of (1.000.000 - website position) at first 1000 top list,
Category_1000_linksin	INTEGER	Sum of websites' Linksin values at first 1000 top list,
Average_1000	INTEGER	Mean of websites' Linksin values at first 1000 top list,
Category_10000	INTEGER	Sum of website in between 1000-10000 of top list,
Category_10000_value	INTEGER	Sum of (1.000.000 - website position) at 1000-10000 of top list,
Category_10000_linksin	INTEGER	Sum of websites' Linksin values at 1000-10000 of top list,
Average_10000	INTEGER	Mean of websites' Linksin values at 1000-10000 of top list,
Category_100000	INTEGER	Sum of website in between 10000-100000 of top list,
Category_100000_value	INTEGER	Sum of (1.000.000 - website position) at 10000-100000 of top list,
Category_100000_linksin	INTEGER	Sum of websites' Linksin values at 10000-100000 of top list,
Average_100000	INTEGER	Mean of websites' Linksin values at 10000-100000 of top list,
Category_1000000	INTEGER	Sum of website in between 100000-1000000 of top list,
Category_1000000_value	INTEGER	Sum of (1.000.000 - website position) at 100000-1000000 of top list,
Category_1000000_linksin	INTEGER	Sum of websites' Linksin values at 100000-1000000 of top list,
Average_1000000	INTEGER	Mean of websites' Linksin values at 100000-1000000 of top list,

APPENDIX B

KEYWORDS AND NUMBER OF OCCURRENCES

Table 24: Occurences of Keywords

Keywords / Turkey (.tr)	# occurence	Italy (.it)	# occurence	Poland (.pl)	# occurence
ana sayfa	88.300.000	home page	434.000.000	strona główna	69.900.000
anasayfa	222.000.000	homepage	676.000.000	dom	228.000.000
Anadolu	26.400.000				
arama	117.000.000	ricerca	209.000.000	Szukać / szukaj	165.000.000
arkadaş	36.900.000				
a.ş. / ltd.	14.200.000	Società per Azioni (S.p.A. OR SpA)	625000000 (273000000 / 266000000)	spółki akcyjnej	1.550.000
bizi arayın	1.210.000	contattaci	36.000.000	kontakt	697.000.000
Dış	127.000.000				
dış ticaret	3.040.000	commercio estero	9.140.000	handlu zagranicznego	799.000
eğitim	49.900.000	formazione	28.700.000	trenować	512.000
firma	222.000.000	società	49.700.000	firma	273.000.000
geri	139.000.000	indietro	103.000.000	z powrotem	2.390.000
hızlı erişim	5.030.000	accesso rapido	479.000	szybki dostęp	1.570.000
hizmet	23.600.000	servizi	136.000.000	usług	18.600.000
İç	148.000.000				
ihracat	4.080.000	esportazioni	615.000	eksport	5.560.000
ileri	18.300.000	avanti	37.000.000	w przód	2.680.000

Table 24 Continued.

Keywords / Turkey (.tr)	# occurence	Italy (.it)	# occurence	Poland (.pl)	# occurence
iletişim	82.300.000	comunicazione	28.700.000	komunikacji	6.670.000
imalat	3.370.000	fabbricazione	1.860.000	produkcji	18.000.000
italya		Italia	372.000.000		
italyan		Italiano	544.000.000		
ithalat	3.030.000	importazioni	671.000	przywóz	165.000
otel	20.400.000	hotel	841.000.000	Zdjęcie	71.900.000
polonyalı				Polaków	8.730.000
polonya				Polska	164.000.000
sıkça	6.440.000				
site haritası	7.880.000	mappa del sito	14.500.000	mapa serwisu	14.300.000
Şti.	54.600.000	società	49.800.000	firma	277.000.000
Şirketi	13.500.000				
ticaret	20.900.000	commercio	22.700.000	handel	78.200.000
turizm	17.800.000	turismo	383.000.000	turystyka	28.300.000
türk	129.000.000				
türkiye	136.000.000				
Üye	154.000.000				
yardım	70.000.000	aiuto	65.600.000	pomoc	144.000.000

APPENDIX C

GENERAL FLOW OF JOB STREAM

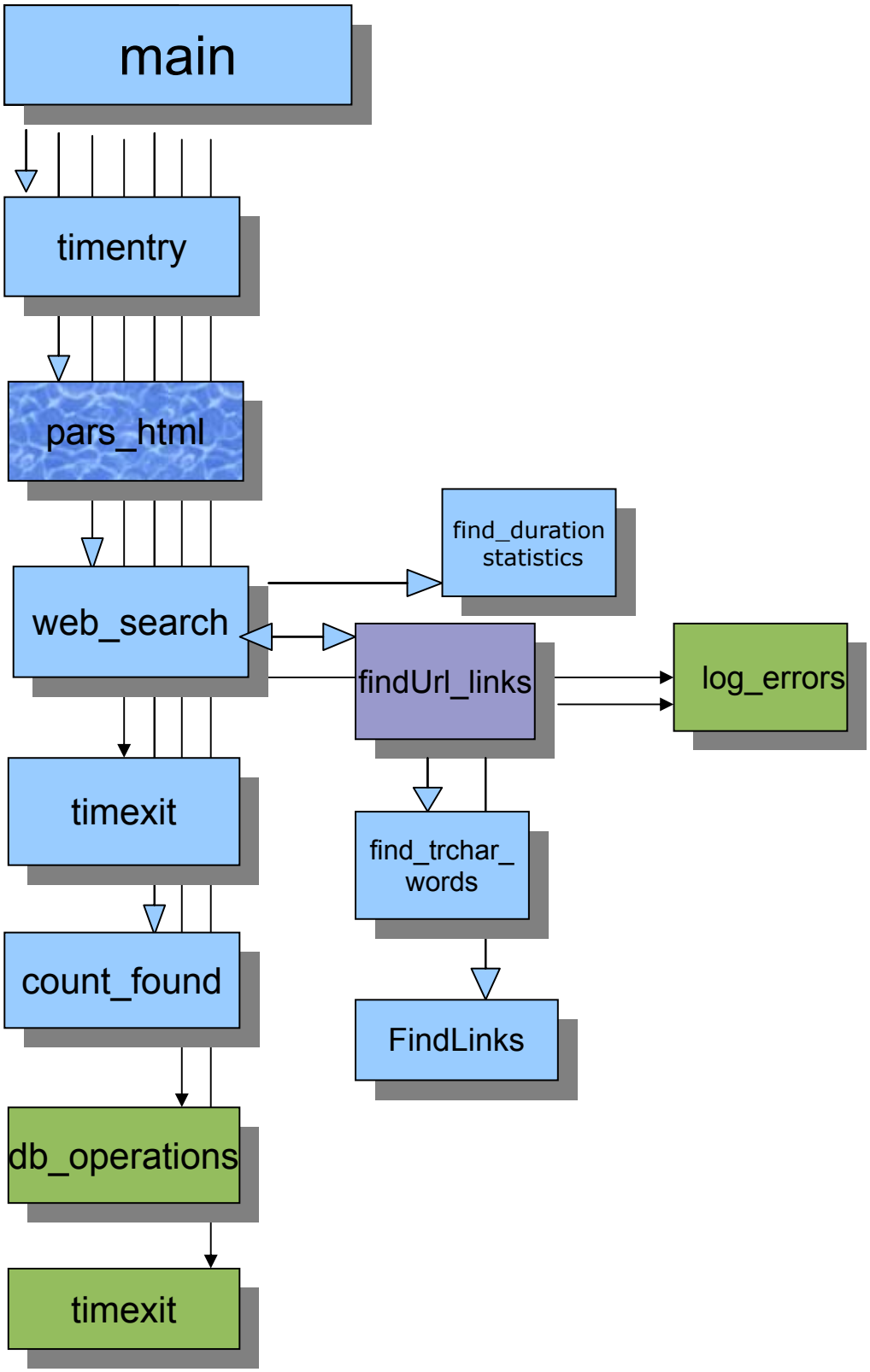

Figure 19: General flow of job stream

APPENDIX D

ALTERNATIVE SCENARIOS USED

Table 25: Sample for Google Search (used for 19 keywords)

Resource	keyword	max.page	max.depth	max.duration
google	anadolu	1000	7	600
google	anasayfa	1000	7	600
google	arama	1000	7	600
google	arkadaş	1000	7	600
google	dış	1000	7	600
google	eğitim	1000	7	600
google	firma	1000	7	600
google	hizmet	1000	7	600
google	geri	1000	7	600
google	İç	1000	7	600
google	İleri	1000	7	600
google	iletişim	1000	7	600
google	Şti	1000	7	600
google	ticaret	1000	7	600
google	Türk	1000	7	600
google	turizm	1000	7	600
google	Türkiye	1000	7	600
google	Üye	1000	7	600
google	yardım	1000	7	600
google	anadolu+anasayfa	1000	7	600
google	anadolu+arama	1000	7	600
google	anadolu+arkadaş	1000	7	600
google	anadolu+dış	1000	7	600
google	anadolu+eğitim	1000	7	600
google	anadolu+firma	1000	7	600
google	anadolu+hizmet	1000	7	600
google	anadolu+geri	1000	7	600
google	anadolu+İç	1000	7	600
google	anadolu+İleri	1000	7	600
google	anadolu+iletişim	1000	7	600
google	anadolu+Şti	1000	7	600
google	anadolu+ticaret	1000	7	600
google	anadolu+Türk	1000	7	600
google	anadolu+turizm	1000	7	600
google	anadolu+Türkiye	1000	7	600
google	anadolu+Üye	1000	7	600
google	anadolu+yardım	1000	7	600
google	anasayfa+arama	1000	7	600
google	anasayfa+arkadaş	1000	7	600
google	anasayfa+dış	1000	7	600

Table 25 Continued.

Resource	keyword	max.page	max.depth	max.duration
...	...	...	...	...
google	İç+Türk	1000	7	600
google	İç+turizm	1000	7	600
google	İç+Türkiye	1000	7	600
google	İç+Üye	1000	7	600
google	İç+yardım	1000	7	600
google	İleri+iletişim	1000	7	600
google	İleri+Şti	1000	7	600
google	İleri+ticaret	1000	7	600
google	İleri+Türk	1000	7	600
google	İleri+turizm	1000	7	600
google	İleri+Türkiye	1000	7	600
google	İleri+Üye	1000	7	600
google	İleri+yardım	1000	7	600
google	iletişim+Şti	1000	7	600
google	iletişim+ticaret	1000	7	600
google	iletişim+Türk	1000	7	600
google	iletişim+turizm	1000	7	600
google	iletişim+Türkiye	1000	7	600
google	iletişim+Üye	1000	7	600
google	iletişim+yardım	1000	7	600
google	Şti+ticaret	1000	7	600
google	Şti+Türk	1000	7	600
google	Şti+turizm	1000	7	600
google	Şti+Türkiye	1000	7	600
google	Şti+Üye	1000	7	600
google	Şti+yardım	1000	7	600
google	ticaret+Türk	1000	7	600
google	ticaret+turizm	1000	7	600
google	ticaret+Türkiye	1000	7	600
google	ticaret+Üye	1000	7	600
google	ticaret+yardım	1000	7	600
google	Türk+turizm	1000	7	600
google	Türk+Türkiye	1000	7	600
google	Türk+Üye	1000	7	600
google	Türk+yardım	1000	7	600
google	turizm+Türkiye	1000	7	600
google	turizm+Üye	1000	7	600
google	turizm+yardım	1000	7	600
google	Türkiye+Üye	1000	7	600
google	Türkiye+yardım	1000	7	600
google	Üye+yardım	1000	7	600

Table 26: Sample for Google Search (used for 50 keywords)

Resource	keyword	max.page	max.depth	max.duration
google	bir	1000	7	600
google	olmak	1000	7	600
google	için	1000	7	600
google	ben	1000	7	600
google	demek	1000	7	600
google	çok	1000	7	600
google	yapmak	1000	7	600
google	gibi	1000	7	600
google	daha	1000	7	600
google	almak	1000	7	600
google	var	1000	7	600
google	kendi	1000	7	600
google	gelmek	1000	7	600
google	ile	1000	7	600
google	vermek	1000	7	600
google	ama	1000	7	600
google	sonra	1000	7	600
google	kadar	1000	7	600
google	yer	1000	7	600
google	insan	1000	7	600
google	değil	1000	7	600
google	her	1000	7	600
google	istemek	1000	7	600
google	yıl	1000	7	600
google	çıkmak	1000	7	600
google	görmek	1000	7	600
google	gün	1000	7	600
google	biz	1000	7	600
google	gitmek	1000	7	600
google	iş	1000	7	600
google	şey	1000	7	600
google	ara	1000	7	600
google	bilmek	1000	7	600
google	zaman	1000	7	600
google	çocuk	1000	7	600
google	iki	1000	7	600
google	bakmak	1000	7	600
google	çalışmak	1000	7	600
google	içinde	1000	7	600
google	büyük	1000	7	600
google	yok	1000	7	600
google	başlamak	1000	7	600
google	yol	1000	7	600
google	kalmak	1000	7	600
google	neden	1000	7	600
google	siz	1000	7	600
google	konu	1000	7	600
google	yapılmak	1000	7	600
google	iyi	1000	7	600
google	kadın	1000	7	600

APPENDIX E

LIST OF SCRIPTS

Web crawling script

```
#! /usr/bin/python
# -*- coding: utf8 -*-             #!/home1/software/application/python-2.5.4/bin/python
import sgmllib, urllib, urlparse
from sgmllib import SGMLParser
import string
import os
import sys
import exceptions
import HTMLParser
import htmllib, formatter
import unicodedata
import re
import sqlite3 as dbapi
from time
import gmtime, strftime, localtime
import subprocess
from subprocess import Popen
import socket
import urllib2
from urllib2 import Request, urlopen, URLError
#
def find_trchar_words(url, searchStr):       # /called by "findUrl_links"
#################################################################
#                                                               #
#  searches given turkish chars and words at the string sent     #
#                                                                #
#################################################################
#
       global tr_char, tr_char_u, tr_char_utf8, tr_word, tr_word_u, tr_word_utf8
              # <<< 14.09.2009
       tr_char =
{'ý':0,'ı':0,'ğ':0,'ü':0,'ş':0,'ö':0,'ç':0,'Ğ':0,'Ü':0,'Ş':0,'İ':0,'Ö':0,'Ç':0,'&uuml;':0
,'&ouml;':0, 'ccedil;':0,'&Ccedil;':0}
                           # <<< 14.09.2009
       tr_char_u =
{u'ý':0,u'ı':0,u'ğ':0,u'ü':0,u'ş':0,u'ö':0,u'ç':0,u'Ğ':0,u'Ü':0,u'Ş':0,u'İ':0,u'Ö':0,u'Ç'
:0,u'&uuml;':0, u'&ouml;':0,u'ccedil;':0,u'&Ccedil;':0}     # <<< 14.09.2009
       tr_char_utf8 =
{u'ý':0,u'ı':0,u'ğ':0,u'ü':0,u'ş':0,u'ö':0,u'ç':0,u'Ğ':0,u'Ü':0,u'Ş':0,u'İ':0,u'Ö':0,u'Ç'
:0,u'&uuml;':0, u'&ouml;':0,u'ccedil;':0,u'&Ccedil;':0}     # <<< 14.09.2009
#
       tr_word = {'Anadolu':0,'anasayfa':0,'ana
sayfa':0,'arama':0,'arkadaş':0,'dış':0,'eğitim':0,'firma':0,'hizmet':0,
       'geri':0,'İç':0,'İleri':0,'iletişim':0,'Şti':0,'ticaret':0,'Türk':0,'turizm':0,'T
ürkiye':0,'Üye':0,'yardım':0}         # <<< 14.09.2009
       tr_word_u = {u'Anadolu':0,u'anasayfa':0,u'ana
sayfa':0,u'arama':0,u'arkadaş':0,u'dış':0,u'eğitim':0,u'firma':0,
       u'hizmet':0,u'geri':0,u'İç':0,u'İleri':0,u'iletişim':0,u'Şti':0,u'ticaret':0,u'Tü
rk':0,u'turizm':0,u'Türkiye':0,
       u'Üye':0,u'yardım':0}                        # <<< 14.09.2009
       tr_word_utf8 = {u'Anadolu':0,u'anasayfa':0,u'ana
sayfa':0,u'arama':0,u'arkadaş':0,u'dış':0,u'eğitim':0,u'firma':0,
       u'hizmet':0,u'geri':0,u'İç':0,u'İleri':0,u'iletişim':0,u'Şti':0,u'ticaret':0,u'Tü
rk':0,u'turizm':0,u'Türkiye':0,
       u'Üye':0,u'yardım':0}                        # <<< 14.09.2009
#******  search for tr char.s ******
       tr_char_len = 0
       for x in tr_char:
             x_tr = x
             xtr = x_tr
```

```
            if len(re.findall(xtr,searchStr)) > 0:
                    tr_char_len = tr_char_len + len(re.findall(xtr,searchStr))
                    tr_char[x] = tr_char[x] + len(re.findall(xtr,searchStr))
#                   print "\n  tr_char>", xtr
#                   print " # occurence is:", len(re.findall(xtr,searchStr))
#                   print "\n *** tr_char length=", tr_char_len
       tr_char_ulen = 0
      for x in tr_char_u:
            xtr = x
           if len(re.findall(xtr,searchStr)) > 0:
                    tr_char_ulen = tr_char_ulen + len(re.findall(xtr,searchStr))
                    tr_char_u[x] = tr_char_u[x] + len(re.findall(xtr,searchStr))
#                   print "\n  tr_char_u >", xtr
#                   print " # occurence is:", len(re.findall(xtr,searchStr))

      tr_char_utf8len = 0
     for x in tr_char_utf8:
            x_tr = x
           xtr = x_tr.encode('utf-8')
           if len(re.findall(xtr,searchStr)) > 0:
                    tr_char_utf8len = tr_char_utf8len + len(re.findall(xtr,searchStr))
                    tr_char_utf8[x] = tr_char_utf8[x] + len(re.findall(xtr,searchStr))
#                   print "\n  tr_char_utf8>", xtr
#                   print " # occurence is:", len(re.findall(xtr,searchStr))
#
#  ********  search for words ********
       tr_word_len = 0
       for x in tr_word:
#           print "\n x>", x
            if len(re.findall(x,searchStr)) > 0:
                    tr_word_len = tr_word_len + len(re.findall(x,searchStr))
                    tr_word[x] = tr_word[x] + len(re.findall(x,searchStr))
#                   print "\n # of ", x, " counted as %d in tr_word " %
len(re.findall(x,searchStr))

       tr_word_ulen = 0
       for x in tr_word_u:
            xtr = x
            if len(re.findall(xtr,searchStr)) > 0:
                    tr_word_ulen = tr_word_ulen + len(re.findall(xtr,searchStr))
       # <<< 22.09.2009
                    tr_word_u[x] = tr_word_u[x] + len(re.findall(xtr,searchStr))
       # <<< 22.09.2009
#                   print "\n # of ", x, " counted as %d in tr_word_u " %
len(re.findall(xtr,searchStr))

       tr_word_utf8len = 0
       for x in tr_word_utf8:
            x_tr = x
            xtr = x_tr.encode('utf-8')
#           print "\n x>", x, "  x_tr>", x_tr, " x_tr.encode('utf-8')>", xtr
            if len(re.findall(xtr,searchStr)) > 0:
                    tr_word_utf8len = tr_word_utf8len + len(re.findall(xtr,searchStr))
                    tr_word_utf8[x] = tr_word_utf8[x] + len(re.findall(xtr,searchStr))
#                   print "\n # of ", xtr, " counted as %d in tr_word_utf8 " %
len(re.findall(xtr,searchStr))
#                   print "\n *** tr_word_utf8 length=", tr_word_utf8len
```

```
#
def count_found():            # called by "main"
#############################################################################
#                                                                           #
#  counts and finds in which list has max. # turkish chars and words.      #
#                                                                           #
#############################################################################
#
        global trchar_keys, trchar_values, trword_keys, trword_values
         count_trchar = 0
         count_trchar_u = 0
         count_trchar_utf8 = 0
            trchar_keys = []
         trchar_values = []
            trword_keys = []
         trword_values = []
#
         for x in trchar:
               count_trchar = count_trchar + trchar[x]
#       print "\n  count_trchar>", count_trchar
#
         for x in trchar_u:
               count_trchar_u = count_trchar_u + trchar_u[x]
#       print "\n  count_trchar_u>", count_trchar_u
#
         for x in trchar_utf8:
               count_trchar_utf8 = count_trchar_utf8 + trchar_utf8[x]
#       print "\n  count_trchar_utf8>", count_trchar_utf8
#
        if count_trchar > count_trchar_u:
         if count_trchar > count_trchar_utf8:      # count_trchar >= all
               trchar_keys = trchar.keys()
            trchar_values = trchar.values()
#             print "\n *** max. count occured in trchar %d" % count_trchar, "  in ",
groot
#             print "\n *** trchar >>>", trchar
         else:                                     # count_trchar_utf8 > all
               trchar_keys = trchar_utf8.keys()
            trchar_values = trchar_utf8.values()
#             print "\n *** max. count occured in trchar_utf8 is %d " %
count_trchar_utf8, "  in ", groot
#             print "\n *** trchar_utf8 >>>", trchar_utf8
       else:                                       # count_trchar_u > trchar
          if count_trchar_u > count_trchar_utf8:   # count_trchar_u > all
               trchar_keys = trchar_u.keys()
            trchar_values = trchar_u.values()
#             print "\n *** max. count occured in trchar_u %d" % count_trchar_u, "  in ",
groot
#             print "\n *** trchar >>>", trchar_u
            else:
               trchar_keys = trchar_utf8.keys()
            trchar_values = trchar_utf8.values()
#             print "\n *** max. count occured in trchar_utf8 is %d " %
count_trchar_utf8, "  in ", groot
#             print "\n *** trchar_utf8 >>>", trchar_utf8
#
#       print "\n trchar_keys >", trchar_keys
#       print "\n trchar_values >", trchar_values
#
     count_trword = 0
     count_trword_u = 0
     count_trword_utf8 = 0
       for x in trword:
             count_trword = count_trword + trword[x]
#       print "\n  count_trword>", count_trword
#
        for x in trword_u:
              count_trword_u = count_trword_u + trword_u[x]
#       print "\n  count_trword_u>", count_trword_u
#
        for x in trword_utf8:
              count_trword_utf8 = count_trword_utf8 + trword_utf8[x]
#       print "\n  count_trword_utf8>", count_trword_utf8
#
        if count_trword > count_trword_u:
         if count_trword > count_trword_utf8:      # count_trword >= all
```

```
            trword_keys = trword.keys()
            trword_values = trword.values()
#           print "\n *** max. count occured in trword is %d" % count_trword, "  in ",
groot
#           print "\n *** trword >>>", trword
        else:                                           # count_trword_utf8 > all
            trword_keys = trword_utf8.keys()
            trword_values = trword_utf8.values()
#           print "\n *** max. count occured in trword_utf8 is %d " %
count_trword_utf8, "  in ", groot
#           print "\n *** trword_utf8 >>>", trword_utf8
      else:                                             # count_trword_u > trword
         if count_trword_u > count_trword_utf8:         # count_trword_u > all
            trword_keys = trword_u.keys()
            trword_values = trword_u.values()
#           print "\n *** max. count occured in trword_u is %d" % count_trword_u, "  in
", groot
#           print "\n *** trword_u >>>", trword_u
          else:
            trword_keys = trword_utf8.keys()
            trword_values = trword_utf8.values()
#           print "\n *** max. count occured in trword_utf8 is %d " %
count_trword_utf8, "  in ", groot
#           print "\n *** trword_utf8 >>>", trword_utf8
#
#       print "\n trword_keys >", trword_keys
#       print "\n trword_values >", trword_values
#
def timentry():               #  called by "main"
#############################################################
#                                                           #
#  gives time stamp when query starts at the Url.     #
#                                                           #
#############################################################
#
        print "\n localtime():", localtime()
        (year, month, day, hour, minute, second, weekday, dayofyear, dst) = localtime()
        entery_year = year
        entery_month = month
        entery_day = day
        entery_hour = hour
        entery_minute = minute
        entery_second = second
        print "\n entery hour:", entery_hour, " entery minute:", entery_minute, " entery
second:", entery_second
        epocha = localtime()
        return entery_year, entery_month, entery_day, entery_hour, entery_minute,
entery_second       # <<< 22.09.2009
#
def timexit(entery_year, entery_month, entery_day, entery_hour, entery_minute,
entery_second, lastpageno):   # <<< 22.09.2009
#       called by "main"
#############################################################################
#                                                                           #
#  gives time stamp when query ends and query time span at the Url.     #
#                                                                           #
#############################################################################
#
        global exit_year, exit_month, exit_day, exit_hour, exit_minute, exit_second
        (year, month, day, hour, minute, second, weekday, dayofyear, dst) = localtime()
        exit_year = year
        exit_month = month
        exit_day = day
        exit_hour = hour
        exit_minute = minute
        exit_second = second
        print "\n entery_day:", entery_day, " entery hour:", entery_hour, " entery
minute:", entery_minute, " entery second:", entery_second
        print "\n   exit_day:", exit_day,   "   exit hour:", exit_hour,   "   exit
minute:", exit_minute,   "   exit second:", exit_second
#
        epochb = localtime()
#
#       print year, month, day, hour, minute, second, weekday, dayofyear, dst
#
        if entery_second > exit_second:
```

```
                exit_second = exit_second + 60
                if exit_minute > 0:
                        exit_minute = exit_minute -1
             else:
                        exit_minute = 59
                        exit_hour = exit_hour -1
        if entery_minute > exit_minute:
                exit_minute = exit_minute + 60
               exit_hour = exit_hour -1
        if entery_hour > exit_hour:
                exit_hour = exit_hour + 24
                exit_day = exit_day -1
        if entery_day > exit_day + 1:                           # <<< 22.09.2009
                entety_day = exit_day                           # <<< 22.09.2009
     else:                                                      # <<< 22.09.2009
                entety_day = entery_day                         # <<< 22.09.2009

        print "\n   day dif.:", exit_day-entety_day,"    hour dif.:", exit_hour-
entery_hour, "    minute dif.:", exit_minute-entery_minute,\
               "    second dif.:", exit_second-entery_second
#
        day_differ = exit_day-entety_day
        hour_differ = exit_hour-entery_hour
        minute_differ = exit_minute-entery_minute
        second_differ = exit_second-entery_second
#
        if exit_second >= 60:                                   # <<< 22.09.2009
              exit_second = exit_second - 60          # <<< 22.09.2009
              exit_minute = exit_minute + 1           # <<< 22.09.2009
        if exit_minute >= 60:                                   # <<< 22.09.2009
              exit_minute = exit_minute - 60          # <<< 22.09.2009
              exit_hour = exit_hour + 1                         # <<< 22.09.2009
        if exit_hour >= 24:                                     # <<< 22.09.2009
              exit_hour = exit_hour - 24                        # <<< 22.09.2009
              exit_day = exit_day + 1                           # <<< 22.09.2009
#
     if mainexit_status == 1:
               con = dbapi.connect('Exitstat_aol2.db', timeout=30)    # 12.12.2209
#              con = sqlite.connect('ExitstatNAR.db')          # 12.12.2009
             con.row_factory = dbapi.Row                        # 12.12.2009
             con.text_factory = str
               c = con.cursor()
#                              create Exitstat.aol2.db if db does not exit
               c.execute('CREATE TABLE IF NOT EXISTS Exitstatistics(Resourse TEXT,
Search_key TEXT, Date_inserted TEXT, Hostname_current TEXT, Hostip_address TEXT,
Lastpageno INTEGER, Entery_year INTEGER, Entery_month INTEGER, Entery_day INTEGER,
Exit_day INTEGER, Day_differ INTEGER, Entery_hour INTEGER, Exit_hour INTEGER, Hour_differ
INTEGER, Entery_minute INTEGER, Exit_minute INTEGER, Minute_differ INTEGER, Entery_second
INTEGER, Exit_second INTEGER, Second_differ INTEGER, PRIMARY KEY (Resourse, Search_key,
Date_inserted, Hostname_current))')
               print resourse, search_key, date_inserted, hostname_current,
hostip_address, lastpageno, entery_year, entery_month, entery_day, exit_day, day_differ,
entery_hour, exit_hour, hour_differ, entery_minute, exit_minute, minute_differ,
entery_second, exit_second, second_differ
#                              1                   2
#       1  2  3  4  5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
#              c.execute('INSERT INTO Exitstatistics
VALUES(%s,%s,%s,%s,%s,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',(resourse, search_key,
               c.execute('INSERT INTO Exitstatistics
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',(resourse, search_key,
date_inserted, hostname_current, hostip_address, lastpageno, entery_year, entery_month,
entery_day, exit_day, day_differ, entery_hour, exit_hour, hour_differ, entery_minute,
exit_minute, minute_differ, entery_second, exit_second, second_differ))

#  Save (commit) the changes
               con.commit()

# close the cursor we done with it
               c.close()
#
def find_duration():          #  called by "web_search"
#############################################################################
#                                                                           #
#  finds partial time spans for the addresses within the same Url.          #
#                                                                           #
#############################################################################
```

```
#
        (year, month, day, hour, minute, second, weekday, dayofyear, dst) = localtime()
        check_exit_day = day
        check_exit_hour = hour
        check_exit_minute = minute
        check_exit_second = second
        duration = 0
#
        epochb = localtime()
#
        if entery_second > check_exit_second:
                check_exit_second = check_exit_second + 60
                if check_exit_minute > 0:
                        check_exit_minute = check_exit_minute -1
             else:
                        check_exit_minute = 59
                        check_exit_hour = check_exit_hour -1
        if entery_minute > check_exit_minute:
                check_exit_minute = check_exit_minute + 60
             check_exit_hour = check_exit_hour -1
        if entery_hour > check_exit_hour:
                check_exit_hour = check_exit_hour + 24
                check_exit_day = check_exit_day -1
        print "\n  minute difference:", check_exit_minute-entery_minute,\
                "    second dif.:", check_exit_second-entery_second
        duration = (check_exit_minute*60 + check_exit_second) - (entery_minute * 60 +
entery_second)
         print "\n duration in find_duration", duration
         return duration
#
def formatdecimal(amount):            #  called by "web_search"
################################################################################
#                                                                              #
#  used for printing integer values in decimal format                          #
#                                                                              #
################################################################################
#
     str1 = str(amount)
     new = ' '
       n = 0
       while len(str1) > 3:
           m = n-3
              new = '.' + str1[-3:] + new
           n = n - 4
           str1 = str1[:-3]
      else:
           m = n-1
           new = str1[-3:] + new
      return "%s" % new
```

```
#
def log_exceptions(responses, url, msg_status, msg_reason, error_count):
#  called when exception occured
#############################################################################
#                                                                           #
#  is called when exception occures                                         #
#                                                                           #
#############################################################################
#
        error_count = error_count +1
        if msg_status != 0:
            print "\n msg_status:", msg_status
            for (key, entry) in responses.items():
#               print "\n key:", key, " entry:", entry
                if key == msg_status:                       # <<< 18.10.2009
                    msg_reason = entry[1]
                    print "\n msg_reason:", msg_reason
                    if msg_reason == " ":                   # <<< 18.10.2009
                        msg_reason = entry[1]               # <<< 18.10.2009
#
         con = dbapi.connect('log_error_aol2.db', timeout=30)  # 12.12.2009
         con.row_factory = dbapi.Row                      # 12.12.2009
         con.text_factory = str
            c = con.cursor()
#                           create log_error_aol2.db if db does not exit
            c.execute('CREATE TABLE IF NOT EXISTS Log_error(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Resourse TEXT, Search_key TEXT, Error_count
INTEGER, Msg_status INTEGER, Msg_reason TEXT, PRIMARY KEY (Root, Date_inserted,
Hostname_current, Error_count))')
            print groot, date_inserted, hostname_current, hostip_address, resourse,
search_key, error_count, msg_status, msg_reason
#
#                                           1 2 3 4 5 6 7 8 9
            c.execute('INSERT INTO Log_error VALUES(?,?,?,?,?,?,?,?,?)',(url,
date_inserted, hostname_current, hostip_address, resourse, search_key, error_count,
msg_status, msg_reason))

#  Save (commit) the changes
            con.commit()
# close the cursor we done with it
            c.close()
        else:
                msg_status = 0
                msg_reason = " "

        return error_count, msg_status, msg_reason
#
class FindLinks(SGMLParser):          #  called by "findUrl_links"
#############################################################################
#                                                                           #
#  parses the HTML and extract the link info which specified as HREF        #
#  attributes of the <A> tags,                                              #
#                                                                           #
#############################################################################
#
    def __init__(self):               # method is a constructor and is automatically
called when an intance of the FindLinks is created.
        self.links = []               # initializes the links intance variable to an
empty list and tehn calls the base class constructor
     SGMLParser.__init__(self)
    def do_a(self, attributes):       # invoked by the SGML parser when it encounters an
<A> tag,
        global count_href      # SGML encounters a <F00> tag corresponding to <A> tag
        for (name, value) in attributes:

     try:                         # <<< 20.09.2009
           if name == "href":
#               print "\n ********** href ********** "
#                print " self:", self, "\n name:", name, "\n  value:", value, "\n
attributes:", attributes
                c_value = value
                i = string.find(c_value, '#')   # find str length up to # char.
                if i >= 0:
                    c_value = c_value[:i]      # Remove # fragment
                i = string.find(c_value, '?')   # find str length up to ? char.
                if i >= 0:
```

```
                    c_value = c_value[:i]             # Remove ? fragment
                i = string.find(c_value, '!')    # find str length up to ! char.
                if i >= 0:
                    c_value = c_value[:i]             # Remove ! fragment
                i = string.find(c_value, 'busy.htm')  # find str length up to busy.htm
str.
                if i >= 0:
                    c_value = c_value[:i]             # Remove busy.htm fragment
                i = string.find(c_value, '<!')        # find str length up to <! chars.
                if i >= 0:
                    c_value = c_value[:i]             # Remove <!? fragment

                words = string.split(c_value)  # Split in whitespace delimited words
return
                string.join(words, "")          # Join words without whitespace
                value = c_value

if self.links.count(value) <= 0 :    # new if new address/value does not
                                     # exist in the list
                    if value != "":                  # if value is not null
                         if value[:5] == "http:":            # if it is a "http:" address

                                      self.links.append(value)  # new add new addr.to the
                                                                          # list
                              count_href = count_href + 1
#                             print " # of href:", count_href
#                          print "\n value>", value,"\n self.link 2nd>",self.links
      except sgmllib.SGMLParseError, ex:                                  # <<< 20.09.2009
              print "Pythonlib's error message: " + str(ex) # <<< 20.09.2009
              line, offset = parser.getpos()               # <<< 20.09.2009
              lines = parser.rawdata.split("\n")                  # <<< 20.09.2009
              print "My extra information: error at line %d offset %d" % parser.getpos()
                                                                  # <<< 20.09.2009
                print lines[line]                                 # <<< 20.09.2009
              print "%*s" % (offset, "^")                         # <<< 20.09.2009
              parser = None                                # <<< 20.09.2009
#
    def do_img(self, attributes):    # invoked by the SGML parser when it encounters an
                                                                      # <img> tag
        global count_img, imagelst
        for (name, value) in attributes:
            if name == "src":
               if imagelst.count(value) <= 0 :     # new if new address/value does not
                                               # exist in the list
                   if value != "":                  # if value is not null
                      imagelst.append(value)     # new add new addr. to the list
#                        print "\n *********** img ************ "
#                        print "\n self:", self, "\n name:", name, "\n  value:", value,
"\n attributes:", attributes
                         count_img += 1
#                        print "\n # of img:", count_img
#
    def do_form(self, attributes):    # invoked by the SGML parser when it encounters an
     # <form> tag
        global count_form, formlst
        for (name, value) in attributes:
           if name == "id" and value == "aspnetForm":
#             print "\n *********** form ************ "
#                print "\n self:", self, "\n name:", name, "\n  value:", value, "\n
attributes:", attributes
              count_form += 1
#               print "\n # of form:", count_form
#
    def getlinks(self):
#       print "\n self only >", self
        return self.links
#
def findUrl_links(url, responses, error_count):
# called by "web_search"
################################################################################
#                                                                              #
#  returns content and MIME type of a Url.                                     #
#                                                                              #
################################################################################
#
        global urlsize
```

```
        fulllinks = []                          ###06.09.2009#######
        ms_reason = " "                         # <<< 20.09.2009
       ms_status = 0                            # <<< 20.09.2009
#
#       timeout in seconds
        timeout = 10                            # <<< 20.09.2009
        socket.setdefaulttimeout(timeout)       # <<< 20.09.2009
#
#       this call to urllib2.urlopen now uses the default timeout   # <<< 20.09.2009
#       we have set in the socket module                            # <<< 20.09.2009

        try:                                                        #  03.09.2009

            f = urllib2.urlopen(url)                                # <<< 20.09.2009

         data = f.read()
#
         find_trchar_words(url, data)       # call for trchar & words search <<<

#        print "\n tr_char>", tr_char, "\n trchar>", trchar # <<< 14.09.2009
            for x in trchar:                                        # <<< 14.09.2009

                trchar[x] = trchar[x] + tr_char[x]                  # <<< 14.09.2009
#           print "\n trchar>", trchar                              # <<< 14.09.2009
#           print "\n tr_char_u>", tr_char_u, "\n trchar_u>", trchar_u      # <<<
14.09.2009
            for x in trchar_u:                                      # <<< 14.09.2009
                trchar_u[x] = trchar_u[x] + tr_char_u[x]            # <<< 14.09.2009
#           print "\n trchar_u>", trchar_u                          # <<< 14.09.2009
#           print "\n tr_char_utf8>", tr_char_utf8, "\n trchar_utf8>", trchar_utf8 # <<<
14.09.2009
            for x in trchar_utf8:                                   # <<< 14.09.2009
                trchar_utf8[x] = trchar_utf8[x] + tr_char_utf8[x]   # <<< 14.09.2009
#           print "\n trchar_utf8>", trchar_utf8                    # <<< 14.09.2009
#           print "\n tr_word>", tr_word, "\n trword>", trword      # <<< 14.09.2009
            for x in trword:                                        # <<< 14.09.2009
                trword[x] = trword[x] + tr_word[x]                  # <<< 14.09.2009
#           print "\n trword>", trword                              # <<< 14.09.2009
#           print "\n tr_word_u>", tr_word_u, "\n trword_u>", trword_u      # <<<
14.09.2009
            for x in trword_u:                                      # <<< 14.09.2009
                trword_u[x] = trword_u[x] + tr_word_u[x]            # <<< 14.09.2009
#           print "\n trword_u>", trword_u                          # <<< 14.09.2009
#           print "\n tr_word_utf8>", tr_word_utf8, "\n trword_utf8>", trword_utf8 # <<<
     # 14.09.2009
            for x in trword_utf8:                                   # <<< 14.09.2009
                trword_utf8[x] = trword_utf8[x] + tr_word_utf8[x]   # <<< 14.09.2009
#           print "\n trword_utf8>", trword_utf8                    # <<< 14.09.2009

#               print "\ start of data[1500] >", data[:1500]
###             print "\ start of data >", data
            headers = f.info()
#               print " *** f.info()>", f.info()
#               print "Read %d bytes from %s." % (len(data),f.geturl())
            urlsize = urlsize + len(data)
#               print "Total %d bytes read" % urlsize
#              print " headers.has_key >", headers.has_key
#              print " url[-5:] >", url[-5:], "\n url[-4:] >", url[-4:], "\n url[:] >",
url[:]
            f.close()

            if headers.has_key("content-type"):

               mimetype = headers["content-type"]
#              print " mimetype>",mimetype
            else:      # No content-type header -- guess type based on extension
               if url[-5:] == ".html":
                   mimetype = "text/html"
               else:
               # Don't know -- worst-case assumption
                   mimetype = "application/octet-stream"
#       start of getlinks
            timeout = 10                                        # <<< 20.09.2009
            socket.setdefaulttimeout(timeout)        # <<< 20.09.2009
            try:                                                # <<< 20.09.2009
#              print " mimetype[:9] >", mimetype[:9]
```

```
                if mimetype[:9] == "text/html":
                        parser = FindLinks()     # call FindLinks() to activate SGML
parser
      # which extracts Links
#                       print "\n parser >", parser

                        parser.feed(data)             # parse fed data
                        parser.close()
                        links = parser.getlinks()    # ***********getlinks()**************
                else:
                        links = []                    # Non-HTML data has no links
                 for link in links:                                    # <<< 20.09.2009
                        full = urlparse.urljoin(url, link)           # <<< 20.09.2009
                        fulllinks.append(full)                       # <<< 20.09.2009
#
            except sgmllib.SGMLParseError, ex:                        # <<< 20.09.2009
                print "Pythonlib's error message: " + str(ex) # <<< 20.09.2009
                line, offset = parser.getpos()               # <<< 20.09.2009
                lines = parser.rawdata.split("\n")           # <<< 20.09.2009
                print "My extra information: error at line %d offset %d" % parser.getpos()
        # <<< 20.09.2009
                        print lines[line]                             # <<< 20.09.2009
                print "%*s" % (offset, "^")                  # <<< 20.09.2009
                parser = None                                # <<< 20.09.2009

         except (socket.error, IOError, timeout, Exception), e:      # <<< 24.09.2009
             if hasattr(e, 'code'):                                  # <<< 24.09.2009
                   ms_status = e.code                                # <<< 24.09.2009
                if hasattr(e, 'reason'):                             # <<< 24.09.2009
                  ms_reason = e.reason                       # <<< 24.09.2009
                    (error_count, msg_status, msg_reason) = log_exceptions(responses, url,
ms_status, ms_reason, error_count)
#
        return fulllinks, ms_status, ms_reason, error_count
#
```

```
def web_search(root, responses, error_count):               #       called by "main"
#########################################################################
#                                                                       #
#  follows internal links to find all reachable documents in the tree   #
#   and uses depth search methos for internal query                     #
#                                                                       #
#########################################################################
#
       global count_form, count_href, count_img, firstlevel, imagelst, urlsize, urlwidth
       global trchar, trchar_u, trchar_utf8, trword, trword_u, trword_utf8
     global no_of_internal_links , no_of_external_links
       global doneLst                                       # <<< 14.09.2009
       global duration                                      # <<< 19.09.2009
       trchar =
{'ý':0,'ı':0,'ğ':0,'ü':0,'ş':0,'ö':0,'ç':0,'Ğ':0,'Ü':0,'Ş':0,'İ':0,'Ö':0,'Ç':0,'&uuml;':0
,'&ouml;':0, 'ccedil;':0,'&Ccedil;':0}
       trchar_u =
{u'ý':0,u'ı':0,u'ğ':0,u'ü':0,u'ş':0,u'ö':0,u'ç':0,u'Ğ':0,u'Ü':0,u'Ş':0,u'İ':0,u'Ö':0,u'Ç'
:0,u'&uuml;':0, u'&ouml;':0,u'ccedil;':0,u'&Ccedil;':0}
       trchar_utf8 =
{u'ý':0,u'ı':0,u'ğ':0,u'ü':0,u'ş':0,u'ö':0,u'ç':0,u'Ğ':0,u'Ü':0,u'Ş':0,u'İ':0,u'Ö':0,u'Ç'
:0,u'&uuml;':0, u'&ouml;':0,u'ccedil;':0,u'&Ccedil;':0}
#
       trword = {'Anadolu':0,'anasayfa':0,'ana
sayfa':0,'arama':0,'arkadaş':0,'dış':0,'eğitim':0,'firma':0,'hizmet':0,
       'geri':0,'İç':0,'İleri':0,'iletişim':0,'Şti':0,'ticaret':0,'Türk':0,'turizm':0,'T
ürkiye':0,'Üye':0,'yardım':0}
       trword_u = {u'Anadolu':0,u'anasayfa':0,u'ana
sayfa':0,u'arama':0,u'arkadaş':0,u'dış':0,u'eğitim':0,u'firma':0,
       u'hizmet':0,u'geri':0,u'İç':0,u'İleri':0,u'iletişim':0,u'Şti':0,u'ticaret':0,u'Tü
rk':0,u'turizm':0,u'Türkiye':0,
       u'Üye':0,u'yardım':0}
       trword_utf8 = {u'Anadolu':0,u'anasayfa':0,u'ana
sayfa':0,u'arama':0,u'arkadaş':0,u'dış':0,u'eğitim':0,u'firma':0,
       u'hizmet':0,u'geri':0,u'İç':0,u'İleri':0,u'iletişim':0,u'Şti':0,u'ticaret':0,u'Tü
rk':0,u'turizm':0,u'Türkiye':0,
       u'Üye':0,u'yardım':0}
#
       todo = [root]                            # list of Url.s which will be searched
       print "\n root >", root, " todo>", todo
#      doneLst = []                             # list of Url.s search were completed
       linkrefs = {root: ["<ROOT>"]}
       linkcount_inkeys = {root:0}
       external_links = []                      # contains external links
       urlsize = 0
       urldebth = 0
       urlwidth = 0
       count_href = 0                           # counts # href links within the Url
       count_img = 0                            # counts # images within the Url
       count_form = 0
       newlinks = []
       firstlevel = []
     level_depth = 0
       imagelst = []
       formlst = []
       lasturl_atfirstlevel = []
       lasturl_atlevel = ''
       stop_it = 1
       check_level = 0
     duration = 0
       msg_status = 0
       msg_reason = " "
#
       print "\n ********************************************************"
       print "\n max_duration:", max_duration, "  max_depth:", max_depth
       print "\n ********************************************************"
#
      while todo and todo[0] not in doneLst and duration < max_duration and check_level <
max_depth: # while Url/root exists and it does not processed before
             print "\n duration in web_search(root)>", duration
             url = todo[0]
             doneLst.append(url)  # put control for Url/root which must exist only
once!
           print "\n url>>>>>>", url, " lasturl_atfirstlevel>",lasturl_atfirstlevel
#            print "\n todo>", todo
#            print "\n doneLst.append(url)>", doneLst
```

```
              del todo[0]
              print "\n Checking in FindLinks>", url
#
            try:
                  (newlinks, msg_status, msg_reason, error_count) = findUrl_links(url,
responses, error_count)
#                 print "\n *** newlinks>", newlinks
                  if urlwidth == 0:
                      if len(newlinks) > 0:
                             firstlevel = newlinks
                             urlwidth = len(newlinks)
                             x = urlwidth
                             while (x):
                                    x = x-1
                                    lasturl_atfirstlevel = newlinks[x]
                                    lasturl_atlevel = newlinks[x]
                                    if lasturl_atfirstlevel[:len(root)] == root:
                                           x = 0
            except (IOError,URLError), e:
                  if hasattr(e, 'reason'):
                      print 'We failed to reach a server.', url
                      print 'Reason: ', e.reason
                       msg_reason = e.reason
                  elif hasattr(e, 'code'):
                      print 'The server couldn\'t fulfill the request.'
                      print 'Error code: ', e.code
                      msg_status = e.code
                      (error_count,msg_status,msg_reason) = log_exceptions(responses,
url, ms_status, ms_reason, error_count)
                       for ref in linkrefs[url]:
                           print "\t *******>" + ref
                      continue
               for link in newlinks:
                  print "\n * link>", link
                  if not linkrefs.has_key(url): #if url/key does not exist at the
linkrefs
                      linkrefs[url] = [link]         # add url/key to the dictionary
                      linkcount_inkeys[url] = 1
#                     print " key & link are new in linkrefs, url>", url, "  link>", link
                   else:
                      linkrefs[url].append(link)    # if url/key exists at the linkrefs,
# add link as value to the
# url/key
                    linkcount_inkeys[url] = linkcount_inkeys[url] + 1

                  level_depth = linkcount_inkeys[url]
#
                   if link[:len(root)] == root:
                       if (link not in doneLst and link not in todo):
                          todo.append(link)
                    else:
                       if link not in external_links:
                          external_links.append(link)
               if url == lasturl_atlevel:
                      stop_it = 0
                     if len(todo) > 1:
                         lasturl_atlevel = todo[len(todo) -1]
                         x = len(todo)
                       while (x):
                             x = x-1
                             lasturl_atlevel = todo[x]
                             if lasturl_atlevel[:len(root)] == root:
                                    x = 0
                                    check_level = check_level + 1
                                    print "\n   *****************************"
                                    print "   *                           *"
                                    print "   * check_level>", check_level,*"
                                    print "   *                           *"
                                    print "   *****************************"
              print "\n        url>>>>>>", url, "\n lasturl_atlevel>", lasturl_atlevel
               duration = find_duration()            # 06.09.2009 exit from url if
duration
                                                     # of query exceeds 10 minutes
               print "\n query duration at the same Url:", duration
#
     print "\n total length of ", root, " is: ", formatdecimal(urlsize), " bytes"
```

```
        print "\n width of %s is %d and Url adresses at the first level: %s " % (root,
urlwidth, firstlevel)
     print "\n last internal address at the first level:", lasturl_atfirstlevel
        print "\n %s contains %d hrefs and %d images, %d forms." % (root, count_href,
count_img, count_form)
#       print "\n Addresses reached from %s are %s " % (root, doneLst)
        no_of_internal_links =  len(linkrefs)
#    print "\n %s contains %d internal addresses which are: %s " % (root,  len(linkrefs),
linkrefs.keys())
        no_of_external_links = len(external_links)
        print "\n # external links: %d " % no_of_external_links
#       print "\n list of external links:", external_links          # <<<  23.09.2009
#
        return msg_status, msg_reason, check_level, error_count
#
def pars_html(file):          #       called by "main"
#########################################################################
#                                                                       #
#  parse lines of html file which contains result of google search      #
#  for keywords which are created at main script                        #
#                                                                       #
#########################################################################
#
        try:                                    # <<< 1-try loop changed at Oct. 9th 2009
                htmlfile = open(file,'r')

                htmldata = htmlfile.read()    # read html file which written at the main
# scrp.
                htmlfile.close()

                parser = htmllib.HTMLParser(formatter.NullFormatter())# then parse
       # what was read
                parser.feed(htmldata)
                parser.close()
                x = 0
                matchlist = []
                blacklist =
[[],['http://209.85.129.132/'],['http://209.85.135.132/'],['http://209.85.229.132/'], \
['http://books.google.co.uk/'],['http://commons.wikimedia.org/'],['http://docs.google.com
/'],['http://groups.google.com/'], \
['http://groups.yahoo.com/'],['http://images.google.co.uk/'],['http://mail.google.com/'],
['http://maps.google.co.uk/'], \
['http://news.google.co.uk/'],['http://translate.google.co.uk/'],['http://video.google.co
.uk/'],['http://video.google.com/'], \
['http://video.yahoo.com/'],['http://wikimapia.org/'],['http://www.blogcatalog.com/'],['h
ttp://www.facebook.com/'], \
['http://www.google.co.uk/'],['http://www.google.com/'],['http://www.youtube.com/']]
# last update at 20.10.2009
#
                while x < 41:
#                       print "\n parser.anchor[",x,"] ",parser.anchorlist[x]
                        match = []
                        match = re.findall(r'http://.*?/',parser.anchorlist[x])
                        if match == [] :
                                match = re.findall(r'www..*?/',parser.anchorlist[x])
                         else:
                                print "\n match->", match
                                        if not
re.findall(r'http://.*?.live.com/',parser.anchorlist[x]):   # block msn adresses
                                      if not
re.findall(r'http://.*?.google.com/',parser.anchorlist[x]): # block browser adresses
                                     if not
re.findall(r'http://.*?.facebook.com/',parser.anchorlist[x]):       # block facebook
adresses
                                           if not
re.findall(r'http://.*?.altavista.com/',parser.anchorlist[x]):      # block browser
adresses
                                             if not
re.findall(r'http://.*?.yahoo.com/',parser.anchorlist[x]):   # block browser adresses
                                              if not
re.findall(r'http://.*?.blog.*?/',parser.anchorlist[x]):    # block blog adresses
                                                if not
re.findall(r'http://blog..*?/',parser.anchorlist[x]):       # block blog adresses
                                                   if
re.findall(r'http://.*?/',parser.anchorlist[x]) not in matchlist and
re.findall(r'http://.*?/',parser.anchorlist[x]) not in blacklist:
```

```
matchlist.append((re.findall(r'http://.*?/',parser.anchorlist[x])))   # 2.9.2009
#                           print "\n matchlist >", matchlist
                        x += 1
        except (IOError,IndexError,URLError), e:# <<<< 2- IndexError is added to
                                                # the except case at Oct. 9th 2009
                if hasattr(e, 'reason'):
                      print 'We failed to reach ', file
                      print 'Reason: ', e.reason
                       msg_reason = e.reason
                  elif hasattr(e, 'code'):
                      print 'The server couldn\'t fulfill the request.'
                      print 'Error code: ', e.code
                      msg_status = e.code
        return matchlist                    # change 31.08   changed 18.10.2009
#
def db_operations(groot, responses, msg_status, msg_reason, check_level):
#       called by "main"
#########################################################################
#                                                                       #
#  TypeError: not all arguments converted during string formatting      #
#  inserts all information gathered related with Url to the db          #
#                                                                       #
#########################################################################
#
        generic_tld =
["aero","asia","biz","cat","com","coop","edu","gov","info","int","jobs","mil","mobi","mus
eum","name", "net","org","pro","tel","travel"]
       cc_tld =
["ac","ad","ae","af","ag","ai","al","am","an","ao","aq","ar","as","at","au","aw","ax","az
","ba","bb","bd","be",
"bf","bg","bh","bi","bj","bm","bn","bo","br","bs","bt","bw","by","bz","ca","cc","cd","cf"
,"cg","ch","ci","ck","cl","cm","cn",
"co","cr","cu","cv","cx","cy","cz","de","dj","dk","dm","do","dz","ec","ee","eg","er","es"
,"et","eu","fi","fj","fk","fm","fo",
"fr","ga","gd","ge","gf","gg","gh","gi","gl","gm","gn","gp","gq","gr","gs","gt","gu","gw"
,"gy","hk","hm","hn","hr","ht","hu",
"id","ie","il","im","in","io","iq","ir","is","it","je","jm","jo","jp","ke","kg","kh","ki"
,"km","kn","kp","kr","kw","ky","kz",
"la","lb","lc","li","lk","lr","ls","lt","lu","lv","ly","ma","mc","md","me","mg","mh","mk"
,"ml","mm","mn","mo","mp","mq","mr",
"ms","mt","mu","mv","mw","mx","my","mz","na","nc","ne","nf","ng","ni","nl","no","np","nr"
,"nu","nz","om","pa","pe","pf","pg",
"ph","pk","pl","pn","pr","ps","pt","pw","py","qa","re","ro","rs","ru","rw","sa","sb","sc"
,"sd","se","sg","sh","si","sk","sl",
"sm","sn","sr","st","su","sv","sy","sz","tc","td","tf","tg","th","tj","tk","tl","tm","tn"
,"to","tr","tt","tv","tw","tz","ua",
"ug","uk","us","uy","uz","va","vc","ve","vg","vi","vn","vu","wf","ws","ye","za","zm","zw"
]
#
        print "\n groot : ", groot
#
        groot_ext = " "
        nation = " "
#
        rfirst = groot.rfind(".")
        rlast = groot.rfind("/")
#
        xx = 0
        for x in generic_tld:
              if groot[rfirst+1:rlast] == x :
                      groot_ext = x
                      xx = 1    # if generic top-level-domain found at the end of
                              # Url address
#
        lfirst = groot.find("//")
        llast = groot.find(".")

        if xx == 1 :              # if last part of Url is not generic top-level-
                                # domain list
              for x in cc_tld:
                      if groot[lfirst+2:llast] == x:
nation = x       # if the first part of the Url address
                  #   matches
                                      # the one in countrycode top-level-domain
                                        # list
```

```
# if last part of Url is not generic top
# level-domain list
                for x in cc_tld:
                      if groot[rfirst+1:rlast] == x :
                          nation = x

                for x in cc_tld:
                      if groot[lfirst+2:llast] == x:
                           nation = x           # if the first part of the Url address
                                              # matches the one in countrycode
                                                 # top-level-domain list
                for x in generic_tld:
                      if groot[rfirst-3:rfirst] == x :  # is second part before the last
                                                        # segment of Url in generic top-
                                                        # level-domain list
                           groot_ext = groot[rfirst-3:rfirst] # assume the part before the
                                                        # last segment of Url address
                                                        # is generic top-level-domain
          if msg_status != 0:
                   print "\n msg_status:", msg_status
                  for (key, entry) in responses.items():
                         print "\n key:", key, " entry:", entry
                         if key == msg_status:                          # <<< 20.09.2009
                                msg_reason = entry[1]
                                print "\n msg_reason:", msg_reason
                                if msg_reason == " ":                   # <<< 20.09.2009
                                       msg_reason = entry[1]            # <<< 20.09.2009
           else:
                  msg_status = 0
                   msg_reason = " "
   #
          print "\n rfirst = groot.rfind('.')>", groot.rfind("."), " groot.rfind('/')>",
   groot.rfind("/")
          print "\n groot[rfirst-3:rfirst]>", groot[rfirst-3:rfirst]
        print "\n groot.find('//')>", groot.find("//"), " groot.find('.')>", groot.find(".")
        print "\n groot[lfirst+2:llast]>", groot[lfirst+2:llast]
          print "\n countrycode tld>", nation, " generic tld>", groot_ext
   #
          print groot, date_inserted, hostname_current, hostip_address, groot_ext, nation,
   urlsize, urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
   count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
   trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
   trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
   trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
   trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11], trchar_values[11],
   trchar_keys[12], trchar_values[12], trchar_keys[13], trchar_values[13], trchar_keys[14],
   trchar_values[14], trchar_keys[15], trchar_values[15], trchar_keys[16],
   trchar_values[16],  trword_keys[0], trword_values[0],  trword_keys[1], trword_values[1],
   trword_keys[2], trword_values[2], trword_keys[3], trword_values[3], trword_keys[4],
   trword_values[4], trword_keys[5], trword_values[5], trword_keys[6], trword_values[6],
   trword_keys[7], trword_values[7], trword_keys[8], trword_values[8], trword_keys[9],
   trword_values[9], trword_keys[10], trword_values[10], trword_keys[11], trword_values[11],
   trword_keys[12], trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
   trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
   trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
   trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason, duration,
   check_level, resourse, search_key, texteditor, lastpageno
   ##
   #       c.execute('INSERT INTO WebSpecs
   VALUES(%s,%s,%s,%s,%s,%s,?,?,?,?,?,?,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,
   %s, \
   #?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%
   s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?,%s,?, \
   #        ?,%s,?,?,%s,%s,%s,?)',
   #                          1                   2                   3                   4
   #       1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
          c.execute('INSERT INTO WebSpecs
   VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
          ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
   ,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
   #       1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   2 3 4 5 6 7 8 9 0 1 2 3 4 5
   #       4                   5                   6                   7                   8
                            9
   (groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
   urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img, count_form,
```

```
trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1], trchar_keys[2],
trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4], trchar_values[4],
trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6], trchar_keys[7],
trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9], trchar_values[9],
trchar_keys[10], trchar_values[10], trchar_keys[11], trchar_values[11], trchar_keys[12],
trchar_values[12], trchar_keys[13], trchar_values[13], trchar_keys[14],
trchar_values[14], trchar_keys[15], trchar_values[15], trchar_keys[16],
trchar_values[16],  trword_keys[0], trword_values[0],  trword_keys[1], trword_values[1],
trword_keys[2], trword_values[2], trword_keys[3], trword_values[3], trword_keys[4],
trword_values[4], trword_keys[5], trword_values[5], trword_keys[6], trword_values[6],
trword_keys[7], trword_values[7], trword_keys[8], trword_values[8], trword_keys[9],
trword_values[9], trword_keys[10], trword_values[10], trword_keys[11], trword_values[11],
trword_keys[12], trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason, duration,
check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
       con.commit()

# close the cursor we done with it
       c.close()
#
################################################################################
#                                                                              #
#  main script                                                                 #
#                                                                              #
#  - gets keywords from keybord to do google search and writes information     #
#    collected to the html file,                                               #
#                                                                              #
#  - calls "parse" routine to query Url addresses in a list from html file,    #
#                                                                              #
#   - then calls "web_search" routine to query each Url and internal addresses#
#  in it last all information gathered is inseted to db at "db_operations"     #
#  rouine.                                                                     #
#                                                                              #
################################################################################
#
# Table mapping response codes to messages; entries have the
# form {code: (shortmessage, longmessage)}.
responses = {
    100: ('Continue', 'Request received, please continue'),
    101: ('Switching Protocols',
          'Switching to new protocol; obey Upgrade header'),

    200: ('OK', 'Request fulfilled, document follows'),
    201: ('Created', 'Document created, URL follows'),
    202: ('Accepted',
          'Request accepted, processing continues off-line'),
    203: ('Non-Authoritative Information', 'Request fulfilled from cache'),
    204: ('No Content', 'Request fulfilled, nothing follows'),
    205: ('Reset Content', 'Clear input form for further input.'),
    206: ('Partial Content', 'Partial content follows.'),

    300: ('Multiple Choices',
          'Object has several resources -- see URI list'),
    301: ('Moved Permanently', 'Object moved permanently -- see URI list'),
    302: ('Found', 'Object moved temporarily -- see URI list'),
    303: ('See Other', 'Object moved -- see Method and URL list'),
    304: ('Not Modified',
          'Document has not changed since given time'),
    305: ('Use Proxy',
          'You must use proxy specified in Location to access this '
          'resource.'),
    307: ('Temporary Redirect',
          'Object moved temporarily -- see URI list'),

    400: ('Bad Request',
          'Bad request syntax or unsupported method'),
    401: ('Unauthorized',
          'No permission -- see authorization schemes'),
    402: ('Payment Required',
          'No payment -- see charging schemes'),
    403: ('Forbidden',
          'Request forbidden -- authorization will not help'),
    404: ('Not Found', 'Nothing matches the given URI'),
```

```
    405: ('Method Not Allowed',
          'Specified method is invalid for this server.'),
    406: ('Not Acceptable', 'URI not available in preferred format.'),
    407: ('Proxy Authentication Required', 'You must authenticate with '
          'this proxy before proceeding.'),
    408: ('Request Timeout', 'Request timed out; try again later.'),
    409: ('Conflict', 'Request conflict.'),
    410: ('Gone',
          'URI no longer exists and has been permanently removed.'),
    411: ('Length Required', 'Client must specify Content-Length.'),
    412: ('Precondition Failed', 'Precondition in headers is false.'),
    413: ('Request Entity Too Large', 'Entity is too large.'),
    414: ('Request-URI Too Long', 'URI is too long.'),
    415: ('Unsupported Media Type', 'Entity body in unsupported format.'),
    416: ('Requested Range Not Satisfiable',
          'Cannot satisfy request range.'),
    417: ('Expectation Failed',
          'Expect condition could not be satisfied.'),

    500: ('Internal Server Error', 'Server got itself in trouble'),
    501: ('Not Implemented',
          'Server does not support this operation'),
    502: ('Bad Gateway', 'Invalid responses from another server/proxy.'),
    503: ('Service Unavailable',
          'The server cannot process the request due to a high load'),
    504: ('Gateway Timeout',
          'The gateway server did not receive a timely response'),
    505: ('HTTP Version Not Supported', 'Cannot fulfill request.'),
    }
#
# mainexit_status = 0
# doneLst = []     # <<<<<<<<<<<<<<<<<<<< 14.09.2009
trword_lst = [u'Anadolu',u'anasayfa',u'ana
sayfa',u'arama',u'arkadaş',u'dış',u'eğitim',u'firma',u'hizmet',u'geri',u'İç',u'İleri',
u'iletişim', u'Şti',u'ticaret',u'Türk',u'turizm',u'Türkiye',u'Üye',u'yardım']
#
# start db operations
#
con = dbapi.connect('web_aol2.db', timeout=30)       # 12.12.2009
con.row_factory = dbapi.Row                          # 12.12.2009
con.text_factory = str
c = con.cursor()
print "\n con >", con
print "\n con.cursor() >", c
#
c.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize INTEGER,
Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER, Count_href
INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0 INTEGER,
Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3 TEXT,
Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5 INTEGER,
Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8 TEXT,
Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT, Tr_charno10
INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12 INTEGER,
Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER, Tr_char15 TEXT,
Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0 TEXT, Tr_wordno0
INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2 INTEGER, Tr_word3
TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5 TEXT, Tr_wordno5
INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7 INTEGER, Tr_word8
TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER, Tr_word10 TEXT, Tr_wordno10
INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12 TEXT, Tr_wordno12 INTEGER,
Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT, Tr_wordno14 INTEGER, Tr_word15 TEXT,
Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16 INTEGER, Tr_word17 TEXT, Tr_wordno17
INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER, Tr_word19 TEXT, Tr_wordno19 INTEGER,
Msg_status INTEGER, Msg_reason TEXT, Query_duration INTEGER, Depth_level INTEGER,
Resourse TEXT, Search_key TEXT, Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root,
Date_inserted, Hostname_current))')
#
print "\n os.name >>", os.name
#
print "\n hostname of the current machine>", socket.gethostname()
hostname_current = socket.gethostname()
#
print "\n IP address of given hostname>", socket.gethostbyname(hostname_current)
hostip_address = socket.gethostbyname(hostname_current)
#
```

```
print "\n ============================================="
#print "\n  Time: ", time.strftime('%c',time.localtime())
print "\n  Time: ", time.asctime(time.localtime())
print "\n ============================================="
#
outFileName = 'outputx.html'
outFileTxt = 'outtxt.txt'
outFileTxtObj = open(outFileTxt,'w')
#
inFileTxt = 'keywords_forAol2.txt'                  # 18.11.2009
FileTxt = open(inFileTxt,'r')                # 18.11.2009
#
while 1:                                            # 18.11.2009
  (year, month, day, hour, minute, second, weekday, dayofyear, dst) = localtime()
  entery_year_keep = year
  entery_month_keep = month
  entery_day_keep = day
  entery_hour_keep = hour
  entery_minute_keep = minute
  entery_second_keep = second
  date_inserted =  str(year * 10000000000 + month * 100000000 + day * 1000000 + hour *
10000+ minute * 100 + second)
  mainexit_status = 0                        # 25.11.2009
  doneLst = []                               # 25.11.2009
  resource = []                                       # 25.11.2009
  line = FileTxt.readline()                           # 18.11.2009
  if line:                                            # 18.11.2009
     print line                                       # 18.11.2009
     splitline = string.split(line)          # 18.11.2009
     (resourse, search_key, maxpageno, maxdepth, maxduration) = splitline # 18.11.2009
     resource.append(resourse)
     max_pageno  = int(maxpageno)                     # 18.11.2009
     max_depth = int(maxdepth)                        # 18.11.2009
     max_duration = int(maxduration)         # 18.11.2009
     print "\n  resource (txt):", resource, "search_key (txt)", search_key, "max_pageno
(int)", max_pageno, "max_depth (int)", max_depth, "max_duration (int)", max_duration
                                       # 18.11.2009
#
     if resource[0] == "google":
        command = 'links -source "http://www.google.co.uk/search?q='
        x = 1
        y = 1
        texteditor = "links"
     if resource[0] == "yahoo":
        command = 'elinks -source "http://search.yahoo.com/search?p='
        x = 1
        texteditor = "elinks"
     if resource[0] == "altavista":
        command = 'elinks -source
"http://www.altavista.com/web/results?itag=ody&kgs=0&kls=0&q='
        x = 0
        texteditor = "elinks"
     if resource[0] == "aol":
        command = 'elinks -source "http://search.aol.com/aol/search?q='
        x = 1
#       texteditor = "lynx"                          12.12.2009
        texteditor = "elinks"
#
if resource[0] == "bing":
        command = 'lynx -source "http://www.bing.com/search?q='
        x = 1
        y = 1
        texteditor = "lynx"                          # 12.12.2009
#
     while (x < max_pageno):
        lastpageno = x
        if resource[0] == "google":
               cmnd = command+search_key+'&hl=en&newwindow=1&start='+ str(x)+'&sa=N"'
         if resource[0] == "yahoo":
               cmnd = command+search_key+'&ei=UTF-8&fr=my-
myy&xargs=0&pstart=1&b='+str(x)+'&xa"'
         if resource[0] == "altavista":
               cmnd = command+search_key+'&stq='+str(x)+'"'
         if resource[0] == "aol":
               cmnd = command+search_key+'&page='+str(x)+'&nt=SG"'
         if resource[0] == "bing":
```

```
             if x == 1:
                cmnd = command+search_key+'&go=&form=QBLH&filt=all"'
           else:
               if y == 2:
                  cmnd = command+search_key+'&filt=all&first='+str(x)+'&FORM=PERE"'
             else:
                  cmnd = command+search_key+'&filt=all&first='+str(x)+'&FORM=PERE'+str(y-
1)+'"'

       outFileObject = open(outFileName,'w')
       print "\n search screen no.:", x, "\n cmnd >>", cmnd
       try:                          #   "if file:" replace with "try:"     25.11.2009
             p = Popen(cmnd, stdout=subprocess.PIPE, shell=True,
executable="/bin/bash")
             commandOutput = p.communicate()[0]
             outFileObject.write(commandOutput)
             outFileObject.close()
             matchlist = pars_html(outFileName)        # change 31.08 changed
18.10.2009
             print "\n *3* "
             if resource[0] == "google":
                    x = y * 10
                    y += 1
           if resource[0] == "yahoo":
                    x = x + 10
           if resource[0] == "altavista":
                    x = x + 10
           if resource[0] == "aol":
                    x = x + 1
             if resource[0] == "bing":
                    x = x + 10
                    y += 1
#            for Url in matchlist:
#            insert call for Url search here
             print "\n matchlist in main >", matchlist
           for groot in matchlist :
                groot = str(groot)                          # <<< 18.09.2009
                  first = groot.rfind("http")                    # <<< 18.09.2009
                  last = groot.rfind("/")                        # <<< 18.09.2009
                   groot = groot[first:last+1]                   # <<< 18.09.2009
                if groot not in doneLst:                         # <<< 14.09.2009
#                    print "\n groot in loop>", groot
                    (entery_year, entery_month, entery_day, entery_hour,
entery_minute, entery_second) = timentry()  # <<< 22.09.2009
                    print "\n groot in loop>", groot
                    error_count = 0                              # count # error
occures for each Url
                   (msg_status, msg_reason, check_level, error_count) =
web_search(groot, responses, error_count)
                    count_found()
                    timexit(entery_year, entery_month, entery_day, entery_hour,
entery_minute, entery_second, lastpageno)# <<< 22.09.2009
                    lastentry = entery_year * 10000000000 + entery_month * 100000000 +
entery_day * 1000000 + entery_hour * 10000 + entery_minute * 100 + entery_second
                    date_inserted =  str(exit_year * 10000000000 + exit_month *
100000000 + exit_day * 1000000 + exit_hour * 10000+ exit_minute * 100 + exit_second)
                    print "\n insert_date:", date_inserted
#
                    db_operations(groot, responses, msg_status, msg_reason,
check_level)
                    outFileTxtObj.write(groot)
       except:                                  # "else:" replaced with "except:"
                print "999 Error Openning File"             # <<< 20.09.2009
#
     print "\n x = ", x
#
     mainexit_status = 1
     print "\n valu of date_inserted:", date_inserted
     timexit(entery_year_keep, entery_month_keep, entery_day_keep, entery_hour_keep,
entery_minute_keep, entery_second_keep, lastpageno)  # <<< 22.09.2009
#
  else: break          # 18.11.2009
outFileTxtObj.close()  # 21.11.2009
FileTxt.close()               # 18.11.2009
```

Queries and Lists .dbs Created by Web Crawling Script

```
#! /usr/bin/python
# -*- coding: utf8 -*-
import urllib
from sgmllib import SGMLParser
import string
import urlparse
import os
import exceptions
import sys
import HTMLParser
import urllib
import htmllib, formatter
from urllib import urlopen
import unicodedata
import re
import time
import sqlite3 as dbapi
from time import gmtime, strftime, localtime
import subprocess
from subprocess import Popen
#
#####################################################################
#                                                                   #
#  main script                                                 #
#                                                                        #
#  lists all URL records inserted by                                 #
#  search_for_links_and_trchars_words_at_URL_resource-name1.py"      #
#  to web.db, error_log.db and Exitstat.db.                    #
#####################################################################
#
# start db operations
con = dbapi.connect('web_bing1.db')    # create web.db if db does not exit
c = con.cursor()
print "\n ****** records in web.db ****** "
#
x = 1
c.execute('SELECT * FROM WebSpecs ORDER BY Root')
for row in c:
    print "\n  Record no.: ", x
    print row
    x = x +1
c.close()
#
try:
       con = dbapi.connect('log_error_bing1.db')    # create web.cb if db does not
exit
       c = con.cursor()
       print "\n ****** records in log_error.db ****** "
#
       x = 1
       c.execute('SELECT * FROM Log_error ORDER BY Root')
       for row in c:
           print "\n  Record no.: ", x
           print row
           x = x +1
except:
       print " exception error occured"
#
con = dbapi.connect('Exitstat_bing1.db')    # create web.cb if db does not exit
c = con.cursor()
#print "\n con >", con
#print "\n con.cursor() >", c
print "\n ****** records in exitstat.db ****** "
#
c.execute('SELECT * FROM Exitstatistics ORDER BY Resourse')
for row in c:
    print row
```

Appends Web Resources to web_all.db

```
#! /usr/bin/python
# -*- coding: utf8 -*-
import urllib
from sgmllib import SGMLParser
import string
import urlparse
import os
import exceptions
import sys
import HTMLParser
import unicodedata
import re
import time
import sqlite3 as dbapi
from time import gmtime, strftime, localtime
import subprocess
from subprocess import Popen
import urllib2
from urllib2 import Request, urlopen, URLError
import urllib, htmllib, formatter
#
##############################################################################
#                                                                            #
#  main script                                                          #
#                                                                            #
#  reads records in "web_"+ resource+ seq_no+ ".db" which is defined from     #
#  terminal and appends to "web_all.db"                                      #
#                                                                            #
##############################################################################
#
#
resource = [""]
#
resource_lst = ["google","yahoo","altavista","aol","bing"]
#
print "\n resource_lst:", resource_lst
while resource[0] not in resource_lst:
    del resource[0]
    print "\n Enter one of the resource name: google/yahoo/altavista/aol:"
    read = sys.stdin.readline()
    found = str(read).find("\n")
    resourse1 = read[0:found]
    resource.append(read[0:found])
    print "\n resource name entered:", resource
#
print "\n enter the sequence of db for %s" % resource
read = sys.stdin.readline()
found = str(read).find("\n")
seq_of_db = read[0:found]
print "\n records of %s web_" % resourse1+seq_of_db, " will be appended to web_all.db"
#
db_name = "web_"+resourse1+seq_of_db
print resourse1, seq_of_db, db_name
#
com = dbapi.connect('web_all.db')    # create web.cb if db does not exit

c = com.cursor()
print "\n com >", com
print "\n com.cursor() >", c
#
c.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
```

```
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
# start db operations
con = dbapi.connect(db_name)
d = con.cursor()
print "\n con >", con
print "\n con.cursor() >", d
#
x = 1
con.text_factory = str
d.execute('SELECT * FROM WebSpecs ORDER BY Root')

for row in d:
        print "\n  Record no.: ", x
        print "\n value of c>", c, "\n  value of row>", row
        x = x +1
        trchar_keys = []
        trchar_values = []
        trword_keys = []
        trword_values = []
        groot = row[0]
        date_inserted = row[1]
        hostname_current = row[2]
        hostip_address = row[3]
        groot_ext = row[4]
        nation = row[5]
        urlsize = row[6]
        urlwidth = row[7]
        no_of_internal_links = row[8]
        no_of_external_links = row[9]
        count_href = row[10]
        count_img = row[11]
        count_form = row[12]
        i = 0
        j = 13
        trchar_valtot = 0
        while i <= 16:
          print "\n row[i+j]>",row[i+j], " row[i+j+1]>",row[i+j+1]
          trchar_keys.append(row[i+j])
          trchar_values.append(row[i+j+1])
          trchar_valtot = trchar_valtot + row[i+j+1]
          print "\n row[i+j+1]=", row[i+j+1], "  trchar valtotal=", trchar_valtot
          i = i + 1
          j = j + 1
        i = 0
        j = 47
        trword_valtot = 0
        while i <= 19:
          print "\n row[i+j]>",row[i+j], " row[i+j+1]>",row[i+j+1]
          trword_keys.append(row[i+j])
          trword_values.append(row[i+j+1])
          trword_valtot = trword_valtot + row[i+j+1]
          print "\n row[i+j+1]=", row[i+j+1], "  trword valtotal=", trword_valtot
          i = i + 1
          j = j + 1
        msg_status = row[87]
        msg_reason = row[88]
        duration = row[89]
        check_level = row[90]
        resourse = row[91]
        search_key = row[92]
        texteditor = row[93]
        lastpageno = row[94]
#        1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
0
```

```
        c.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
#      1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
#      4                 5                   6                    7
8
#                 9
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes                                    newwwwwwwwwwwwwww
       com.commit()
        print groot, date_inserted, hostname_current, hostip_address, groot_ext,
nation, urlsize, urlwidth, no_of_internal_links, no_of_external_links, count_href,
count_img, count_form, trchar_keys[0], trchar_values[0], trchar_keys[1],
trchar_values[1], trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3],
trchar_keys[4], trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6],
trchar_values[6], trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8],
trchar_keys[9], trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno
# close the cursor we done with it
d.close()
c.close()
```

```
#! /usr/bin/python
# -*- coding: utf8 -*-
import urllib
from sgmllib import SGMLParser
import string
import urlparse
import os
import exceptions
import sys
import HTMLParser
import unicodedata
import re
import time
import sqlite3 as dbapi
from time import gmtime, strftime, localtime
import subprocess
from subprocess import Popen
import urllib2
from urllib2 import Request, urlopen, URLError
import urllib, htmllib, formatter
#
def
callfor_tr_insert(groot,date_inserted,hostname_current,c_tr,c2,c4,c6,c8,c10,c12,c14,xx,
y,y2,y4,y6,y8,y10,y12,y14):
#       called by "main"
################################################################
#                                                              #
#   record was decided to be inserted to web.tr and related     #
#                                                    domains     #
################################################################
#
                c_tr.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
        ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                com_tr.commit()
                xx = xx + 1
                print "\n web_tr.db'ye yazıldı"
#
                if groot_ext == "com":
                   c2.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
        ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
```

```
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                    com2.commit()
                    y = y + 1
                    y2 = y2 + 1
                    print "\n comtr.db'ye yazıldı"
                 else:
                      if groot_ext == "edu":
                       c4.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                       com4.commit()
                       y = y + 1
                       y4 = y4 + 1
                       print "\n edutr.db'ye yazıldı"
                    else:
                         if groot_ext == "gov":
                          c6.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
```

```
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                          com6.commit()
                          y = y + 1
                            y6 = y6 + 1
                            print "\n govtr.db'ye yazıldı"
                       else:
                          if groot_ext == "net":
                             c8.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                             com8.commit()
                             y = y + 1
                               y8 = y8 + 1
                               print "\n nettr.db'ye yazıldı"
                          else:
                               if groot_ext == "org":
                                c10.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                                com10.commit()
                                y = y + 1
                                  y10 = y10 + 1
                                  print "\n orgtr.db'ye yazıldı"
                             else:
                                  if groot_ext == "info":
                                   c12.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
```

```
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                                   com12.commit()
                                   y = y + 1
                                     y12 = y12 + 1
                                     print "\n infotr.db'ye yazıldı"
                                else:
                                   c14.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                                com14.commit()
                                y = y + 1
                                  y14 = y14 + 1
                                  print "\n othertr.db'ye yazıldı"
                 return xx,y,y2,y4,y6,y8,y10,y12,y14
#
def callfor_nontr_insert(groot, date_inserted,
hostname_current,c_nontr,c1,c3,c5,c7,c9,c11,c13,yy,y,y1,y3,y5,y7,y9,y11,y13):
#       called by "main"
#################################################################
#                                                               #
#   record was decided to be inserted to web.nontr and related  #
#                                                   domains     #
#                                                               #
#################################################################
#
                    c_nontr.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
```

```
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                    com_nontr.commit()
                    yy = yy + 1
                    print "\n web_nontr.db'ye yazıldı"
                   if groot_ext == "com":
                      c1.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                       com1.commit()
                       y = y + 1
                         y1 = y1 + 1
                       print "\n com.db'ye yazıldı"
                   else:
                      if groot_ext == "edu":
                         c3.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
```

```
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                          com3.commit()
                          y = y + 1
                          y3 = y3 + 1
                          print "\n edu.db'ye yazıldı"
                       else:
                            if groot_ext == "gov":
                               c5.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                             com5.commit()
                             y = y + 1
                             y5 = y5 + 1
                             print "\n gov.db'ye yazıldı"
                          else:
                             if groot_ext == "net":
                                  c7.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
       ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                                com7.commit()
                                y = y + 1
                                  y7 = y7 + 1
                                print "\n net.db'ye yazıldı"
                             else:
```

```
                                    if groot_ext == "org":
                                        c9.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\

?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                                  com9.commit()
                                  y = y + 1
                                    y9 = y9 + 1
                                    print "\n org.db'ye yazıldı"
                               else:
                                    if groot_ext == "info":
                                        c11.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
        ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                                     com11.commit()
                                        y = y + 1
                                         y11 = y11 + 1
                                         print "\n info.db'ye yazıldı"
                                   else:
                                        c13.execute('INSERT INTO WebSpecs
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
\
        ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',     \
(groot, date_inserted, hostname_current, hostip_address, groot_ext, nation, urlsize,
urlwidth, no_of_internal_links, no_of_external_links, count_href, count_img,
count_form, trchar_keys[0], trchar_values[0], trchar_keys[1], trchar_values[1],
trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3], trchar_keys[4],
trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6], trchar_values[6],
trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8], trchar_keys[9],
```

```
trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno))
#  Save (commit) the changes
                                    com13.commit()
                                    y = y + 1
                                      y13 = y13 + 1
                                      print "\n other.db'ye yazıldı"
                    return yy,y,y1,y3,y5,y7,y9,y11,y13
#
##############################################################################
#                                                                            #
# start main - decomposes all records in "web_"+resource+seq_no+".db" records #
#         to domain1_tr, domain1_nontr...domainn_tr, domainn_nontr,          #
#             web_tr, web_nontr                                              #
#                                                                            #
##############################################################################
#
inFileTxt = 'ip_addresses_turkey.txt'
FileTxt = open(inFileTxt,'r')
#
x = 0
s1 = []
s2 = []
while x>=0:
        line = FileTxt.readline()
        print line
        if line:
                x = x + 1
                print "\n x = ", x
                splitline = string.split(line)
                (start_ipaddr, end_ipaddr, countr_code, country) = splitline
                s1.append(start_ipaddr)
                s2.append(end_ipaddr)
                 print "\n  start_ipaddr:", start_ipaddr, "end_ipaddr", end_ipaddr
         else: break
print "\n s1: ", s1
print "\n s2: ", s2
#
FileTxt.close()
#
resource = [""]
resource_lst = ["google","yahoo","altavista","aol","bing"]
#
print "\n resource_lst:", resource_lst
#while resource[0] not in resource_lst:
del resource[0]
print "\n Enter one of the resource name: google/yahoo/altavista/aol:"
read = sys.stdin.readline()
found = str(read).find("\n")
resourse1 = read[0:found]
resource.append(read[0:found])
print "\n resource name entered:", resource
#
print "\n enter the sequence of db for %s" % resource
read = sys.stdin.readline()
found = str(read).find("\n")
seq_of_db = read[0:found]
print "\n records of %s " % resourse1+seq_of_db, " will be appended to appropriate
domain.db"
#
#
db_name = "web_"+resourse1+seq_of_db+".db"
print resourse1, seq_of_db, db_name
#
```

```
com1 = dbapi.connect('domain_com.db')    # create domain_com.db if db does not exit

c1 = com1.cursor()
print "\n com >", com1, " com1.cursor() >", c1
#
com2 = dbapi.connect('domain_comtr.db')    # create domain_comtr.db if db does not exit

c2 = com2.cursor()
print "\n com2 >", com2, " com2.cursor() >", c2
#
com3 = dbapi.connect('domain_edu.db')    # create domain_edu.db if db does not exit

c3 = com3.cursor()
print "\n com3 >", com3, " com3.cursor() >", c3
#
com4 = dbapi.connect('domain_edutr.db')    # create domain_edutr.db if db does not exit

c4 = com4.cursor()
print "\n com4 >", com4, " com4.cursor() >", c4
#
com5 = dbapi.connect('domain_gov.db')    # create domain_gov.db if db does not exit

c5 = com5.cursor()
print "\n com5 >", com5, " com5.cursor() >", c5
#
com6 = dbapi.connect('domain_govtr.db')    # create domain_govtr.db if db does not exit

c6 = com6.cursor()
print "\n com6 >", com6, " com6.cursor() >", c6
#
com7 = dbapi.connect('domain_net.db')    # create domain_net.db if db does not exit

c7 = com7.cursor()
print "\n com7 >", com7, " com7.cursor() >", c7
#
com8 = dbapi.connect('domain_nettr.db')    # create domain_nettr.db if db does not exit

c8 = com8.cursor()
print "\n com8 >", com8, " com8.cursor() >", c8
#
com9 = dbapi.connect('domain_org.db')    # create domain_org.db if db does not exit

c9 = com9.cursor()
print "\n com9 >", com9, " com9.cursor() >", c9
#
com10 = dbapi.connect('domain_orgtr.db')    # create domain_orgtr.db if db does not
exit
c10 = com10.cursor()
print "\n com10 >", com10, " com10.cursor() >", c10
#
com11 = dbapi.connect('domain_info.db')    # create domain_info.db if db does not exit

c11 = com11.cursor()
print "\n com11 >", com11, " com11.cursor() >", c11
#
com12 = dbapi.connect('domain_infotr.db')    # create domain_infotr.db if db does not
exit
c12 = com12.cursor()
print "\n com12 >", com12, " com12.cursor() >", c12
#
com13 = dbapi.connect('domain_other.db')    # create domain_other.db if db does not
exit
c13 = com13.cursor()
print "\n com13 >", com11, " com13.cursor() >", c13
#
com14 = dbapi.connect('domain_othertr.db')    # create domain_othertr.db if db does not
exit
c14 = com14.cursor()
print "\n com14 >", com14, " com14.cursor() >", c14
#
com_tr = dbapi.connect('web_tr.db')    # create domain_othertr.db if db does not exit

c_tr = com_tr.cursor()
print "\n com_tr >", com_tr, " comtr.cursor() >", c_tr
#
```

```
com_nontr = dbapi.connect('web_nontr.db')    # create domain_othertr.db if db does not
exit
c_nontr = com_nontr.cursor()
print "\n com_nontr >", com_nontr, " com14.cursor() >", c_nontr
#
c1.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c2.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c3.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
```

```
#
c4.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c5.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c6.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c7.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
```

```
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c8.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c9.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c10.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
```

```
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c11.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c12.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c13.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
```

```
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c14.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c_tr.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
c_nontr.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links INTEGER,
Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT, Tr_charno0
INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2 INTEGER, Tr_char3
TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER, Tr_char5 TEXT, Tr_charno5
INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7 TEXT, Tr_charno7 INTEGER, Tr_char8
TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT, Tr_charno9 INTEGER, Tr_char10 TEXT,
Tr_charno10 INTEGER, Tr_char11 TEXT, Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12
INTEGER, Tr_char13 TEXT, Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER,
Tr_char15 TEXT, Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0
TEXT, Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER, Tr_word5
TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7 TEXT, Tr_wordno7
INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT, Tr_wordno9 INTEGER,
```

```
Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT, Tr_wordno11 INTEGER, Tr_word12
TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT, Tr_wordno13 INTEGER, Tr_word14 TEXT,
Tr_wordno14 INTEGER, Tr_word15 TEXT, Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16
INTEGER, Tr_word17 TEXT, Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER,
Tr_word19 TEXT, Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT,
Query_duration INTEGER, Depth_level INTEGER, Resourse TEXT, Search_key TEXT,
Text_editor TEXT, Lastpageno INTEGER, PRIMARY KEY (Root, Date_inserted,
Hostname_current))')
#
# start db operations
con = dbapi.connect(db_name)
d = con.cursor()
print "\n con >", con
print "\n con.cursor() >", d
#
groot = " "
#
x = 0
y = 0
y1 = 0
y2 = 0
y3 = 0
y4 = 0
y5 = 0
y6 = 0
y7 = 0
y8 = 0
y9 = 0
y10 = 0
y11 = 0
y12 = 0
y13 = 0
y14 = 0
xx = 0     # no.of tr-webs inserted
yy = 0     # no.of nontr-webs inserted
skipped_records = 0
zerosize_records = 0
con.text_factory = str
d.execute('SELECT * FROM WebSpecs ORDER BY Root')

for row in d:
     x = x +1
     print "\n  # Record read: ", x
#    print "\n  value of row>", row
     print "\n urlsize>", row[6], "  error_code>", row[87]
     if row[6] == 0 and row[87] > 0:
               print "\n record deleted - value of row>", row
               skipped_records = skipped_records + 1
     else:
          if row[6] == 0 :
            zerosize_records = zerosize_records + 1
#
            trchar_keys = []
              trchar_values = []
              trword_keys = []
              trword_values = []
              groot = row[0]
            date_inserted = row[1]
            hostname_current = row[2]
            hostip_address = row[3]
            groot_ext = row[4]
            nation = row[5]
            urlsize = row[6]
            urlwidth = row[7]
            no_of_internal_links = row[8]
            no_of_external_links = row[9]
            count_href = row[10]
              count_img = row[11]
              count_form = row[12]
            i = 0
              j = 13
            trchar_valtot = 0
            while i <= 16:
#              print "\n row[i+j]>",row[i+j], " row[i+j+1]>",row[i+j+1]
               trchar_keys.append(row[i+j])
               trchar_values.append(row[i+j+1])
```

```
             trchar_valtot = trchar_valtot + row[i+j+1]
               i = i + 1
               j = j + 1
              print "\n trchar valtotal=", trchar_valtot
              i = 0
               j = 47
              trword_valtot = 0
              other_trword_count = 0          # count other keywords (than "geri" and
"firma") which is >0
               total_of_geri_firma = 0
              while i <= 19:
#                 print "\n row[i+j]>",row[i+j], " row[i+j+1]>",row[i+j+1]
                  trword_keys.append(row[i+j])
                  trword_values.append(row[i+j+1])
                trword_valtot = trword_valtot + row[i+j+1]
                if (i+j+1) != 52 or (i+j+1) != 58:
                   if row[i+j+1] > 0:
                      other_trword_count = other_trword_count + 1
               i = i + 1
               j = j + 1
                total_of_geri_firma = row[52] + row[58]
              print "\n trword valtotal=", trword_valtot, "  total_of_geri_firma=",
total_of_geri_firma
              msg_status = row[87]
              msg_reason = row[88]
              duration = row[89]
              check_level = row[90]
              resourse = row[91]
              search_key = row[92]
              texteditor = row[93]
              lastpageno = row[94]
#
                print groot, date_inserted, hostname_current, hostip_address,
groot_ext, nation, urlsize, urlwidth, no_of_internal_links, no_of_external_links,
count_href, count_img, count_form, trchar_keys[0], trchar_values[0], trchar_keys[1],
trchar_values[1], trchar_keys[2], trchar_values[2], trchar_keys[3], trchar_values[3],
trchar_keys[4], trchar_values[4], trchar_keys[5], trchar_values[5], trchar_keys[6],
trchar_values[6], trchar_keys[7], trchar_values[7], trchar_keys[8], trchar_values[8],
trchar_keys[9], trchar_values[9], trchar_keys[10], trchar_values[10], trchar_keys[11],
trchar_values[11], trchar_keys[12], trchar_values[12], trchar_keys[13],
trchar_values[13], trchar_keys[14], trchar_values[14], trchar_keys[15],
trchar_values[15], trchar_keys[16], trchar_values[16],  trword_keys[0],
trword_values[0],  trword_keys[1], trword_values[1], trword_keys[2], trword_values[2],
trword_keys[3], trword_values[3], trword_keys[4], trword_values[4], trword_keys[5],
trword_values[5], trword_keys[6], trword_values[6], trword_keys[7], trword_values[7],
trword_keys[8], trword_values[8], trword_keys[9], trword_values[9], trword_keys[10],
trword_values[10], trword_keys[11], trword_values[11], trword_keys[12],
trword_values[12], trword_keys[13], trword_values[13], trword_keys[14],
trword_values[14], trword_keys[15], trword_values[15], trword_keys[16],
trword_values[16], trword_keys[17], trword_values[17], trword_keys[18],
trword_values[18], trword_keys[19], trword_values[19], msg_status, msg_reason,
duration, check_level, resourse, search_key, texteditor, lastpageno
#
              if groot_ext == " " and nation == " ":
                 first_doubleslah = groot.find("//")
                last_slash = groot.rfind("/")
                print "\n groot[first_doubleslah+2:last_slash]:",
groot[first_doubleslah+2:last_slash]
                ip_addr = ''
                 piece = ''
                for piece in groot[first_doubleslah+2:last_slash]:
#                   print "\n piece: ", piece
                    if piece != '.':
                      ip_addr += piece

                 print "\n ip_addr: ", ip_addr
                 found = 0
                 x1 = 0
#                print "\n s1[1]: ", s1[1]
#                print "\n s2[0] ", s2[0]
                 while x1 < 305 and found == 0:
                    if ip_addr >= s1[x1] and ip_addr <= s2[x1]:
                       found = 1
                       print "\n ************************** FOUND
************************* "
                     print "\n s1[x1] in first: ", s1[x1]
```

```
                    print "\n s2[x1] in first: ", s2[x1]
                 else:
                    x1 = x1 + 1
              if found == 1:
                 print "\n address found in range, address is ", ip_addr
#                print "\n address ranges, starting addr: ", start_addr, " ending
addr: ", end_addr
                 print "\n address ranges, starting addr: ", s1[x1], " ending addr:
", s2[x1]
                 nation = "tr"
#
#            if nation == "tr" or trchar_valtot > 100 or trword_valtot > 100:     #
domain is as assumed "tr"
             if nation == "tr":      # domain is as assumed "tr"
# <<<<<<  >>>>>>
                (xx,y,y2,y4,y6,y8,y10,y12,y14)
=callfor_tr_insert(groot,date_inserted,hostname_current,c_tr,c2,c4,c6,c8,c10,c12,c14,
xx,y,y2,y4,y6,y8,y10,y12,y14)
#
              else:    # domain is assumed as non-tr
                 if nation != " ":    # if domain is other than "tr"
#  <<<<<<<   >>>>>>>
                    (yy,y,y1,y3,y5,y7,y9,y11,y13) =
callfor_nontr_insert(groot,date_inserted,hostname_current,c_nontr,c1,c3,c5,c7,c9,c11,c1
3,yy, y,y1,y3,y5,y7,y9,y11,y13)
                 else:
                    if trchar_valtot > 100 or trword_valtot > 100:  # domain is assumed
as "tr"
                       if trword_valtot > 0 :
                          if total_of_geri_firma > 0:
                            if trword_valtot == total_of_geri_firma:
#  <<<<<<    >>>>>>>>
                                  (yy,y,y1,y3,y5,y7,y9,y11,y13) =
callfor_nontr_insert(groot,date_inserted,hostname_current,c_nontr,c1,c3,c5,c7,c9,
c11,c13,yy,y,y1,y3,y5,y7,y9,y11,y13)
                               else:
#  <<<<<<<<    >>>>>>         if trchar_valtot == 0 :
#                                 (yy,y,y1,y3,y5,y7,y9,y11,y13) =
callfor_nontr_insert(groot,date_inserted,hostname_current,c_nontr,c1,c3,c5,c7,c9,
c11,c13,yy,y,y1,y3,y5,y7,y9,y11,y13)
#                                else:
                                  (xx,y,y2,y4,y6,y8,y10,y12,y14)
=callfor_tr_insert(groot,date_inserted,hostname_current,c_tr,c2,c4,c6,c8,
c10,c12,c14,xx,y,y2,y4,y6,y8,y10,y12,y14)
                               else:
#  <<<<<<    >>>>>>>         if other_trword_count > 1:      #  if no.of keywords other
than (firma + geri) > 1
                                 (xx,y,y2,y4,y6,y8,y10,y12,y14)
=callfor_tr_insert(groot,date_inserted,hostname_current,c_tr,c2,c4,c6,c8,c10,
c12,c14,xx,y,y2,y4,y6,y8,y10,y12,y14)
                            else:
#  <<<<<    >>>>>>
                              (yy,y,y1,y3,y5,y7,y9,y11,y13) =
callfor_nontr_insert(groot,date_inserted,hostname_current,c_nontr,c1,c3,c5,c7,c9,
c11,c13,yy,y,y1,y3,y5,y7,y9,y11,y13)
#
                    else:                       # "else:" replaced with "except:"
                       if trword_valtot > 0 :
                          if total_of_geri_firma > 0:
                            if trword_valtot == total_of_geri_firma:
#  <<<<<<    >>>>>>>>
                                  (yy,y,y1,y3,y5,y7,y9,y11,y13) =
callfor_nontr_insert(groot,date_inserted,hostname_current,c_nontr,c1,c3,c5,c7,c9,
c11,c13,yy,y,y1,y3,y5,y7,y9,y11,y13)
                               else:
 #  <<<<<<<<    >>>>>>
                                  (xx,y,y2,y4,y6,y8,y10,y12,y14)
=callfor_tr_insert(groot,date_inserted,hostname_current,c_tr,c2,c4,c6,c8,
c10,c12,c14,xx,y,y2,y4,y6,y8,y10,y12,y14)
                               else:
#  <<<<<<    >>>>>>>
                                 (xx,y,y2,y4,y6,y8,y10,y12,y14)
=callfor_tr_insert(groot,date_inserted,hostname_current,c_tr,c2,c4,c6,c8,c10,
c12,c14,xx,y,y2,y4,y6,y8,y10,y12,y14)
                            else:
#  <<<<<    >>>>>>
```

```
                          (yy,y,y1,y3,y5,y7,y9,y11,y13) =
callfor_nontr_insert(groot,date_inserted,hostname_current,c_nontr,c1,c3,c5,c7,c9,
c11,c13,yy,y,y1,y3,y5,y7,y9,y11,y13)
#
print "\n  No.of records read: ", x
print "\n  No.of records inserted: ", y
print "\n  No.of records not selected: ", skipped_records
print "\n  No.of records selected having zero size: ", zerosize_records
print "\n  No.of records inserted to com.tr: ", y2
print "\n  No.of records inserted to edu.tr: ", y4
print "\n  No.of records inserted to gov.tr: ", y6
print "\n  No.of records inserted to net.tr: ", y8
print "\n  No.of records inserted to org.tr: ", y10
print "\n  No.of records inserted to info.tr: ", y12
print "\n  No.of records inserted to other.tr: ", y14
print "\n  No.of records inserted to com: ", y1
print "\n  No.of records inserted to edu: ", y3
print "\n  No.of records inserted to gov: ", y5
print "\n  No.of records inserted to net: ", y7
print "\n  No.of records inserted to org: ", y9
print "\n  No.of records inserted to info: ", y11
print "\n  No.of records inserted to other: ", y13
#
print "\n  No.of records inserted to web_tr: ", xx
print "\n  No.of records inserted to web_nontr: ", yy
print "\n statistics for %s " % db_name, " was reported."
# close the cursor we done with it
d.close()
c1.close()
c2.close()
c3.close()
c4.close()
c5.close()
c6.close()
c7.close()
c8.close()
c9.close()
c10.close()
c11.close()
c12.close()
c13.close()
c14.close()
c_tr.close()
c_nontr.close()
```

```
#! /usr/bin/python
# -*- coding: utf8 -*-
import urllib
from sgmllib import SGMLParser
import string
import urlparse
import os
import exceptions
import sys
import HTMLParser
import unicodedata
import re
import time
import sqlite3 as dbapi
from time import gmtime, strftime, localtime
import subprocess
from subprocess import Popen
import urllib2
from urllib2 import Request, urlopen, URLError
import urllib, htmllib, formatter
#
def find_duration(row1,date_prevread):       #  called by main
#############################################################################
#                                                                           #
#  finds caching time difference (in seconds) within the same URLs Url.      #
#                                                                           #
#############################################################################
#
      year_prev   =  int(date_prevread)/10000000000
      month_prev  = (int(date_prevread) -  year_prev * 10000000000)/100000000
       day_prev    = (int(date_prevread) - (year_prev * 10000000000 + month_prev *
100000000)) /1000000
       hour_prev   = (int(date_prevread) - (year_prev * 10000000000 + month_prev *
100000000 + day_prev * 1000000)) /10000
       minute_prev = (int(date_prevread) - (year_prev * 10000000000 + month_prev *
100000000 + day_prev * 1000000 + hour_prev * 10000)) /100
       second_prev = (int(date_prevread) - (year_prev * 10000000000 + month_prev *
100000000 + day_prev * 1000000 + hour_prev * 10000 + minute_prev * 100))
      year   =  int(row1)/10000000000
      month  = (int(row1) -  year * 10000000000)/100000000
       day    = (int(row1) - (year * 10000000000 + month * 100000000)) /1000000
       hour   = (int(row1) - (year * 10000000000 + month * 100000000 + day *
1000000)) /10000
       minute = (int(row1) - (year * 10000000000 + month * 100000000 + day *
1000000 + hour * 10000)) /100
       second = (int(row1) - (year * 10000000000 + month * 100000000 + day *
1000000 + hour * 10000 + minute * 100))

       time_diff = 0
#

       if second_prev > second:
              second = second + 60
              if minute > 0:
                       minute = minute -1
               else:
                       minute = 59
                       hour = hour -1
       if minute_prev > minute:
              minute = minute + 60
               hour = hour -1
       if hour_prev > hour:
              hour = hour + 24
              day = day -1
       print "\n  minute difference:", minute-minute_prev,"    second dif.:",
second-second_prev
       time_diff = (minute * 60 + second) - (minute_prev * 60 + second_prev)
       print "\n cashing period found (in seconds)", time_diff
       return time_diff
```

```
#
#############################################################################
#                                                                           #
# start main - reads all records in "web/domain_resource+seq_no"+".db" and  #
# reports statistics related to the db.                                     #
#                                                                           #
#############################################################################
#
resource = [""]
del resource[0]
print "\n Enter file_name or domain_name :"
read = sys.stdin.readline()
found = str(read).find("\n")
resourse1 = read[0:found]
resource.append(read[0:found])
print "\n file_name or domain_name entered (withouth .db):", resource
#
db_name = resourse1+".db"
print resourse1, db_name
#
# start db operations
con = dbapi.connect(db_name)
c = con.cursor()
print "\n con >", con
print "\n con.cursor() >", c
#
c.execute('CREATE TABLE IF NOT EXISTS WebSpecs(Root TEXT, Date_inserted TEXT,
Hostname_current TEXT, Hostip_address TEXT, Root_ext TEXT, Nation TEXT, Urlsize
INTEGER, Urlwidth INTEGER, No_of_internal_links INTEGER, No_of_external_links
INTEGER, Count_href INTEGER, Count_img INTEGER, Count_form INTEGER, Tr_char0 TEXT,
Tr_charno0 INTEGER, Tr_char1 TEXT, Tr_charno1 INTEGER, Tr_char2 TEXT, Tr_charno2
INTEGER, Tr_char3 TEXT, Tr_charno3 INTEGER, Tr_char4 TEXT, Tr_charno4 INTEGER,
Tr_char5 TEXT, Tr_charno5 INTEGER, Tr_char6 TEXT, Tr_charno6 INTEGER, Tr_char7
TEXT, Tr_charno7 INTEGER, Tr_char8 TEXT, Tr_charno8 INTEGER, Tr_char9 TEXT,
Tr_charno9 INTEGER, Tr_char10 TEXT, Tr_charno10 INTEGER, Tr_char11 TEXT,
Tr_charno11 INTEGER, Tr_char12 TEXT, Tr_charno12 INTEGER, Tr_char13 TEXT,
Tr_charno13 INTEGER, Tr_char14 TEXT, Tr_charno14 INTEGER, Tr_char15 TEXT,
Tr_charno15 INTEGER, Tr_char16 TEXT, Tr_charno16 INTEGER, Tr_word0 TEXT,
Tr_wordno0 INTEGER, Tr_word1 TEXT, Tr_wordno1 INTEGER, Tr_word2 TEXT, Tr_wordno2
INTEGER, Tr_word3 TEXT, Tr_wordno3 INTEGER, Tr_word4 TEXT, Tr_wordno4 INTEGER,
Tr_word5 TEXT, Tr_wordno5 INTEGER, Tr_word6 TEXT, Tr_wordno6 INTEGER, Tr_word7
TEXT, Tr_wordno7 INTEGER, Tr_word8 TEXT, Tr_wordno8 INTEGER, Tr_word9 TEXT,
Tr_wordno9 INTEGER, Tr_word10 TEXT, Tr_wordno10 INTEGER, Tr_word11 TEXT,
Tr_wordno11 INTEGER, Tr_word12 TEXT, Tr_wordno12 INTEGER, Tr_word13 TEXT,
Tr_wordno13 INTEGER, Tr_word14 TEXT, Tr_wordno14 INTEGER, Tr_word15 TEXT,
Tr_wordno15 INTEGER, Tr_word16 TEXT, Tr_wordno16 INTEGER, Tr_word17 TEXT,
Tr_wordno17 INTEGER, Tr_word18 TEXT, Tr_wordno18 INTEGER, Tr_word19 TEXT,
Tr_wordno19 INTEGER, Msg_status INTEGER, Msg_reason TEXT, Query_duration INTEGER,
Depth_level INTEGER, Resourse TEXT, Search_key TEXT, Text_editor TEXT, Lastpageno
INTEGER, PRIMARY KEY (Root, Date_inserted, Hostname_current))')
#
x = 0
y = 0
groot = " "
#                yyyymmddhhmmss
date_prevread = "20090110091011"
cashing_period = 0
average_cashing_period = 0
urlsize_prevread = 0
urlwidth_prevread = 0
highest_check_level = 0
#
urlsize = 0
urlwidth = 0
no_of_internal_links = 0
no_of_external_links = 0
count_href = 0
count_img = 0
count_form = 0
duration = 0
check_level = 0
#
con.text_factory = str
c.execute('SELECT * FROM WebSpecs ORDER BY Root')
#
for row in c:
```

```
     x = x +1
     print "\n  # Record read: ", x
     print "\n  Row content>", row
#
     if row[0] == groot:
          root_eq = 1                # if the new root equals to previously read
          if row[1] > date_prevread and row[6] != urlsize_prevread and row[7] !=
urlwidth_prevread:     # compare date of newer is the latest and
#                                      size and urlwidth must be changed
             time_diff = find_duration(row[1],date_prevread)
             cashing_period = cashing_period + time_diff
             y = y + 1
     groot = row[0]
     date_prevread = row[1]
     urlsize_prevread = row[6]
     urlwidth_prevread = row [7]
     urlsize = urlsize + row[6]
     urlwidth = urlwidth + row[7]
     no_of_internal_links = no_of_internal_links + row[8]
     no_of_external_links = no_of_external_links + row[9]
     count_href = count_href + row[10]
     count_img = count_img + row[11]
     count_form = count_form + row[12]
     duration = duration + row[89]
     check_level = check_level + row[90]
     if row[90] > highest_check_level:
        highest_check_level = row[90]

print "\n statistics for %s domain" % db_name
print " ==================================="
#
print "\n  count of all Urls having size > 0 : ", x
print "\n  count of Urls having size = 0 : ", xx
minutes = average_cashing_period / 60
seconds = average_cashing_period - minutes * 60
average_cashing_period = cashing_period / y
print "\n  Average caching period (in seconds): ",
average_cashing_period, " ( %d minutes " % minutes, " %d seconds)" %
seconds
print "\n  total no. of repeated records having size > 0 : ", y
print "\n  count of repeated records which differs than the previous
record: ", yy
print "\n  observation of same URL if it exists at least twice (unique
count for each repeating URL): ", unique_count
print "\n  Average URL size: ", urlsize / x
print "\n  Average URL width: ", urlwidth / x
print "\n  Average internal links: ", no_of_internal_links / x
print "\n  Average external links: ", no_of_external_links / x
print "\n  Average HREF link info: ", count_href / x
print "\n  Average IMG link: ", count_img / x
print "\n  Average FORM link: ", count_form / x
print "\n  Average search duration (in seconds): ", duration / x
# print "\n  Max. depth level reached: ", highest_check_level
# print "\n  Total # depths were searched: ", check_level
# print "\n  Average depth level: ", check_level / x#
#
# close the cursor we done with it
c.close()
```

Combine Exitstat.dbs

```
#! /usr/bin/python
# -*- coding: utf8 -*-
import urllib
from sgmllib import SGMLParser
import string
import urlparse
import os
import exceptions
import sys
import HTMLParser
import unicodedata
import re
import time
import sqlite3 as dbapi
from time import gmtime, strftime, localtime
import subprocess
from subprocess import Popen
import urllib2
from urllib2 import Request, urlopen, URLError
import urllib, htmllib, formatter
#
#
##############################################################################
#                                                                            #
#  main script                                                               #
#
#
#  reads records in "Exitstat_"+ resource+ seq_no+ ".db" which is defined from #
#
#
# terminal and appends to "Exitstat_all.db"                                  #
#                                                                            #
##############################################################################
#
resource = [""]
resource_lst = ["google","yahoo","altavista","aol","bing"]
#
print "\n resource_lst:", resource_lst
while resource[0] not in resource_lst:
    del resource[0]
    print "\n Enter one of the resource name: google/yahoo/altavista/aol:"
    read = sys.stdin.readline()
    found = str(read).find("\n")
    resourse1 = read[0:found]
    resource.append(read[0:found])
    print "\n resource name entered:", resource
#
print "\n enter the sequence of db for %s" % resource
read = sys.stdin.readline()
found = str(read).find("\n")
seq_of_db = read[0:found]
print "\n records of %s " % resourse1+seq_of_db, " will be appended to appropriate
domain.db"
#
db_name = "Exitstat_"+resourse1+seq_of_db+".db"
print resourse1, seq_of_db, db_name
#
com = dbapi.connect('Exitstat_all.db', timeout=30)          # create domain_com.db
if db does not exit
com.row_factory = dbapi.Row                          # 12.12.2009
com.text_factory = str
c = com.cursor()
print "\n com >", com, " com.cursor() >", c
#                              create Exitstat.all.db if db does not exit
c.execute('CREATE TABLE IF NOT EXISTS Exitstatistics(Resourse TEXT, Search_key
TEXT, Date_inserted TEXT, Hostname_current TEXT, Hostip_address TEXT, Lastpageno
INTEGER, Entery_year INTEGER, Entery_month INTEGER, Entery_day INTEGER, Exit_day
INTEGER, Day_differ INTEGER, Entery_hour INTEGER, Exit_hour INTEGER, Hour_differ
INTEGER, Entery_minute INTEGER, Exit_minute INTEGER, Minute_differ INTEGER,
Entery_second INTEGER, Exit_second INTEGER, Second_differ INTEGER, PRIMARY KEY
(Resourse, Search_key, Date_inserted, Hostname_current))')
#
```

```
# start db operations
con = dbapi.connect(db_name)
con.row_factory = dbapi.Row
con.text_factory = str
d = con.cursor()
print "\n con >", con, " con.cursor() >", d
#
d.execute('SELECT * FROM Exitstatistics ORDER BY Resourse')
x = 0
y = 0
try:
  for row in d:
     x = x +1
     print "\n  # of record read: ", x
     print "\n  value of row>", row
     resourse = row[0]
     search_key = row[1]
     date_inserted = row[2]
     hostname_current = row[3]
     hostip_address = row[4]
     lastpageno = row[5]
     entery_year = row[6]
     entery_month = row[7]
     entery_day = row[8]
     exit_day = row[9]
     day_differ = row[10]
     entery_hour = row[11]
     exit_hour = row[12]
     hour_differ = row[13]
     entery_minute = row[14]
     exit_minute = row[15]
     minute_differ = row[16]
     entery_second = row[17]
     exit_second = row[18]
     second_differ = row[19]
#                         1                   2
#      1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
     c.execute('INSERT INTO Exitstatistics
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',(resourse, search_key,
date_inserted, hostname_current, hostip_address, lastpageno, entery_year,
entery_month, entery_day, exit_day, day_differ, entery_hour, exit_hour,
hour_differ, entery_minute, exit_minute, minute_differ, entery_second,
exit_second, second_differ))
     com.commit()
     print "\n  record inserted "
     print resourse, search_key, date_inserted, hostname_current, hostip_address,
lastpageno, entery_year, entery_month, entery_day, exit_day, day_differ,
entery_hour, exit_hour, hour_differ, entery_minute, exit_minute, minute_differ,
entery_second, exit_second, second_differ
     y = y +1
except:
     print "\n error when read/insert>", row
#
print "\n  Tot. no. of record read: ", x
print "\n  Tot. no. of record inserted: ", y
print "\n statistics for %s " % db_name, " was reported."
# close the cursor we done with it
d.close()
c.close()
```

```
#! /usr/bin/python
# -*- coding: utf8 -*-
import urllib
from sgmllib import SGMLParser
import string
import urlparse
import os
import exceptions
import sys
import HTMLParser
import unicodedata
import re
import time
import sqlite3 as dbapi
from time import gmtime, strftime, localtime
import subprocess
from subprocess import Popen
import urllib2
from urllib2 import Request, urlopen, URLError
import urllib, htmllib, formatter
#
def find_duration(day_cum, hour_cum, minute_cum, second_cum, day_differ,
hour_differ, minute_differ, second_differ):  #  called by main
#############################################################################
#                                                                           #
#  finds caching time difference (in seconds) within the same URLs Url.   #
#                                                                           #
#############################################################################
#
        day_cum = day_cum + day_differ
        hour_cum = hour_cum + hour_differ
        minute_cum = minute_cum + minute_differ
        second_cum = second_cum + second_differ
#
        if second_cum >= 60:
                second_cum = second_cum - 60
                 minute_cum = minute_cum + 1
        if minute_cum >= 60:
                minute_cum = minute_cum -60
                hour_cum = hour_cum + 1
        if hour_cum >= 24:
                hour_cum = hour_cum - 24
                 day_cum = day_cum + 1
        print "\n cumulative time found: ", day_cum, " days, ", hour_cum, " hours,
", minute_cum, " minutes, ", second_cum, " seconds "
        return day_cum, hour_cum, minute_cum, second_cum
#
#############################################################################
#                                                                           #
# start main - reads all records in "Exitstat_db_name"+".db" and reports    #
#              statistics related to the db.                                #
#                                                                           #
#############################################################################
#
resource = [""]
del resource[0]
print "\n Enter Exitstat_db name:"
read = sys.stdin.readline()
found = str(read).find("\n")
resourse1 = read[0:found]
resource.append(read[0:found])
print "\n Exitstat_name entered (withouth .db):", resource
#
db_name = resourse1+".db"
print resourse1, db_name
#
# start db operations
con = dbapi.connect(db_name)
c = con.cursor()
print "\n con >", con
print "\n con.cursor() >", c
#
```

```
c.execute('CREATE TABLE IF NOT EXISTS Exitstatistics(Resourse TEXT, Search_key
TEXT, Date_inserted TEXT, Hostname_current TEXT, Hostip_address TEXT, Lastpageno
INTEGER, Entery_year INTEGER, Entery_month INTEGER, Entery_day INTEGER, Exit_day
INTEGER, Day_differ INTEGER, Entery_hour INTEGER, Exit_hour INTEGER, Hour_differ
INTEGER, Entery_minute INTEGER, Exit_minute INTEGER, Minute_differ INTEGER,
Entery_second INTEGER, Exit_second INTEGER, Second_differ INTEGER, PRIMARY KEY
(Resourse, Search_key, Date_inserted, Hostname_current))')
#
x = 0
y = 0
day_cum = 0
hour_cum = 0
minute_cum = 0
second_cum = 0
#                yyyymmddhhmmss
date_prevread = "20090110091011"
#
Altavista_count = 0
Altavista_day = 0
Altavista_hour = 0
Altavista_minute = 0
Altavista_second = 0
Aol_count = 0
Aol_day = 0
Aol_hour = 0
Aol_minute = 0
Aol_second = 0
Bing_count = 0
Bing_day = 0
Bing_hour = 0
Bing_minute = 0
Bing_second = 0
Google_count = 0
Google_day = 0
Google_hour = 0
Google_minute = 0
Google_second = 0
Yahoo_count = 0
Yahoo_day = 0
Yahoo_hour = 0
Yahoo_minute = 0
Yahoo_second = 0
duration = 0
#
con.text_factory = str
c.execute('SELECT * FROM Exitstatistics ORDER BY Resourse')
#
for row in c:
     x = x +1
     print "\n  # Record read: ", x
     print "\n  Row content>", row
#
     resourse = row[0]
     search_key = row[1]
     date_inserted = row[2]
     hostname_current = row[3]
     hostip_address = row[4]
     lastpageno = row[5]
     entery_year = row[6]
     entery_month = row[7]
     entery_day = row[8]
     exit_day = row[9]
     day_differ = row[10]
     entery_hour = row[11]
     exit_hour = row[12]
     hour_differ = row[13]
     entery_minute = row[14]
     exit_minute = row[15]
     minute_differ = row[16]
     entery_second = row[17]
     exit_second = row[18]
     second_differ = row[19]
#
     if row[0] == "altavista":
        Altavista_count = Altavista_count + 1
```

```
       (Altavista_day, Altavista_hour, Altavista_minute, Altavista_second) =
find_duration(Altavista_day, Altavista_hour, Altavista_minute, Altavista_second,
day_differ, hour_differ, minute_differ, second_differ)
     else:
       if row[0] == "aol":
           Aol_count = Aol_count + 1
           (Aol_day, Aol_hour, Aol_minute, Aol_second) = find_duration(Aol_day,
Aol_hour, Aol_minute, Aol_second, day_differ, hour_differ, minute_differ,
second_differ)
        else:
           if row[0] == "bing":
               Bing_count = Bing_count + 1
               (Bing_day, Bing_hour, Bing_minute, Bing_second) =
find_duration(Bing_day, Bing_hour, Bing_minute, Bing_second, day_differ,
hour_differ, minute_differ, second_differ)
           else:
              if row[0] == "google":
                  Google_count = Google_count + 1
                  (Google_day, Google_hour, Google_minute, Google_second) =
find_duration(Google_day, Google_hour, Google_minute, Google_second, day_differ,
hour_differ, minute_differ, second_differ)
              else:
                  Yahoo_count = Yahoo_count + 1
                  (Yahoo_day, Yahoo_hour, Yahoo_minute, Yahoo_second) =
find_duration(Yahoo_day, Yahoo_hour, Yahoo_minute, Yahoo_second, day_differ,
hour_differ, minute_differ, second_differ)
#
print "\n statistics for %s " % db_name
print " ================================="
if Altavista_count > 0:
   print "\n For %d Altavista runs found " % Altavista_count, "and %d days: " %
Altavista_day, "%d hours: " % Altavista_hour, "%d minutes: " % Altavista_minute,
"%d seconds " % Altavista_second, "was spent."
if Aol_count > 0:
   print "\n For %d Aol runs found " % Aol_count, "and %d days: " % Aol_day, "%d
hours: " % Aol_hour, "%d minutes: " % Aol_minute, "%d seconds " % Aol_second, "was
spent."
if Bing_count >0:
   print "\n For %d Bing runs found " % Bing_count, "and %d days: " % Bing_day,
"%d hours: " % Bing_hour, "%d minutes: " % Bing_minute, "%d seconds " %
Bing_second, "was spent."
if Google_count >0:
   print "\n For %d Google runs found " % Google_count, "and %d days: " %
Google_day, "%d hours: " % Google_hour, "%d minutes: " % Google_minute, "%d
seconds " % Google_second, "was spent."
if Yahoo_count >0:
   print "\n For %d Yahoo runs found " % Yahoo_count, "and %d days: " % Yahoo_day,
"%d hours: " % Yahoo_hour, "%d minutes: " % Yahoo_minute, "%d seconds " %
Yahoo_second, "was spent."
#
print "\n Total no. of records read: ", x
# close the cursor we done with it
```

Alexa_get.py

```
#!/usr/bin/python
# -*- coding: utf8 -*-             #!/home1/software/application/python-
2.5.4/bin/python
import os,re,urllib,pickle
db1 = 'glo.dump'
db2 = 'clong.dump'
clong = dict()
glo = [('name','cRank','cShort','linksIn')]
downCtr = 10

# Traffic Rank & Country Regular Expression
reg = re.compile('(\d+)</div>\W+<div class=\"label\">Traffic Rank in <a
href=[^\"]+\"([\w ]+)\">(\w+)</a>')
# LinksIn Regular Expression
linkedin = re.compile('<a href=\"/site/linksin/[\w.]+\">([\d,]+)</a>')

if __name__ == '__main__':
        # Load Data
        if os.path.exists(db1):
                dump = open(db1,'r')
                glo = pickle.load(dump)
                dump.close()
        else:
                # Initial Load from top-1m
                for line in open('top-1m.csv','r').readlines():
                        glo.append((line[:-1].split(',')[1],))
        if os.path.exists(db2):
                dump = open(db2,'r')
                clong = pickle.load(dump)
                dump.close()

        # Update Missing Data
#       for ctr in range(len(glo)):
#       for ctr in range(100001,200001):            inek41
#       for ctr in range(200001,400001):            inek42
#       for ctr in range(400001,600001):            inek43
#       for ctr in range(600001,800001):            inek44
#       for ctr in range(800001,1000001):           inek45
        for ctr in range(41610,100001):       #  100000 -> 100001
                print "\n counter=", ctr        # add_in
                if len(glo[ctr]) == 1:
                        st = glo[ctr][0]
                        c = urllib.urlopen('http://www.alexa.com/siteinfo/%s' %
st).read()
                        res = reg.findall(c)
                        linkres = linkedin.findall(c)
                        if res and linkres:
                                glo[ctr] =
(st,res[0][0],res[0][2],linkres[0].replace(',',''))
                                if not clong.has_key(res[0][2]):
                                        clong[res[0][2]]= res[0][1]
                        else:
                                print 'error in'+st

                        # Save & Print Data at every 100 updates
                        downCtr -= 1
                        if downCtr == 0 or ctr == 100000 :        # "or ctr ==
100000" added
                                downCtr += 100
                                print "\n downCtr= ", downCtr           # add_in
                                # Save Data
                                dump = open(db1,'w')
                                pickle.dump(glo,dump)
                                dump.close()
                                dump = open(db2,'w')
                                pickle.dump(clong,dump)
                                dump.close()
                                # Print Data
                                out = open('data.csv','w')
                                for ctr in range(len(glo)):

        out.write(str(ctr)+','+','.join(glo[ctr])+'\n')
```

```
out.close()
out = open('countryList.csv','w')
for key in clong:
        out.write(key + ',' + clong[key] + '\n')
out.close()
  print " lastkey_written:", key           # add_in
```

read_glo_dump_append_alexa_db_new

```
#!/usr/bin/python
# -*- coding: utf8 -*-             #!/home1/software/application/python-
2.5.4/bin/python
import os,re,urllib,pickle
import string
import sys
import exceptions
import HTMLParser
import htmllib, formatter
import unicodedata
import sqlite3 as dbapi
#
generic_tld =
["aero","asia","biz","cat","com","coop","edu","gov","info","int","jobs","mil","mob
i","museum","name",
"net","org","pro","tel","travel"]
#
cc_tld = ["ac","ad","ae","af","ag","ai","al","am",
"an","ao","aq","ar","as","at","au","aw","ax","az",
"ba","bb","bd","be","bf","bg","bh","bi","bj","bl",
"bm","bn","bo","br","bs","bt","bv","bw","by","bz",
"ca","cc","cd","cf","cg",
"ch","ci","ck","cl","cm","cn","co","cr","cu",
"cv","cx","cy","cz",
"de","dj","dk","dm","do","dz",
"ec","ee","eg","eh","er","es","et","eu",
"fi","fj","fk","fm","fo","fr",
"ga","gb","gd","ge","gf","gg","gh","gi","gl","gm","gn","gp",
"gq","gr","gs","gt","gu","gw","gy",
"hk","hm","hn","hr","ht","hu",
"id","ie","il","im","in","io","iq","ir","is","it",
"je","jm","jo","jp",
"ke","kg","kh","ki","km","kn","kp","kr","kw","ky",
"kz","la","lb","lc","li","lk","lr","ls","lt","lu","lv","ly",
"ma","mc","md","me","mf","mg","mh","mk","ml","mm",
"mn","mo","mp","mq","mr","ms","mt","mu","mv",
"mw","mx","my","mz",
"na","nc","ne","nf","ng","ni","nl","no","np","nr","nu","nz",
"om",
"pa","pe","pf","pg","ph","pk","pl","pm","pn",
"pr","ps","pt","pw","py",
"qa","re","ro","rs","ru","rw",
"sa","sb","sc","sd","se","sg","sh","si","sj",
"sk","sl","sm","sn","so","sr","st","su","sv","sy","sz",
"tc","td","tf","tg","th","tj","tk","tl","tm",
"tn","to","tl","tr","tt","tv","tw","tz",
"ua","ug","uk","um","us","uy","uz",
"va","vc","ve","vg","vi","vn","vu",
"wf","ws",
"ye","yt","yu",
"za","zm","zw"]
#
resource = [""]
del resource[0]
print "\n Enter input data .csv file name :"
read = sys.stdin.readline()
found = str(read).find("\n")
resourse1 = read[0:found]
resource.append(read[0:found])
print "\n file_name name entered (withouth .csv):", resource
#
csv_name = resourse1+".csv"
print resourse1, csv_name
#
#
print "\n enter starting record at %s" % csv_name
read = sys.stdin.readline()
first_rec = int(read)
print "\n %s" % csv_name, "file will be processed starting from first %d record" %
first_rec
#
print "\n enter last record which will be read at %s" % csv_name
```

```
read = sys.stdin.readline()
last_rec = int(read)
print "\n %s" % csv_name, "file will be processed up to last %d record
(inclusive)" % last_rec
#
#
# start db operations
# glo = [('cShort','cRank','name','wRank','linksIn')]
#       ['DE', '9', 'spiegel.de','136', '33030']
#
con = dbapi.connect('alexa.db', timeout=30)
con.row_factory = dbapi.Row
con.text_factory = str
c = con.cursor()
print "\n con >", con
print "\n con.cursor() >", c
#
c.execute('CREATE TABLE IF NOT EXISTS AlexaRecs(CShort TEXT, WRank INTEGER, Name
TEXT, CRank INTEGER, LinksIn INTEGER, PRIMARY KEY (CShort, WRank, Name))')
glo = []
f = open(csv_name,'r')
x = 0          # counter for the whole records read
i = 0          # counter for the records in between first_rec and last_rec
for line in f.readlines():
   print "\n line=", line
#    for x in range(first_rec,last_rec):
   if x >= first_rec and x <= last_rec:
        glo.append(line[:-1].split(','))
#       print "\n glo=", glo
       print "\n x=", x
       print "\n i=", i
       print "\n", glo[i]
       print "\n", glo[i][0]
       print "\n", glo[i][1]
       if len(glo[i]) > 2:                    # csv record read:
142,wretch.cc,2,TW,22075
#              ['137', 'sogou.com', '23', 'CN', '17143']
              print "\n", glo[i][2]
              print "\n", glo[i][3]
              print "\n", glo[i][4]
              cShort = glo[i][3]
              cRank = int(glo[i][2])
              name = glo[i][1]
              wRank = int(glo[i][0])
              linksIn = int(glo[i][4])
#              print cShort, wRank, name, cRank, linksIn
       else:                          # if csv record has two fields: 35,nasza-
klasa.pl
              cShort = " "
              cRank = 0
              name = glo[i][1]
              wRank = int(glo[i][0])
              linksIn = 0
               last_dot = 0
               first_slash = 0
              first_slash = name.find("/")
               if first_slash > 0:
                 name = name[:first_slash]
              last_dot = name.rfind(".")
              cShort = name[last_dot+1:last_dot+4]nasza-klasa.pl
              print "\n if len(glo[i]) < 2: 1st. cShort:", cShort
              k = 0
              found = 0
              for k, item in enumerate(generic_tld): # check if domain name exits
#                  print k, item
                   if item == cShort:
                      found = k
                      cShort = "UD"
#              k = generic_tld.index(cShort.lower())
              if found == 0: # if cShort does not exit at the generic_tld list
check country codes
                   cShort = name[last_dot+1:last_dot+3]
                  print "\n if len(glo[i]) < 2: 2nd. cShort:", cShort
                  j = 0
                  for j, item in enumerate(cc_tld):
#                      print j, item
```

```
                    if item == cShort:
                       found = j
                if found == 0:      # if cShort does not exit at the cc_tld list
it is called "undefined"
                    cShort = "UD"
#               print cShort, wRank, name, cRank, linksIn
#               else:
#                  cShort = "ud"

        cShort = cShort.upper()
        print cShort, wRank, name, cRank, linksIn
#                                                   1 2 3 4 5
        c.execute('INSERT INTO AlexaRecs VALUES(?,?,?,?,?)',(cShort, wRank, name,
cRank, linksIn))
#       Save (commit) the changes
        con.commit()
#       close the cursor we done with it
        c.close()
        i = i + 1
   x = x + 1
   if x > last_rec:
      f.close()
      sys.exit(1)
```

```
#!/usr/bin/python
import os,re,urllib,pickle
import string
import os
import sys
import exceptions
import HTMLParser
import htmllib, formatter
import unicodedata
import re
import sqlite3 as dbapi
#
#
cc_tld = ["ac","ad","ae","af","ag","ai","al","am",
"an","ao","aq","ar","as","at","au","aw","ax","az",
"ba","bb","bd","be","bf","bg","bh","bi","bj","bl",
"bm","bn","bo","br","bs","bt","bv","bw","by","bz",
"ca","cc","cd","cf","cg",
"ch","ci","ck","cl","cm","cn","co","cr","cu",
"cv","cx","cy","cz",
"de","dj","dk","dm","do","dz",
"ec","ee","eg","eh","er","es","et","eu",
"fi","fj","fk","fm","fo","fr",
"ga","gb","gd","ge","gf","gg","gh","gi","gl","gm","gn","gp",
"gq","gr","gs","gt","gu","gw","gy",
"hk","hm","hn","hr","ht","hu",
"id","ie","il","im","in","io","iq","ir","is","it",
"je","jm","jo","jp",
"ke","kg","kh","ki","km","kn","kp","kr","kw","ky",
"kz","la","lb","lc","li","lk","lr","ls","lt","lu","lv","ly",
"ma","mc","md","me","mf","mg","mh","mk","ml","mm",
"mn","mo","mp","mq","mr","ms","mt","mu","mv",
"mw","mx","my","mz",
"na","nc","ne","nf","ng","ni","nl","no","np","nr","nu","nz",
"om",
"pa","pe","pf","pg","ph","pk","pl","pm","pn",
"pr","ps","pt","pw","py",
"qa","re","ro","rs","ru","rw",
"sa","sb","sc","sd","se","sg","sh","si","sj",
"sk","sl","sm","sn","so","sr","st","su","sv","sy","sz",
"tc","td","tf","tg","th","tj","tk","tl","tm",
"tn","to","tl","tr","tt","tv","tw","tz",
"ua","ud","ug","uk","um","us","uy","uz",
"va","vc","ve","vg","vi","vn","vu",
"wf","ws",
"ye","yt","yu",
"za","zm","zw"]

country = ["Ascension Island","Andorra","United Arab
Emirates","Afghanistan","Antigua and Barbuda","Anguilla","Albania","Armenia",
"Netherlands Antilles","Angola","Antarctica","Argentina","American
Samoa","Austria","Australia","Aruba","Aland Islands","Azerbaijan",
"Bosnia and Herzegovina","Barbados","Bangladesh","Belgium","Burkina
Faso","Bulgaria","Bahrain","Burundi","Benin","Saint Barthelemy",
"Bermuda","Brunei","Bolivia","Brazil","Bahamas","Bhutan","Bouvet
Island","Botswana","Belarus","Belize",
"Canada","Cocos (Keeling) Islands","Democratic Republic of the Congo","Central
African Republic","Republic of the Congo",
"Switzerland","Cote d'Ivoire","Cook Islands","Chile","Cameroon","People's Republic
of China","Colombia","Costa Rica","Cuba",
"Cape Verde","Christmas Island","Cyprus","Czech Republic",
"Germany","Djibouti","Denmark","Dominica","Dominican Republic","Algeria",
"Ecuador","Estonia","Egypt","Western
Sahara","Eritrea","Spain","Ethiopia","European Union",
"Finland","Fiji","Falkland Islands","Federated States of Micronesia","Faroe
Islands","France",
"Gabon","United Kingdom","Grenada","Georgia","French
Guiana","Guernsey","Ghana","Gibraltar","Greenland","Gambia","Guinea","Guadeloupe",
"Equatorial Guinea","Greece","South Georgia and the South Sandwich
Islands","Guatemala","Guam","Guinea-Bissau","Guyana",
"Hong Kong","Heard Island and McDonald
Islands","Honduras","Croatia","Haiti","Hungary",
```

```
"Indonesia","Ireland","Israel","Isle of Man","India","British Indian Ocean
Territory","Iraq","Iran","Iceland","Italy",
"Jersey","Jamaica","Jordan","Japan",
"Kenya","Kyrgyzstan","Cambodia","Kiribati","Comoros","Saint Kitts and
Nevis","North Korea","South Korea","Kuwait","Cayman Islands",
"Kazakhstan","Laos","Lebanon","Saint Lucia","Liechtenstein","Sri
Lanka","Liberia","Lesotho","Lithuania","Luxembourg","Latvia","Libya",
"Morocco","Monaco","Moldova","Montenegro","Saint Martin","Madagascar","Marshall
Islands","Republic of Macedonia","Mali","Myanmar",
"Mongolia","Macau","Northern Mariana
Islands","Martinique","Mauritania","Montserrat","Malta","Mauritius","Maldives",
"Malawi","Mexico","Malaysia","Mozambique",
"Namibia","New Caledonia","Niger","Norfolk
Island","Nigeria","Nicaragua","Netherlands","Norway","Nepal","Nauru","Niue","New
Zealand",
"Oman",
"Panama","Peru","French Polynesia","Papua New
Guinea","Philippines","Pakistan","Poland","Saint Pierre and Miquelon","Pitcairn
Islands",
"Puerto Rico","Palestine","Portugal","Palau","Paraguay",
"Qatar","Reunion","Romania","Serbia","Russia","Rwanda",
"Saudi Arabia","Solomon Islands","Seychelles","Sudan","Sweden","Singapore","Saint
Helena","Slovenia","Svalbard and Jan Mayen islands",
"Slovakia","Sierra Leone","San Marino","Senegal","Somalia","Suriname","Sao Tome
and Principe","Soviet Union","El Salvador","Syria","Swaziland",
"Turks and Caicos Islands","Chad","French Southern and Antarctic
Lands","Togo","Thailand","Tajikistan","Tokelau","East Timor","Turkmenistan",
"Tunisia","Tonga","East Timor","Turkey","Trinidad and
Tobago","Tuvalu","Taiwan","Tanzania",
"Ukraine","UnDefined","Uganda","United Kingdom","US Minor Outlying
Islands","United States","Uruguay","Uzbekistan",
"Vatican City","Saint Vincent and the Grenadines","Venezuela","British Virgin
Islands","United States Virgin Islands","Vietnam","Vanuatu",
"Wallis and Futuna","Samoa (Western Samoa)",
"Yemen","Mayotte","Yugoslavia",
"South Africa","Zambia","Zimbabwe"]
#
#
con = dbapi.connect('alexa.db', timeout=30)
con.row_factory = dbapi.Row
con.text_factory = str
c = con.cursor()
print "\n con >", con
print "\n con.cursor() >", c
#
c.execute('CREATE TABLE IF NOT EXISTS AlexaRecs(CShort TEXT, WRank INTEGER, Name
TEXT, CRank INTEGER, LinksIn INTEGER, PRIMARY KEY (CShort, WRank, Name))')
#
con = dbapi.connect('summary.db', timeout=30)
con.row_factory = dbapi.Row
con.text_factory = str
d = con.cursor()
print "\n con >", con
print "\n con.cursor() >", d
#
d.execute('CREATE TABLE IF NOT EXISTS Summary(CShort text, CLong text,
Category_1000 integer, Category_1000_value integer, Category_1000_linksIn integer,
Average_1000 integer, Category_10000 integer, Category_10000_value integer,
Category_10000_linksIn integer, Average_10000 integer, Category_100000 integer,
Category_100000_value integer, Category_100000_linksIn integer, Average_100000
integer, Category_1000000 integer, Category_1000000_value integer,
Category_1000000_linksIn integer, Average_1000000 integer, PRIMARY KEY (CShort,
CLong))')
con.text_factory = str
c.execute('SELECT * FROM AlexaRecs ORDER BY CShort')
x = 0
y = 0
cShort = " "
cShort_eq = 0
category_1000 = 0
category_1000_value = 0
category_1000_linksIn = 0
average_1000 = 0
category_10000 = 0
category_10000_value = 0
category_10000_linksIn = 0
```

```
average_10000 = 0
category_100000 = 0
category_100000_value = 0
category_100000_linksIn = 0
average_100000 = 0
category_1000000 = 0
category_1000000_value = 0
category_1000000_linksIn = 0
average_1000000 = 0
for row in c:
     x = x +1
     print "\n  # Record read: ", x
     print "\n  Row content>", row
     if row[0] != cShort:
        if cShort_eq == 0:           # if the new root equals to previously read
           cShort_eq = 1
        else:
           if category_1000 != 0 : average_1000 = category_1000_linksIn /
category_1000
           if category_10000 != 0 : average_10000 = category_10000_linksIn /
category_10000
           if category_100000 != 0 : average_100000 = category_100000_linksIn /
category_100000
           if category_1000000 != 0 : average_1000000 = category_1000000_linksIn /
category_1000000
          i = 0
          j = 0
          for j, item in enumerate(cc_tld):
#             print j, item
              if item == cShort.lower():
                 i = j
           print "\n cShort.lower():", cShort.lower(), "  i:", i, " country[i]:",
country[i]
          cLong = country[i]
#          write to summary.db
          print "\n country code", cShort, " country ", cLong, "\n tot.no.of webs
at category_1000 ", category_1000, " sum of positions at category_1000
",category_1000_value, " sum of linksIn ref. val. at category_1000 ",
category_1000_linksIn, "average of linksIn at category_1000 ", average_1000, "\n
tot.no.of webs at category_10000 ",category_10000,  " sum of positions at
category_10000 ",category_10000_value, " sum of linksIn ref. val. at
category_10000 ", category_10000_linksIn, "average of linksIn at category_10000 ",
average_10000, "\n tot.no.of webs at category_100000 ",category_100000, " sum of
positions at category_100000 ",category_100000_value, " sum of linksIn ref. val.
at category_100000 ", category_100000_linksIn, "average of linksIn at
category_100000 ", average_100000, "\n tot.no.of webs at category_1000000
",category_1000000," sum of positions at category_1000000 ",
category_1000000_value, " sum of linksIn ref. val. at category_1000000 ",
category_1000000_linksIn, "average of linksIn at category_1000000 ",
average_1000000
#                                                                    1
#                                          1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
          d.execute('INSERT INTO Summary
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',(cShort, cLong, category_1000,
category_1000_value, category_1000_linksIn, average_1000, category_10000,
category_10000_value, category_10000_linksIn, average_10000, category_100000,
category_100000_value, category_100000_linksIn, average_100000, category_1000000,
category_1000000_value, category_1000000_linksIn, average_1000000))
#          Save (commit) the changes
          con.commit()
#          close the cursor we done with it
          d.close()
           y = y + 1
           category_1000 = 0
           category_1000_value = 0
           category_1000_linksIn = 0
           average_1000 = 0
           category_10000 = 0
           category_10000_value = 0
           category_10000_linksIn = 0
           average_10000 = 0
           category_100000 = 0
           category_100000_value = 0
           category_100000_linksIn = 0
           average_100000 = 0
           category_1000000 = 0
```

```
            category_1000000_value = 0
            category_1000000_linksIn = 0
            average_1000000 = 0
      cShort = row[0]
      wRank = row[1]
      name = row[2]
      cRank = row [3]
      linksIn = row[4]
      if int(wRank) <= 1000:
         category_1000 = category_1000 + 1
         category_1000_value = category_1000_value + 1000000 -int(wRank)
        category_1000_linksIn = category_1000_linksIn + int(linksIn)
      if int(wRank) > 1000 and int(wRank) <= 10000:
         category_10000 = category_10000 + 1
         category_10000_value = category_10000_value + 1000000 - int(wRank)
        category_10000_linksIn = category_10000_linksIn + int(linksIn)
      if int(wRank) > 10000 and int(wRank) <= 100000:
         category_100000 = category_100000 + 1
         category_100000_value = category_100000_value + 1000000 - int(wRank)
        category_100000_linksIn = category_100000_linksIn + int(linksIn)
      if int(wRank) > 100000 and int(wRank) <= 1000000:
         category_1000000 = category_1000000 + 1
         category_1000000_value = category_1000000_value + 1000000 - int(wRank)
        category_1000000_linksIn = category_1000000_linksIn + int(linksIn)
# close the cursor we done with it
c.close()
if category_1000 != 0 : average_1000 = category_1000_linksIn / category_1000
if category_10000 != 0 : average_10000 = category_10000_linksIn / category_10000
if category_100000 != 0 : average_100000 = category_100000_linksIn /
category_100000
if category_1000000 != 0 : average_1000000 = category_1000000_linksIn /
category_1000000
i = 0
j = 0
for j, item in enumerate(cc_tld):
#  print j, item
   if item == cShort.lower():
      i = j
print "\n cShort.lower():", cShort.lower(), "  i:", i, " country[i]:", country[i]
cLong = country[i]
#          write to summary.db
print "\n country code", cShort, " country ", cLong, "\n tot.no.of webs at
category_1000 ", category_1000, " sum of positions at category_1000
",category_1000_value, " sum of linksIn ref. val. at category_1000 ",
category_1000_linksIn, "average of linksIn at category_1000 ", average_1000, "\n
tot.no.of webs at category_10000 ",category_10000,  " sum of positions at
category_10000 ",category_10000_value, " sum of linksIn ref. val. at
category_10000 ", category_10000_linksIn, "average of linksIn at category_10000 ",
average_10000, "\n tot.no.of webs at category_100000 ",category_100000, " sum of
positions at category_100000 ",category_100000_value, " sum of linksIn ref. val.
at category_100000 ", category_100000_linksIn, "average of linksIn at
category_100000 ", average_100000, "\n tot.no.of webs at category_1000000
",category_1000000," sum of positions at category_1000000 ",
category_1000000_value, " sum of linksIn ref. val. at category_1000000 ",
category_1000000_linksIn, "average of linksIn at category_1000000 ",
average_1000000
#                                                1
#                                  1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
d.execute('INSERT INTO Summary
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',(cShort, cLong, category_1000,
category_1000_value, category_1000_linksIn, average_1000, category_10000,
category_10000_value, category_10000_linksIn, average_10000, category_100000,
category_100000_value, category_100000_linksIn, average_100000, category_1000000,
category_1000000_value, category_1000000_linksIn, average_1000000))
#          Save (commit) the changes
con.commit()
#          close the cursor we done with it
d.close()
x = 0
con.text_factory = str
d.execute('SELECT * FROM Summary ORDER BY CLong')
for row in d:
     x = x +1
     print "\n  # Record read: ", x
     print "\n  Row content>", row
d.close()
```

```
#! /usr/bin/python
# -*- coding: utf8 -*-             #!/home1/software/application/python-
2.5.4/bin/python
import sgmllib, urllib, urlparse
from sgmllib import SGMLParser
import string
import os
import sys
import exceptions
import HTMLParser
import htmllib, formatter
import unicodedata
import re
import sqlite3 as dbapi
import time
from time import gmtime, strftime, localtime
import subprocess
from subprocess import Popen
import socket
import urllib2
from urllib2 import Request, urlopen, URLError
#
def count_found():            # called by "main"
######################################################################
#                                                                    #
#  counts and finds in which list has max.                           #
#  turkish chars and words.                                          #
#                                                                    #
######################################################################
#
        global trword_keys, trword_values
        trword_keys = []
         trword_values = []
         count_trword = 0
         count_trword_u = 0
         count_trword_utf8 = 0
        for x in trword:
              count_trword = count_trword + trword[x]
        for x in trword_u:
              count_trword_u = count_trword_u + trword_u[x]
        for x in trword_utf8:
              count_trword_utf8 = count_trword_utf8 + trword_utf8[x]
        if count_trword > count_trword_u:
           if count_trword > count_trword_utf8:      # count_trword >= all
              trword_keys = trword.keys()
              trword_values = trword.values()
           else:                                     # count_trword_utf8 > all
              trword_keys = trword_utf8.keys()
              trword_values = trword_utf8.values()
         else:                                       # count_trword_u > trword
           if count_trword_u > count_trword_utf8:    # count_trword_u > all
              trword_keys = trword_u.keys()
              trword_values = trword_u.values()
           else:
              trword_keys = trword_utf8.keys()
              trword_values = trword_utf8.values()
#
def timentry():               #  called by "main"
############################################################
#                                                          #
#  gives time stamp when query starts at the Url.          #
#                                                          #
############################################################
#
        print "\n localtime():", localtime()
        (year, month, day, hour, minute, second, weekday, dayofyear, dst) =
localtime()
        entery_year = year
        entery_month = month
        entery_day = day
        entery_hour = hour
        entery_minute = minute
```

```
        entery_second = second
        print "\n entery hour:", entery_hour, " entery minute:", entery_minute, "
entery second:", entery_second
        epocha = localtime()
        return entery_year, entery_month, entery_day, entery_hour, entery_minute,
entery_second
#
def timexit(entery_year, entery_month, entery_day, entery_hour, entery_minute,
entery_second, mainexit_status, nation):
#       called by "main"
#####################################################################
#                                                                   #
# gives time stamp when query ends and query time span at the Url. #
#                                                                   #
#####################################################################
#
        global exit_year, exit_month, exit_day, exit_hour, exit_minute,
exit_second
        (year, month, day, hour, minute, second, weekday, dayofyear, dst) =
localtime()
        exit_year = year
        exit_month = month
        exit_day = day
        exit_hour = hour
        exit_minute = minute
        exit_second = second
        print "\n entery_day:", entery_day, " entery hour:", entery_hour, " entery
minute:", entery_minute, " entery second:", entery_second
        print "\n   exit_day:", exit_day,   "   exit hour:", exit_hour,   "   exit
minute:", exit_minute,   "   exit second:", exit_second
#
        epochb = localtime()
#
        if entery_second > exit_second:
                exit_second = exit_second + 60
                if exit_minute > 0:
                          exit_minute = exit_minute -1
                 else:
                          exit_minute = 59
                          exit_hour = exit_hour -1
        if entery_minute > exit_minute:
                exit_minute = exit_minute + 60
                 exit_hour = exit_hour -1
        if entery_hour > exit_hour:
                exit_hour = exit_hour + 24
                exit_day = exit_day -1
        if entery_day > exit_day + 1:                       # <<< 22.09.2009
                entety_day = exit_day                       # <<< 22.09.2009
         else:                                              # <<< 22.09.2009
                entety_day = entery_day                     # <<< 22.09.2009

        print "\n   day dif.:", exit_day-entety_day,"    hour dif.:", exit_hour-
entery_hour, "    minute dif.:", exit_minute-entery_minute,\
                "    second dif.:", exit_second-entery_second
#
        day_differ = exit_day-entety_day
        hour_differ = exit_hour-entery_hour
        minute_differ = exit_minute-entery_minute
        second_differ = exit_second-entery_second
#
        if exit_second >= 60:                               # <<< 22.09.2009
              exit_second = exit_second - 60                # <<< 22.09.2009
              exit_minute = exit_minute + 1                 # <<< 22.09.2009
        if exit_minute >= 60:                               # <<< 22.09.2009
              exit_minute = exit_minute - 60                # <<< 22.09.2009
              exit_hour = exit_hour + 1                     # <<< 22.09.2009
        if exit_hour >= 24:                                 # <<< 22.09.2009
              exit_hour = exit_hour - 24                    # <<< 22.09.2009
              exit_day = exit_day + 1                       # <<< 22.09.2009
#
         if mainexit_status == 1:
                print nation, date_inserted, hostname_current, hostip_address,
entery_year, entery_month, entery_day, exit_day, day_differ, entery_hour,
exit_hour, hour_differ, entery_minute, exit_minute, minute_differ, entery_second,
exit_second, second_differ
```

```
#
def find_duration():          #  called by "web_search"
#####################################################################
#                                                                   #
#  finds partial time spans for the addresses within the same Url. #
#                                                                   #
#####################################################################
#
        (year, month, day, hour, minute, second, weekday, dayofyear, dst) =
localtime()
        check_exit_day = day
        check_exit_hour = hour
        check_exit_minute = minute
        check_exit_second = second
#       duration = 0               <<<<<<<<<<<<<<<<<<< fault
#
        epochb = localtime()
#
        if entery_second > check_exit_second:
                check_exit_second = check_exit_second + 60
                if check_exit_minute > 0:
                          check_exit_minute = check_exit_minute -1
                 else:
                          check_exit_minute = 59
                          check_exit_hour = check_exit_hour -1
        if entery_minute > check_exit_minute:
                check_exit_minute = check_exit_minute + 60
                 check_exit_hour = check_exit_hour -1
        if entery_hour > check_exit_hour:
                check_exit_hour = check_exit_hour + 24
                check_exit_day = check_exit_day -1
#       print "\n  minute difference:", check_exit_minute-entery_minute,\
                "    second dif.:", check_exit_second-entery_second
        duration = (check_exit_minute*60 + check_exit_second) - (entery_minute *
60 + entery_second)
#       print "\n duration in find_duration", duration
        return duration
#
def formatdecimal(amount):            #  called by "web_search"
#####################################################################
#                                                                   #
#  used for printing integer values in decimal format                    #
#                                                                   #
#####################################################################
#
        str1 = str(amount)
        new = ' '
        n = 0
        while len(str1) > 3:
           m = n-3
           new = '.' + str1[-3:] + new
           n = n - 4
           str1 = str1[:-3]
        else:
           m = n-1
           new = str1[-3:] + new
        return "%s" % new

#
class FindLinks(SGMLParser):          #  called by "findUrl_links"
#####################################################################
#                                                                   #
# parses the HTML and extract the link info which specified as HREF#
#  attributes of the <A> tags,                                      #
#                                                                   #
#####################################################################
#
    def __init__(self):               # method is a constructor and is
automatically called when an intance of the FindLinks is created.
        self.links = []               # initializes the links intance variable to
an empty list and tehn calls the base class constructor
        SGMLParser.__init__(self)
    def do_a(self, attributes):                     # invoked by the SGML parser
when it encounters an <A> tag,
        global count_href                           #   SGML encounters a <F00>
tag corresponding to <A> tag
```

```
        for (name, value) in attributes:

          try:                                          # <<< 20.09.2009
            if name == "href":

                c_value = value
                i = string.find(c_value, '#')         # find str length up to #
char.
                if i >= 0:
                    c_value = c_value[:i]             # Remove # fragment
                i = string.find(c_value, '?')         # find str length up to ?
char.
                if i >= 0:
                    c_value = c_value[:i]             # Remove ? fragment
                i = string.find(c_value, '!')         # find str length up to !
char.
                if i >= 0:
                    c_value = c_value[:i]             # Remove ! fragment
                i = string.find(c_value, 'busy.htm')  # find str length up to
busy.htm str.
                if i >= 0:
                    c_value = c_value[:i]             # Remove busy.htm fragment
                i = string.find(c_value, '<!')        # find str length up to <!
chars.
                if i >= 0:
                    c_value = c_value[:i]             # Remove <!? fragment

                words = string.split(c_value)         # Split in whitespace
delimited words return
                string.join(words, "")                # Join words without
whitespace
                value = c_value

                if self.links.count(value) <= 0 :     # new if new address/value
does not exist in the list
                    if value != "":                   # if value is not null
                         if value[:5] == "http:":     # if it is a "http:" address

                                      self.links.append(value)      # new add new
addr. to the list
                               count_href = count_href + 1

          except sgmllib.SGMLParseError, ex:                       # <<<
20.09.2009
              print "Pythonlib's error message: " + str(ex) # <<< 20.09.2009
              line, offset = parser.getpos()                       # <<<
20.09.2009
              lines = parser.rawdata.split("\n")                   # <<<
20.09.2009
              print "My extra information: error at line %d offset %d" %
parser.getpos()        # <<< 20.09.2009
             print lines[line]                                     # <<<
20.09.2009
              print "%*s" % (offset, "^")                          # <<<
20.09.2009
              parser = None                                        # <<<
20.09.2009
#
    def do_img(self, attributes):                     # invoked by the SGML parser
when it encounters an <img> tag
        global count_img, imagelst
        for (name, value) in attributes:
            if name == "src":
              if imagelst.count(value) <= 0 :      # new if new address/value
does not exist in the list
                  if value != "":                  # if value is not null
                      imagelst.append(value)       # new add new addr. to the
list
#                     print "\n *********** img ************ "
#                     print "\n self:", self, "\n name:", name, "\n  value:",
value, "\n attributes:", attributes
                             count_img += 1
#                     print "\n # of img:", count_img
#
    def do_form(self, attributes):                    # invoked by the SGML parser
when it encounters an <form> tag
```

```
        global count_form, formlst
        for (name, value) in attributes:
#           if (name == "id" and value == "aspnetForm") or value.find("form"):
            if name == "id" and value == "aspnetForm":
#               print "\n *********** form ************ "
#               print "\n self:", self, "\n name:", name, "\n  value:", value, "\n
attributes:", attributes
                    count_form += 1
#              print "\n # of form:", count_form
    def getlinks(self):
#       print "\n self only >", self
        return self.links
#
# define our method
def replace_all(data_str, dic):
    for i, j in dic.iteritems():
        data_str = data_str.replace(i, j)
    return data_str
def findUrl_links(url, responses, error_count):
# called by "web_search"
#####################################################################
#                                                                   #
#  returns content and MIME type of a Url.                          #
#                                                                   #
#####################################################################
#
        global urlsize
        global wordref_lst            # <<<<<<<< new   21.03.2010
        global reps1, reps2
        global charset, charset_found
        fulllinks = []                                      ###06.09.2009#######
        ms_reason = " "
        # <<< 20.09.2009
        ms_status = 0
                                                            # <<< 20.09.2009
        reps = {'www':' ', 'href':' ', 'http':' ', 'id=':' ', '</span>':' ',
'</td>':' ', '</div>':' ', '</li>':' ', '<b><br':' ', '.':' ', '"':' ', '1':' ',
'2':' ', '3':' ', '4':' ', '5':' ', '6':' ', '7':' ', '8':' ', '9':' ', '0':' ',
'*':' ', ',':' ', '<':' ', 'Ã©':' ', '!':' ', '^':' ', '+':' ', '%':' ', '&':' ',
'/':' ', '(':' ', ')':' ', '=':' ', '?':' ',  ';':' ', '>':' ', ':':' ', '#':' ',
'$':' ', '{':' ', '[':' ', ']':' ', '}':' ', '@':' ', 'â,¬':' ', 'Â¨':' ', '~':'
', 'Â´':' ', '`':' ', '|':' ', '-':' ', '_':' ', 'Ã¾':'ÅŸ', 'Ã½':'Ä±', 'Ã□':'i',
'Ã°':'ÄŸ', 'Ãž':'ÅŸ', 'Â©':' ', 'Ã‡': 'Ã§', 'Ã-': 'Ã¶', 'Ãœ': 'Ã¼', 'Åž': 'ÅŸ',
'Äž': 'ÄŸ', 'WQQ':' ', 'XQQ':' '}
#       reps2 = {'\xfe':'ÅŸ', '\xc5\x9f':'ÅŸ' ,'\xfd':'Ä±', '\xc4\xb1':'Ä±',
'\xdd':'I', '\xfc':'Ã¼', '\xc3\xbc':'Ãœ', '\xf0':'ÄŸ', '\xc4\x9f':'ÄŸ',
'\xd6':'Ã¶', '\xe7':'Ã§', '\\':' ',  '\'':' '}
        reps1 = {}
        reps2 = {u'Ã¾':u'ÅŸ', u'Ã½':u'Ä±', u'Ã□':u'Ä°', u'Ã°':u'ÄŸ',
u'Äž':u'Åž',u'Ã‡': u'Ã§', u'Ã-': u'Ã¶', u'Ãœ': u'Ã¼', u'Åž': u'ÅŸ', u'Äž': u'ÄŸ',
}
#
#       excluded:  '\'':' ', '''''':' ',
#
#       timeout in seconds
        timeout = 10
        # <<< 20.09.2009
        socket.setdefaulttimeout(timeout)
        # <<< 20.09.2009

#       this call to urllib2.urlopen now uses the default timeout
        # <<< 20.09.2009
#       we have set in the socket module                    # <<< 20.09.2009

        try:                                                # <<< 03.09.2009
            charset_found = " "
            charset = " "
            f = urllib2.urlopen(url)                        # <<< 20.09.2009
            data = f.read()
            headers = f.info()
            f.close()
            if headers.has_key("content-type"):

                mimetype = headers["content-type"]
                print " mimetype>",mimetype
                 charset_found = mimetype[19:]
```

```
                print " charset_found=", charset_found
            else:
                # No content-type header -- guess type based on extension
                if url[-5:] == ".html":
                    mimetype = "text/html"
                else:
                # Don't know -- worst-case assumption
                    mimetype = "application/octet-stream"
#
            if charset_found == "ISO-8859-9" or charset_found == "iso-8859-9" or
charset_found == "so-8859-9" or charset_found == "SO-8859-9":
                charset = "iso-8859-9"
            else:
                if  charset_found == "ISO-8859-1" or charset_found == "iso-8859-1"
or charset_found == "so-8859-1" or charset_found == "SO-8859-1":
                    charset = "iso-8859-1"
                else:
#                   if charset_found == "UTF-8" or charset_found == "utf-8" or
charset_found == "tf-8" or charset_found == "TF-8":
#                      charset = "utf-8"
#                   else:
                      if charset_found == "WINDOWS-1254" or charset_found ==
"windows-1254":
                         charset = "windows-1254"
#
            print " charset=", charset
            if charset != " ":
#             replace "reps1" table each time according to the charset found
              for i, j in reps.iteritems():
                  a = str(i)
                  b = str(j)
                  print "\n a=", a, "  b=", b, "  j=", j
                  if not reps1.has_key(a):
                     if b != " ":
                        reps1[a] = b
                     else:
                        reps1[a] = " "
#
               print "\n reps1 after first replacement:", reps1
#
              inFileTxt = 'sed_inputs.txt'
              s = open(inFileTxt,'r')
              line = s.readline()
#             l = s.readline()
#              line = l.decode(charset)
              while line:
                  print "\n sed_input file read>",line      # variables to be
replaced in the file
                  splitline = string.split(line)
                  (c, d) = splitline
                  print "\n c:", c, "  d:", d
                   tobe_changed = c
                   proposed_change = d
                  print "\n tobe_changed:", tobe_changed, "  proposed_change:",
proposed_change
                  if not reps1.has_key(tobe_changed):
                      if proposed_change != "null":
                         reps1[tobe_changed] = proposed_change
                      else:
                         reps1[tobe_changed] = " "
                   line = s.readline()
              s.close()
              print "\n reps1 after second replacement:", reps1
#       bind the returned text of the method
#       to a variable and print it
              print "\n ********************** data before decoded >", data
#             data = data.decode(charset)
#             print "\n ********************** data after decoded >", data
             data_new = replace_all(data, reps1)
             print "\n >>>>>>>>>>>>>>> data_new>", data_new

             g = open("file_encode.txt", 'w')             # <<<<<<< new
              g.write(data_new)                           # <<<<<<< new
             g.close()                                    # <<<<<<< new
```

```
              splitline = string.split(data_new)            # <<<<<<<<< new
22.03.2010

              for word in splitline:
#                 word = replace_all(word, reps2)
#                 words = string.split(word)         # Split in whitespace
delimited words return
                  string.join(word, "")              # Join words without
whitespace

                   if len(word) > 1:
#                    print "\n word in splitline>", word.lower()
                     if not wordref_lst.has_key(word.lower()):      # if url/key
does not exist at the linkrefs
                        wordref_lst[word.lower()] = 1               # add url/key
to the dictionary

#                       print "\n key & link are new in linkrefs, url>",
wordref_lst
                     else:
                        wordref_lst[word.lower()] = wordref_lst[word.lower()] + 1
# if url/key exists at the linkrefs, add link as value to the url/key
              print "\n   ****************************"
              print "   *                              *"
              print "   * HEADERS INFO >", headers
              print "   *                              *"
              print "   ****************************"

#       start of getlinks
              timeout = 10                                          # <<<
20.09.2009
              socket.setdefaulttimeout(timeout)                     # <<<
20.09.2009
              try:                                                  # <<<
20.09.2009
#                 print " mimetype[:9] >", mimetype[:9]
                  if mimetype[:9] == "text/html":
                       parser = FindLinks()     # call FindLinks() to activate
SGML parser which extracts Links
#                      print "\n parser >", parser

                       parser.feed(data)             # parse fed data
                       parser.close()
                       links = parser.getlinks()    # ***getlinks()***
                  else:
                       links = []                    # Non-HTML data has no links
                   for link in links:                               # <<<
20.09.2009
                        full = urlparse.urljoin(url, link)          # <<<
20.09.2009
                        fulllinks.append(full)                      # <<<
20.09.2009
#
              except sgmllib.SGMLParseError, ex:                    # <<<
20.09.2009
                  print "Pythonlib's error message: " + str(ex)     # <<<
20.09.2009
                  line, offset = parser.getpos()                    # <<<
20.09.2009
                  lines = parser.rawdata.split("\n")        # <<< 20.09.2009
                  print "My extra information: error at line %d offset %d" %
parser.getpos()       # <<< 20.09.2009
                  print lines[line]                                 # <<<
20.09.2009
                  print "%*s" % (offset, "^")                       # <<<
20.09.2009
                  parser = None                                     # <<<
20.09.2009

        except (socket.error, IOError, timeout, Exception), e:      # <<<
24.09.2009
            if hasattr(e, 'code'):                                  # <<<
24.09.2009
                ms_status = e.code                                  # <<<
24.09.2009
```

```
            if hasattr(e, 'reason'):                                # <<<
24.09.2009
                ms_reason = e.reason                                # <<<
24.09.2009
#
        return fulllinks, ms_status, ms_reason, error_count
def web_search(root, responses, error_count):
#       called by "main"
#####################################################################
#                                                                   #
#  follows internal links to find all reachable documents in the    #
#     tree and uses depth search methos for internal query           #
#                                                                   #
#####################################################################
#
        global count_form, count_href, count_img, firstlevel, imagelst, urlsize,
urlwidth
        global trword, trword_u, trword_utf8
         global no_of_internal_links , no_of_external_links
        global doneLst                                          # <<< 14.09.2009
        global duration                                         # <<< 19.09.2009

        trword =
{'bir':0,'olmak':0,'iÃ§in':0,'ben':0,'demek':0,'Ã§ok':0,'yapmak':0,'gibi':0,'daha'
:0,'almak':0,
'var':0,'kendi':0,'gelmek':0,'ile':0,'vermek':0,'ama':0,'sonra':0,'kadar':0,'yer':
0,'insan':0,
'deÄŸil':0,'her':0,'istemek':0,'yÄ±l':0,'Ã§Ä±kmak':0,'gÃ¶rmek':0,'gÃ¼n':0,'biz':0,
'gitmek':0,'iÅŸ':0,
'Åžey':0,'ara':0,'bilmek':0,'zaman':0,'Ã§ocuk':0,'iki':0,'bakmak':0,'Ã§alÄ±ÅŸmak':
0,'iÃ§inde':0,'bÃ¼yÃ¼k':0
,'yok':0,'baÅŸlamak':0,'yol':0,'kalmak':0,'neden':0,'siz':0,'konu':0,'yapÄ±lmak':0
,'iyi':0,'kadÄ±n':0}
        trword_u =
{u'bir':0,u'olmak':0,u'iÃ§in':0,u'ben':0,u'demek':0,u'Ã§ok':0,u'yapmak':0,u'gibi':
0,u'daha':0,u'almak':0,
u'var':0,u'kendi':0,u'gelmek':0,u'ile':0,u'vermek':0,u'ama':0,u'sonra':0,u'kadar':
0,u'yer':0,u'insan':0,
u'deÄŸil':0,u'her':0,u'istemek':0,u'yÄ±l':0,u'Ã§Ä±kmak':0,u'gÃ¶rmek':0,u'gÃ¼n':0,u
'biz':0,u'gitmek':0,u'iÅŸ':0,
u'Åžey':0,u'ara':0,u'bilmek':0,u'zaman':0,u'Ã§ocuk':0,u'iki':0,u'bakmak':0,u'Ã§alÄ
±ÅŸmak':0,u'iÃ§inde':0,u'bÃ¼yÃ¼k':0,
u'yok':0,u'baÅŸlamak':0,u'yol':0,u'kalmak':0,u'neden':0,u'siz':0,u'konu':0,u'yapÄ±
lmak':0,u'iyi':0,u'kadÄ±n':0}

        trword_utf8 =
{u'bir':0,u'olmak':0,u'iÃ§in':0,u'ben':0,u'demek':0,u'Ã§ok':0,u'yapmak':0,u'gibi':
0,u'daha':0,u'almak':0,
u'var':0,u'kendi':0,u'gelmek':0,u'ile':0,u'vermek':0,u'ama':0,u'sonra':0,u'kadar':
0,u'yer':0,u'insan':0,
u'deÄŸil':0,u'her':0,u'istemek':0,u'yÄ±l':0,u'Ã§Ä±kmak':0,u'gÃ¶rmek':0,u'gÃ¼n':0,u
'biz':0,u'gitmek':0,u'iÅŸ':0,
u'Åžey':0,u'ara':0,u'bilmek':0,u'zaman':0,u'Ã§ocuk':0,u'iki':0,u'bakmak':0,u'Ã§alÄ
±ÅŸmak':0,u'iÃ§inde':0,u'bÃ¼yÃ¼k':0,
u'yok':0,u'baÅŸlamak':0,u'yol':0,u'kalmak':0,u'neden':0,u'siz':0,u'konu':0,u'yapÄ±
lmak':0,u'iyi':0,u'kadÄ±n':0}
#
        todo = [root]                           # list of Url.s which will be
searched
#       print "\n root >", root, " todo>", todo
#       doneLst = []                            # list of Url.s search were
completed
        linkrefs = {root: ["<ROOT>"]}
        linkcount_inkeys = {root:0}
        external_links = []                     # contains external links
        urlsize = 0
        urldebth = 0
        urlwidth = 0
        count_href = 0                          # counts # href links within the
Url
        count_img = 0                           # counts # images within the Url
        count_form = 0
        newlinks = []
        firstlevel = []
         level_depth = 0
        imagelst = []
```

```
        formlst = []
        lasturl_atfirstlevel = []
        lasturl_atlevel = ''
        stop_it = 1
        check_level = 0
         duration = 0
        msg_status = 0
        msg_reason = " "
#
        print "\n ********************************************************"
        print "\n max_duration:", max_duration, "  max_depth:", max_depth
        print "\n ********************************************************"
#
         while todo and todo[0] not in doneLst and duration < max_duration and
check_level < max_depth: # while Url/root exists and it does not processed
                                            # before
#           if todo[0] not in doneLst and check_level < 4:
              print "\n duration in web_search(root)>", duration
              url = todo[0]
              doneLst.append(url)           # put control for Url/root which must
exist only once!
#              print "\n url>>>>>>", url, "
lasturl_atfirstlevel>",lasturl_atfirstlevel
#              print "\n todo>", todo
#              print "\n doneLst.append(url)>", doneLst
              del todo[0]
              print "\n Checking in FindLinks>", url
#
               try:
#################>
                  (newlinks, msg_status, msg_reason, error_count) =
findUrl_links(url, responses, error_count)
#                 print "\n *** newlinks>", newlinks
                  if urlwidth == 0:
                       if len(newlinks) > 0:
                                       firstlevel = newlinks
                               urlwidth = len(newlinks)
                               x = urlwidth
                               while (x):
                                       x = x-1
                                       lasturl_atfirstlevel = newlinks[x]
                                       lasturl_atlevel = newlinks[x]
                                       if lasturl_atfirstlevel[:len(root)] == root:
                                               x = 0
               except (IOError,URLError), e:
                  if hasattr(e, 'reason'):
                      print 'We failed to reach a server.', url
                      print 'Reason: ', e.reason
                       msg_reason = e.reason
                  elif hasattr(e, 'code'):
                      print 'The server couldn\'t fulfill the request.'
                      print 'Error code: ', e.code
                      msg_status = e.code
                       for ref in linkrefs[url]:
                           print "\t *******>" + ref
                      continue
               for link in newlinks:
                   print "\n * link>", link
                  if not linkrefs.has_key(url):      # if url/key does not exist
at the linkrefs
                      linkrefs[url] = [link]         # add url/key to the
dictionary
                      linkcount_inkeys[url] = 1
#                     print " key & link are new in linkrefs, url>", url, "
link>", link
                   else:
                      linkrefs[url].append(link)    # if url/key exists at the
linkrefs, add link as value to the url/key
                       linkcount_inkeys[url] = linkcount_inkeys[url] + 1

                  level_depth = linkcount_inkeys[url]
#
                   if link[:len(root)] == root:
                       if (link not in doneLst and link not in todo):
                          todo.append(link)
                  else:
```

```
                    if link not in external_links:
                        external_links.append(link)
                if url == lasturl_atlevel:
                    stop_it = 0
                     if len(todo) > 1:
                       lasturl_atlevel = todo[len(todo) -1]
                       x = len(todo)
                        while (x):
                            x = x-1
                            lasturl_atlevel = todo[x]
                            if lasturl_atlevel[:len(root)] == root:
                                    x = 0
                                    check_level = check_level + 1
                                    print "\n   ******************************"
                                    print "   *                            *"
                                    print "   * check_level>", check_level, "
*"
                                    print "   *                            *"
                                    print "   ******************************"
             print "\n        url>>>>>>", url, "\n lasturl_atlevel>",
lasturl_atlevel
              duration = find_duration()                  # 06.09.2009 exit from
url if duration of query exceeds 10 minutes
              print "\n query duration at the same Url:", duration
#
        print "\n total length of ", root, " is: ", formatdecimal(urlsize), "
bytes"
        print "\n width of %s is %d and Url adresses at the first level: %s " %
(root, urlwidth, firstlevel)
        print "\n last internal address at the first level:", lasturl_atfirstlevel
        print "\n %s contains %d hrefs and %d images, %d forms." % (root,
count_href, count_img, count_form)
#       print "\n Addresses reached from %s are %s " % (root, doneLst)
        no_of_internal_links =  len(linkrefs)
#       print "\n %s contains %d internal addresses which are: %s " % (root,
len(linkrefs), linkrefs.keys())
        no_of_external_links = len(external_links)
        print "\n # external links: %d " % no_of_external_links
#       print "\n list of external links:", external_links          # <<<
        23.09.2009
#
        return msg_status, msg_reason, check_level, error_count
#
def read_Alexadb(nation, website):           #       called by "main"
#####################################################################
#                                                                   #
#  reads Alexa.db and appends url_names and nation to matchlist     #
#                                                                   #
#####################################################################
#
        matchlist = []
#
        try:
                con = dbapi.connect('alexa_tr.db', timeout=30)
                con.row_factory = dbapi.Row
                con.text_factory = str
                d = con.cursor()
#
                d.execute('CREATE TABLE IF NOT EXISTS AlexaRecs(CShort, WRank,
Name, CRank, LinksIn, PRIMARY KEY (CShort, WRank, Name))')
                con.text_factory = str
# start where it stops
                 if len(website) > 1 and len(nation) > 1:
                   x = 0
                   d.execute('SELECT * FROM AlexaRecs WHERE Name = "%s"' % website)
                   if row[2] == website :
                       print "\n row[2]:", row[2]

                   for row in d:
                       print "\n row:", row
                       if row[0] == nation and row[2] != website:
                           print "\n row[2]:", row[2]
                          matchlist.append(row[2])
                          x += 1
                       else:
                           print "\n matchlist in read_Alexadb >", matchlist
```

```
# start from beginning of the count list
                else:
                  if len(nation) > 1:
                     x = 0
                     d.execute('SELECT * FROM AlexaRecs WHERE CShort = "%s"' %
nation)

                     print "\n nation:", nation
                     for row in d:
                         print "\n row:", row
                         if row[0] == nation :
                             print "\n row[2]:", row[2]
                           matchlist.append(row[2])
                           x += 1
                         else:
                             print "\n matchlist in read_Alexadb >", matchlist


         except:
                print "999 Error Openning File"

        print "\n *1* "
#       print "\n matchlist in read_Alexadb >", matchlist
        return matchlist
```

```
#
######################################################################
#                                                                    #
#  main script                                                       #
#                                                                    #
#  - gets keywords from keybord to do google search and writes       #
#    information collected to the html file,                         #
#  - calls "parse" routine to query Url addresses in a list from     #
#    html file,                                                      #
#  - then calls "web_search" routine to query each Url and internal  #
#    addresses in it,                                                #
#  - and all keywords are collected.                                 #
#                                                                    #
######################################################################
#
# Table mapping response codes to messages; entries have the
# form {code: (shortmessage, longmessage)}.
responses = {
    100: ('Continue', 'Request received, please continue'),
    101: ('Switching Protocols',
          'Switching to new protocol; obey Upgrade header'),

    200: ('OK', 'Request fulfilled, document follows'),
    201: ('Created', 'Document created, URL follows'),
    202: ('Accepted',
          'Request accepted, processing continues off-line'),
    203: ('Non-Authoritative Information', 'Request fulfilled from cache'),
    204: ('No Content', 'Request fulfilled, nothing follows'),
    205: ('Reset Content', 'Clear input form for further input.'),
    206: ('Partial Content', 'Partial content follows.'),

    300: ('Multiple Choices',
          'Object has several resources -- see URI list'),
    301: ('Moved Permanently', 'Object moved permanently -- see URI list'),
    302: ('Found', 'Object moved temporarily -- see URI list'),
    303: ('See Other', 'Object moved -- see Method and URL list'),
    304: ('Not Modified',
          'Document has not changed since given time'),
    305: ('Use Proxy',
          'You must use proxy specified in Location to access this '
          'resource.'),
    307: ('Temporary Redirect',
          'Object moved temporarily -- see URI list'),

    400: ('Bad Request',
          'Bad request syntax or unsupported method'),
    401: ('Unauthorized',
          'No permission -- see authorization schemes'),
    402: ('Payment Required',
          'No payment -- see charging schemes'),
    403: ('Forbidden',
          'Request forbidden -- authorization will not help'),
    404: ('Not Found', 'Nothing matches the given URI'),
    405: ('Method Not Allowed',
          'Specified method is invalid for this server.'),
    406: ('Not Acceptable', 'URI not available in preferred format.'),
    407: ('Proxy Authentication Required', 'You must authenticate with '
          'this proxy before proceeding.'),
    408: ('Request Timeout', 'Request timed out; try again later.'),
    409: ('Conflict', 'Request conflict.'),
    410: ('Gone',
          'URI no longer exists and has been permanently removed.'),
    411: ('Length Required', 'Client must specify Content-Length.'),
    412: ('Precondition Failed', 'Precondition in headers is false.'),
    413: ('Request Entity Too Large', 'Entity is too large.'),
    414: ('Request-URI Too Long', 'URI is too long.'),
    415: ('Unsupported Media Type', 'Entity body in unsupported format.'),
    416: ('Requested Range Not Satisfiable',
          'Cannot satisfy request range.'),
    417: ('Expectation Failed',
          'Expect condition could not be satisfied.'),

    500: ('Internal Server Error', 'Server got itself in trouble'),
    501: ('Not Implemented',
          'Server does not support this operation'),
    502: ('Bad Gateway', 'Invalid responses from another server/proxy.'),
```

```
    503: ('Service Unavailable',
          'The server cannot process the request due to a high load'),
    504: ('Gateway Timeout',
          'The gateway server did not receive a timely response'),
    505: ('HTTP Version Not Supported', 'Cannot fulfill request.'),
    }
#
# doneLst = []     # <<<<<<< 14.09.2009
trword_lst =
[u'bir',u'olmak',u'iÃ§in',u'ben',u'demek',u'Ã§ok',u'yapmak',u'gibi',u'daha',u'alma
k',
u'var',u'kendi',u'gelmek',u'ile',u'vermek',u'ama',u'sonra',u'kadar',u'yer',u'insan
',
u'deÄŸil',u'her',u'istemek',u'yÄ±l',u'Ã§Ä±kmak',u'gÃ¶rmek',u'gÃ¼n',u'biz',u'gitmek
',u'iÅŸ',
u'Åžey',u'ara',u'bilmek',u'zaman',u'Ã§ocuk',u'iki',u'bakmak',u'Ã§alÄ±Åžmak',u'iÃ§i
nde',u'bÃ¼yÃ¼k',
u'yok',u'baÄŸlamak',u'yol',u'kalmak',u'neden',u'siz',u'konu',u'yapÄ±lmak',u'iyi',u
'kadÄ±n']
#
doneLst = []
resource = []
wordref_lst = {}
count_ref = {}
#charset ='ISO-8859-9'
charset_read = " "
url_processed = 0
#
print "\n os.name >>", os.name
#
print "\n hostname of the current machine>", socket.gethostname()
hostname_current = socket.gethostname()
#
print "\n IP address of given hostname>", socket.gethostbyname(hostname_current)
hostip_address = socket.gethostbyname(hostname_current)
#
print "\n ================================================="
#print "\n  Time: ", time.strftime('%c',time.localtime())
print "\n  Time: ", time.asctime(time.localtime())
print "\n ================================================="
#
nation = ""
resource = [""]
del resource[0]
print "\n Enter short code of country to start webcrawling from the beginning:"
read = sys.stdin.readline()
found = str(read).find("\n")
resourse1 = read[0:found]
resource.append(read[0:found])
print "\n short code of selected country:", resource
#
# nation = resourse1
nation = resource[0]
print "\n  short nation code:", nation
#
resource = [""]
del resource[0]
print "\n Enter website name only to start next the given website :"
read = sys.stdin.readline()
found = str(read).find("\n")
resourse1 = read[0:found]
resource.append(read[0:found])
print "\n Website name entered :", resource
website = resource[0]
print "\n  Website name entered:", website

if len(website) > 1:
   inFileTxt = 'keywords_inout.txt'
   s = open(inFileTxt,'r')
   line = s.readline()
#   l = s.readline()
#   line = l.decode(charset)
   while line:
      print "\n sed_input file read>",line   # variables to be replaced in the
file
      splitline = string.split(line)
```

```
      (charset_read,i,j) = splitline
      wordref_lst[i] = int(j)
      print "\n i >", i, " j >", j
#      print "\n i.decode(charset_read)>", j.decode(charset_read), "
j.decode(charset_read)>", j.decode(charset_read)
      line = s.readline()
   s.close()
#   print "\n wordref_lst stored >", wordref_lst
#while 1:
(year, month, day, hour, minute, second, weekday, dayofyear, dst) = localtime()
entery_year_keep = year
entery_month_keep = month
entery_day_keep = day
entery_hour_keep = hour
entery_minute_keep = minute
entery_second_keep = second
date_inserted =  str(year * 10000000000 + month * 100000000 + day * 1000000 + hour
* 10000+ minute * 100 + second)
mainexit_status = 0

maxdepth = 7
#maxduration = 600
maxduration = 60
max_depth = int(maxdepth)
max_duration = int(maxduration)
print "\n  max_depth (int)", max_depth, "max_duration (int)", max_duration


#
matchlist = read_Alexadb(nation, website)
print "\n *3* "

for groot in matchlist :
    groot = "http://www." + str(groot) + "/"
    if groot not in doneLst:
       print "\n groot in loop>", groot
       (entery_year, entery_month, entery_day, entery_hour, entery_minute,
entery_second) = timentry()
#       print "\n groot in loop>", groot
       error_count = 0                        # count # error occures for each Url
       (msg_status, msg_reason, check_level, error_count) = web_search(groot,
responses, error_count)
       count_found()
       timexit(entery_year, entery_month, entery_day, entery_hour, entery_minute,
entery_second, mainexit_status, nation)
       lastentry = entery_year * 10000000000 + entery_month * 100000000 +
entery_day * 1000000 + entery_hour * 10000 + entery_minute * 100 + entery_second
       date_inserted =  str(exit_year * 10000000000 + exit_month * 100000000 +
exit_day * 1000000 + exit_hour * 10000+ exit_minute * 100 + exit_second)
#
    url_processed = url_processed + 1
#    print "\n >>>>>>>>>>>>>  %s " % groot, " is added as the %d nd/rd/th" %
url_processed, "record of the database <<<<<<<<<<<<<"
    if charset != " ":
       g = open("keywords_inout.txt", 'w')           # <<<<<<< new
       for i, j in wordref_lst.iteritems():
#          print "\n i =", i, "    ", " j=", j
           gdata = charset+"   "+i+"   "+str(j)+"\n"
           g.write(gdata)                            # <<<<<<< new
#           g.write(gdata.encode(charset))
       g.close()
       k = open("websites_executed.txt", 'a')        # <<<<<<< new
       gdata = groot+"   "+charset_found+"\n"
       k.write(gdata)
#       k.write(gdata.encode(charset))               # <<<<<<< new
       k.close()

mainexit_status = 1
print "\n valu of date_inserted:", date_inserted
timexit(entery_year_keep, entery_month_keep, entery_day_keep, entery_hour_keep,
entery_minute_keep, entery_second_keep, mainexit_status, nation)
print " wordref_lst = ", wordref_lst
print "\n"
print "\n ###################################################################"
print " reps1 = ", reps1
print "\n ###################################################################"
```

```
print " reps2 = ", reps2
print "\n ########################################################"
print "\n"
for i, j in wordref_lst.iteritems():
   if not count_ref.has_key(j):
      count_ref[j] = [i]
   else:
      count_ref[j].append(i)
print " count_ref = ", count_ref
print "\n ########################################################"
k = open("keywords_counted.txt", 'w')         # <<<<<<< new
for i, j in count_ref.iteritems():
   print i,j
   gdata = "   "+str(i)+"   "+str(j)+"\n"
#
   k.write(gdata)
k.close()
```

APPENDIX F

RANKING OF COUNTRIES

Table 27: Ranking of Countries Based on Traffic Density Data from Alexa.com (no. of webs at first category)

RANK	Country Long	total no. of webs at category 1 - 1.000	Σ(1.000.000 - positions at category 1 - 1.000)	sum of links_in values at category 1 - 1.000	average of links_in at category 1 - 1.000	total no. of webs at category 1.001 - 10.000	Σ(1.000.000 - positions at category 1.001 - 10.000)	sum of links_in values at category 1.001 - 10.000	average of links_in at category 1.001 - 10.000	total no. of webs at category 10.001 - 100.000	Σ(1.000.000 - positions at category 10.001 - 100.000)	sum of links_in values at category 10.001 - 100.000	average of links_in at category 10.001 - 100.000	total no. of webs at category 100.001 - 1.000.000	Σ(1.000.000 - positions at category 100.001 - 1.000.000)	sum of links_in values at category 100.001 - 1.000.000
1	United States	431	430784910	14340410	33272	3360	3,342E+09	13570052	4038	27969	2,647E+10	18894623	675	196205	9,4278E+10	64648980
2	People's Republic of China	127	126939719	1353704	10659	847	842432403	1195827	1411	8014	7,568E+09	2579284	321	59450	3,0202E+10	4926975
3	Japan	61	60969995	693855	11374	468	465464648	782823	1672	3981	3,768E+09	1615487	405	33824	1,5709E+10	5704302
4	India	55	54967645	117183	2130	578	574874529	1012215	1751	7037	6,642E+09	2535108	360	60068	2,8643E+10	47868066
5	Germany	41	40977672	297137	7247	454	451495286	820898	1808	4290	4,053E+09	1657597	386	55045	2,4009E+10	5153816
6	France	33	32982042	432897	13118	264	262493316	477760	1809	2177	2,058E+09	1089193	500	15497	7250576974	2355998
7	Russia	29	28988692	415577	14330	320	318140208	405786	1268	3396	3,204E+09	986697	290	38399	1,7446E+10	7035578
8	Brazil	17	16994328	188111	11065	156	155052936	175186	1122	1674	1,583E+09	435615	260	13137	6046122677	5558182
9	United Kingdom	16	15992147	380930	23808	195	193931941	483021	2477	1890	1,784E+09	1212737	641	23008	9968736978	4217832
10	Italy	14	13993854	73503	5250	147	146222134	324375	2206	1590	1,502E+09	566061	356	18074	8087949119	6926925
11	Spain	13	12993605	152708	11746	180	179030596	377135	2095	1689	1,597E+09	931540	551	13552	6136535889	1864380
12	UnDefined	12	11992066	0	0	321	319108315	0	0	7014	6,618E+09	0	0	171332	6,6356E+10	0
13	Mexico	12	11995215	58437	4869	130	129266338	133131	1024	1230	1,164E+09	299205	243	8079	3873860247	1401898
14	**Turkey**	**11**	**10992909**	**28803**	**2618**	**105**	**1,04E+08**	**124582**	**1186**	**1223**	**1,15E+09**	**274363**	**224**	**12551**	**5,82E+09**	**1551600**
15	Poland	9	8996299	76159	8462	88	87488637	191464	2175	866	818891457	489323	565	9226	4013737040	1140882
16	Taiwan	9	8994231	47076	5230	59	58680893	54741	927	542	512370680	140514	259	3585	1697657187	1578687
17	Indonesia	8	7996518	111013	13876	74	73620005	108831	1470	905	853563976	251820	278	10457	4651270967	10549994
18	Iran	7	6996287	43096	6156	126	125266934	117390	931	1263	1,195E+09	224829	178	9155	4674374547	4747069
19	Saudi Arabia	7	6996883	22582	3226	120	119355594	128514	1070	1075	1,016E+09	259042	240	6277	3275831651	907602
20	Egypt	5	4996822	5426	1085	87	86560090	94537	1086	627	593890203	136400	217	3010	1529232076	2745119
21	Canada	5	4997371	16179	3235	55	54730522	198790	3614	484	457458939	318313	657	6925	2794170562	1940493
22	Australia	5	4997745	54273	10854	47	46728693	101322	2155	484	457253852	235617	486	7697	3185398314	1824947
23	South Korea	5	4997083	11362	2272	38	37778515	32667	859	390	368346281	63810	163	2410	1296994813	537763
24	Thailand	4	3998050	13335	3333	55	54671053	55792	1014	560	530197234	206886	369	4959	2280637794	5109831
25	Vietnam	4	3998730	12027	3006	53	52687411	39390	743	415	392773556	90097	217	2767	1316971331	180893
26	Netherlands	4	3998100	19798	4949	50	49721840	112127	2242	706	665131697	323771	458	10029	4286549137	1594333
27	Republic	4	3997150	16554	4138	26	25870614	139680	5372	287	271097871	304569	1061	3377	1416477222	512693
28	Argentina	4	3998233	60549	15137	19	18881650	31349	1649	309	292519927	98739	319	3496	1531138492	1542651
29	Hong Kong	3	2997625	4722	1574	24	23866011	52575	2190	206	195421073	499193	2423	1669	748499400	4409864
30	Sweden	2	1999025	10868	5434	30	29840652	71249	2374	384	362218258	142680	371	4732	2083484599	344451
31	Ukraine	2	1998967	3454	1727	23	22865607	51990	2260	324	305744764	116304	358	4270	1836766203	694558
32	Hungary	2	1999071	12501	6250	21	20891898	60215	2867	307	290350556	189981	618	2633	1162809545	337073
33	Algeria	2	1998149	1872	936	18	17892243	9914	550	120	113382965	26539	221	884	424461919	1357736
34	Austria	2	1998993	7200	3600	16	15900539	30096	1881	219	206821039	83391	380	3654	1566501311	631832
35	Portugal	2	1999188	5695	2847	8	7963149	13742	1717	162	153588139	47980	296	1757	759754439	129726
36	Philippines	2	1999396	25380	12690	8	7950794	10753	1344	103	97282270	41224	400	1615	657384060	1853122
37	Greece	1	999775	2456	2456	28	27844353	21816	779	348	328941076	91747	263	4079	1792813642	655573
38	Belgium	1	999801	5504	5504	21	20846359	21857	1040	166	157133605	91412	550	1970	797933916	144509
39	South Africa	1	999852	2035	2035	20	19886128	23532	1176	360	338611847	86010	238	4986	2197397076	1493647
40	Romania	1	999751	3829	3829	19	18872893	43122	2269	340	320728329	122081	359	4103	1782276066	291148
41	Norway	1	999536	2086	2086	18	17897768	29721	1651	210	198452581	79029	376	2484	1110219508	2304790
42	Malaysia	1	999747	2357	2357	17	16908442	27742	1631	165	155786179	67509	409	2198	940800585	600575
43	Switzerland	1	999745	5864	5864	16	15898624	37325	2332	137	129219812	77721	567	2708	1103335144	612784
44	Pakistan	1	999850	1450	1450	15	14908900	20145	1343	433	406377675	82997	191	5085	2478641673	13676139
45	Israel	1	999531	3029	3029	15	14916470	23420	1561	152	143174716	51625	339	1810	787320950	145956
46	Venezuela	1	999803	2198	2198	15	14911644	10490	699	136	128615860	41233	303	1122	510657867	63895
47	Finland	1	999582	2578	2578	14	13923758	20176	1441	206	193963380	75301	365	2250	1025440499	1458478
48	Peru	1	999716	2702	2702	14	13913764	13651	975	96	90609243	36385	379	1284	570235334	963638
49	Denmark	1	999609	2823	2823	11	10934956	16438	1494	190	179269493	48423	254	2883	1243335399	581178
50	Chile	1	999706	4152	4152	11	10946329	19157	1741	118	111926135	45667	387	1097	501557370	102865
51	Bulgaria	1	999189	2157	2157	11	10927725	40196	3654	105	99208176	358495	3414	1224	538108441	222850
52	Ireland	1	999595	2602	2602	10	9933869	21768	2176	98	92199640	52308	533	1561	668838015	125289
53	Slovakia	1	999278	2929	2929	9	8953340	9821	1091	121	114174358	43116	356	1195	506624478	121677
54	Croatia	1	999056	1771	1771	9	8950636	10862	1206	67	63105066	24708	368	814	365634377	83344
55	Colombia	1	999782	2176	2176	8	7953026	8018	1002	133	125955886	41891	314	1349	618929211	71863
56	Singapore	1	999576	2132	2132	7	6953637	12851	1835	113	107417484	37905	335	980	430088032	943202
57	Bangladesh	1	999575	891	891	5	4975030	1074	214	155	145493441	17470	112	1436	741442791	3089836
58	United Arab Emirates	1	999584	1846	1846	5	4966764	12695	2539	75	70902935	34084	454	601	260473041	35894
59	Sri Lanka	1	999125	954	954	5	4963224	1384	276	66	62098918	11326	171	755	349072710	2618094
60	Nigeria	1	999293	906	906	4	3973434	10365	2591	101	94747513	14041	139	608	363905965	459746

Table 27 Continued.

RANK	Country Long	total no. of webs at category 1 - 1.000	Σ(1.000.000 - positions at category 1 - 1.000)	sum of links_in values at category 1 - 1.000	average of links_in at category 1 - 1.000	total no. of webs at category 1.001 - 10.000	Σ(1.000.000 - positions at category 1.001 - 10.000)	sum of links_in values at category 1.001 - 10.000	average of links_in at category 1.001 - 10.000	total no. of webs at category 10.001 - 100.000	Σ(1.000.000 - positions at category 10.001 - 100.000)	sum of links_in values at category 10.001 - 100.000	average of links_in at category 10.001 - 100.000	total no. of webs at category 100.001 - 1.000.000	Σ(1.000.000 - positions at category 100.001 - 1.000.000)	sum of links_in values at category 100.001 - 1.000.000
61	New Zealand	1	999308	2392	2392	4	3983133	24024	6006	62	58609476	34482	556	1040	418610350	520846
62	Kuwait	1	999231	564	564	3	2980781	2305	768	67	63824786	13164	196	449	226003658	20966
63	Morocco	1	999273	2170	2170	3	2980859	1050	350	48	45311572	13749	286	641	287015291	49592
64	Tuvalu	1	999523	0	0	2	1989213	0	0	42	39609599	0	0	739	292885182	0
65	Libya	1	999121	879	879	2	1987382	569	284	31	29234586	2549	82	341	165803795	10418
66	Dominican Republic	1	999101	1297	1297	1	993658	891	891	38	35750389	8566	225	429	193103418	30418
67	Ecuador	1	999171	1340	1340	1	991635	1702	1702	29	27415642	17075	588	409	187876606	28153
68	Azerbaijan	0	0	0	0	10	9931784	6935	693	129	122125029	34206	265	698	367607318	24117
69	Kazakhstan	0	0	0	0	8	7946725	1666	208	113	105960362	13025	115	932	462922325	39740
70	United Kingdom	0	0	0	0	6	5965672	0	0	129	121778041	0	0	5386	1956335797	0
71	Latvia	0	0	0	0	6	5973344	4463	743	42	39233938	9723	231	579	246754444	36736
72	Ascension Island	0	0	0	0	5	4974656	6208	1241	82	77043822	14316	174	939	414051310	66561
73	Lithuania	0	0	0	0	5	4978996	5554	1110	71	66857457	24880	350	932	403602329	74638
74	Slovenia	0	0	0	0	5	4968756	7442	1488	49	46302651	17822	363	704	293837774	46542
75	Belarus	0	0	0	0	4	3986258	2499	624	75	70772946	17035	227	1175	554821334	62861
76	Estonia	0	0	0	0	4	3971786	3827	956	40	37750997	16937	423	414	171970049	28283
77	Qatar	0	0	0	0	3	2980714	3090	1030	31	29176934	5281	170	224	108680546	875010
78	Lebanon	0	0	0	0	3	2983071	788	262	24	22784153	8909	371	160	77566479	11306
79	Puerto Rico	0	0	0	0	3	2983936	2055	685	20	18993181	4415	220	147	66016622	9390
80	East Timor	0	0	0	0	3	2982754	0	0	6	5758865	0	0	21	6939061	0
81	Yemen	0	0	0	0	2	1981843	1064	532	29	27172330	3013	103	305	159245391	14259
82	Costa Rica	0	0	0	0	2	1991972	3958	1979	28	26387194	5603	200	296	123097939	454928
83	European Union	0	0	0	0	2	1981915	0	0	27	25565634	0	0	1113	422938525	0
84	Oman	0	0	0	0	2	1989974	1027	513	26	24398136	2979	114	235	110477581	9410
85	Samoa (Western Samoa	0	0	0	0	2	1982487	0	0	13	12287636	0	0	266	112711242	795
86	Uruguay	0	0	0	0	2	1989066	1799	899	8	7672132	3587	448	231	96311111	26388
87	Ghana	0	0	0	0	2	1987641	1830	915	8	7567143	1773	221	121	61559703	436882
88	Montenegro	0	0	0	0	2	1989147	0	0	8	7558333	0	0	228	87485863	0
89	Syria	0	0	0	0	1	994209	220	220	58	54603818	5477	94	434	230144356	14967
90	Jordan	0	0	0	0	1	997401	1021	1021	24	22635483	5602	233	192	84553510	9506
91	Nepal	0	0	0	0	1	996615	830	830	22	20749668	2967	134	284	124372856	442380
92	Kenya	0	0	0	0	1	998464	905	905	22	20499780	6278	285	359	172769444	18429
93	Bosnia and Herzegovina	0	0	0	0	1	997197	2777	2777	22	20899775	10045	456	294	127434863	16248
94	Luxembourg	0	0	0	0	1	996536	1260	1260	19	17828454	3340	175	241	99759500	13876
95	Georgia	0	0	0	0	1	990679	846	846	16	15054212	1783	111	232	105784610	7732
96	Cuba	0	0	0	0	1	996271	985	985	16	14981742	4027	251	130	69109688	12107
97	Guatemala	0	0	0	0	1	998574	1145	1145	13	12317323	3457	265	185	80110934	19211
98	Cameroon	0	0	0	0	1	993071	56	56	12	11232653	3363	280	83	44003760	436259
99	Paraguay	0	0	0	0	1	996658	958	958	10	9408221	3355	335	107	45245672	444186
100	Bolivia	0	0	0	0	1	998013	1205	1205	10	9394441	3529	352	167	82538938	452316
101	El Salvador	0	0	0	0	1	998129	1075	1075	10	9407709	2914	291	120	49790552	13199
102	Cambodia	0	0	0	0	1	991312	692	692	10	9361501	1575	157	174	90030780	870163
103	Madagascar	0	0	0	0	1	993723	301	301	9	8436361	2138	237	141	56746219	20648
104	Honduras	0	0	0	0	1	995886	941	941	8	7494075	3230	403	44	18793513	2501
105	Armenia	0	0	0	0	1	990471	815	815	8	7349026	1065	133	247	110372233	10570
106	Nicaragua	0	0	0	0	1	995864	942	942	7	6568916	2384	340	99	45052439	8302
107	Senegal	0	0	0	0	1	991673	792	792	7	6571462	1048	149	74	32786334	5545
108	Trinidad and Tobago	0	0	0	0	1	994865	884	884	5	4749422	1866	373	66	32659196	3105
109	Soviet Union	0	0	0	0	1	993975	0	0	5	4710277	0	0	148	59131700	0
110	Mauritius	0	0	0	0	1	991381	907	907	4	3737747	1261	315	49	19236427	2272
111	Bahrain	0	0	0	0	1	996882	971	971	4	3856336	2098	524	98	45591343	5402
112	Brunei	0	0	0	0	1	991672	742	742	3	2825912	773	257	49	22477724	434696
113	Cote d Ivoire	0	0	0	0	1	994532	0	0	2	1878393	0	0	21	9580217	0
114	Jamaica	0	0	0	0	1	991931	860	860	0	0	0	0	53	21923848	2608
115	Iraq	0	0	0	0	0	0	0	0	73	68805700	9024	123	410	232992285	17439
116	Cocos (Keeing Islands)	0	0	0	0	0	0	0	0	28	26513340	0	0	964	358899964	0
117	Sudan	0	0	0	0	0	0	0	0	25	23640325	1874	74	194	119421222	870509
118	Tunisia	0	0	0	0	0	0	0	0	21	19684588	3778	179	234	113572021	27327
119	Repubic of Macedonia	0	0	0	0	0	0	0	0	18	17053912	3416	189	189	83751144	9395

Table 28: Ranking of Countries Based on Traffic Density Data from Alexa.com (for all categories)

RANK	Country Long	**Traffic density for all categories: 5 points for 1-1.000, 4 points for 1.001-10.000, 3 points for 10.001-100.000, 2 points for 100.001-1.000.000**	total no. of webs at category 1 - 1.000	total no. of webs at category 1.001 - 10.000	total no. of webs at category 10.001 - 100000	total no. of webs at category 100.001 - 1.000.000
1	United States	491912	431	3360	27969	196205
2	UnDefined	365050	12	321	7014	171332
3	People's Republic of China	146965	127	847	8014	59450
4	India	143834	55	578	7037	60068
5	Germany	124981	41	454	4290	55045
6	Russia	88411	29	320	3396	38399
7	Japan	81768	61	468	3981	33824
8	United Kingdom	52546	16	195	1890	23008
9	Italy	41576	14	147	1590	18074
10	France	38746	33	264	2177	15497
11	Spain	32956	13	180	1689	13552
12	Brazil	32005	17	156	1674	13137
13	**Turkey**	**29246**	**11**	**105**	**1223**	**12551**
14	Indonesia	23965	8	74	905	10457
15	Iran	22638	7	126	1263	9155
16	Netherlands	22396	4	50	706	10029
17	Poland	21447	9	88	866	9226
18	Mexico	20428	12	130	1230	8079
19	Australia	17059	5	47	484	7697
20	Saudi Arabia	16294	7	120	1075	6277
21	Canada	15547	5	55	484	6925
22	Thailand	11838	4	55	560	4959
23	Pakistan	11534	1	15	433	5085
24	United Kingdom	11183	0	6	129	5386
25	South Africa	11137	1	20	360	4986
26	Sweden	10746	2	30	384	4732
27	Ukraine	9614	2	23	324	4270
28	Greece	9319	1	28	348	4079
29	Romania	9307	1	19	340	4103
30	Taiwan	9077	9	59	542	3585
31	Egypt	8274	5	87	627	3010
32	Austria	8039	2	16	219	3654
33	Argentina	8015	4	19	309	3496
34	Czech Repubic	7739	4	26	287	3377
35	Vietnam	7011	4	53	415	2767
36	Denmark	6385	1	11	190	2883
37	Hungary	6281	2	21	307	2633
38	South Korea	6167	5	38	390	2410
39	Switzerland	5896	1	16	137	2708
40	Norway	5675	1	18	210	2484
41	Finland	5179	1	14	206	2250
42	Malaysia	4964	1	17	165	2198

Table 29: Ranking of countries based on traffic density data from Alexa.com (sum of links in)

RANK	Country Long	total no. of webs at category 1 - 1.000	Σ(1.000.000 - positions at category 1 - 1.000)	**sum of links_in values at category 1 - 1.000**	average of links_in at category 1 - 1.000	total no. of webs at category 1.001 - 10.000	Σ(1.000.000 - positions at category 1.001 - 10.000)	sum of links_in values at category 1.001 - 10.000	average of links_in at category 1.001 - 10.000	total no. of webs at category 10.001 - 100.000	Σ(1.000.000 - positions at category 10.001 - 100.000)	sum of links_in values at category 10.001 - 100.000	average of links_in at category 10.001 - 100.000	total no. of webs at category 100.001 - 1.000.000	Σ(1.000.000 - positions at category 100.001 - 1.000.000)	sum of links_in values at category 100.001 - 1.000.000
1	United States	431	4,31E+08	14340410	33272	3360	3,34E+09	13570052	4038	27969	2,6466E+10	18894623	675	196205	9,4278E+10	64648980
2	People's Republic of China	127	1,27E+08	1353704	10659	847	8,42E+08	1195827	1411	8014	7568441538	2579284	321	59450	3,0202E+10	4926975
3	Japan	61	60969995	693855	11374	468	4,65E+08	782823	1672	3981	3767655637	1615487	405	33824	1,5709E+10	5704302
4	France	33	32982042	432897	13118	264	2,62E+08	477760	1809	2177	2057778451	1089193	500	15497	7250576974	2355998
5	Russia	29	28988692	415577	14330	320	3,18E+08	405786	1268	3396	3204111877	986697	290	38399	1,7446E+10	7035578
6	United Kingdom	16	15992147	380930	23808	195	1,94E+08	483021	2477	1890	1784150515	1212737	641	23008	9968736978	4217832
7	Germany	41	40977672	297137	7247	454	4,51E+08	820898	1808	4290	4052718761	1657597	386	55045	2,4009E+10	5153816
8	Brazil	17	16994328	188111	11065	156	1,55E+08	175186	1122	1674	1582708737	435615	260	13137	6046122677	5558182
9	Spain	13	12993605	152708	11746	180	1,79E+08	377135	2095	1689	1597409217	931540	551	13552	6136535889	1864380
10	India	55	54967645	117183	2130	578	5,75E+08	1012215	1751	7037	6642390095	2535108	360	60068	2,8643E+10	47868066
11	Indonesia	8	7996518	111013	13876	74	73620005	108831	1470	905	853563976	251820	278	10457	4651270967	10549994
12	Poland	9	8996299	76159	8462	88	87488637	191464	2175	866	818891457	489323	565	9226	4013737040	1140882
13	Italy	14	13993854	73503	5250	147	1,46E+08	324375	2206	1590	1501701900	566061	356	18074	8087949119	6926925
14	Argentina	4	3998233	60549	15137	19	18881650	31349	1649	309	292519927	98739	319	3496	1531138492	1542651
15	Mexico	12	11995215	58437	4869	130	1,29E+08	133131	1024	1230	1163567623	299205	243	8079	3873860247	1401898
16	Australia	5	4997745	54273	10854	47	46728693	101322	2155	484	457253852	235617	486	7697	3185398314	1824947
17	Taiwan	9	8994231	47076	5230	59	58680893	54741	927	542	512370680	140514	259	3585	1697657187	1578687
18	Iran	7	6996287	43096	6156	126	1,25E+08	117390	931	1263	1194914952	224829	178	9155	4674374547	4747069
19	**Turkey**	**11**	**1,1E+07**	**28803**	**2618**	**105**	**1E+08**	**124582**	**1186**	**1223**	**1,15E+09**	**274363**	**224**	**12551**	**5,823E+09**	**1551600**
20	Philippines	2	1999396	25380	12690	8	7950794	10753	1344	103	97282270	41224	400	1615	657384060	1853122
21	Saudi Arabia	7	6996883	22582	3226	120	1,19E+08	128514	1070	1075	1015804920	259042	240	6277	3275831651	907602
22	Netherlands	4	3998100	19798	4949	50	49721840	112127	2242	706	665131697	323771	458	10029	4286549137	1594333
23	Czech Republic	4	3997150	16554	4138	26	25870614	139680	5372	287	271097871	304569	1061	3377	1416477222	512693
24	Canada	5	4997371	16179	3235	55	54730522	198790	3614	484	457458939	318313	657	6925	2794170562	1940493
25	Thailand	4	3998050	13335	3333	55	54671053	55792	1014	560	530197234	206886	369	4959	2280637794	5109831
26	Hungary	2	1999071	12501	6250	21	20891898	60215	2867	307	290350556	189981	618	2633	1162809545	337073
27	Vietnam	4	3998730	12027	3006	53	52687411	39390	743	415	392773556	90097	217	2767	1316971331	180893
28	South Korea	5	4997083	11362	2272	38	37778515	32667	859	390	368346281	63810	163	2410	1296994813	537763
29	Sweden	2	1999025	10868	5434	30	29840652	71249	2374	384	362218258	142680	371	4732	2083484599	344451
30	Austria	2	1998993	7200	3600	16	15900539	30096	1881	219	206821039	83391	380	3654	1566501311	631832
31	Switzerland	1	999745	5864	5864	16	15898624	37325	2332	137	129219812	77721	567	2708	1103335144	612784
32	Portugal	2	1999188	5695	2847	8	7963149	13742	1717	162	153588139	47980	296	1757	759754439	129726
33	Belgium	1	999801	5504	5504	21	20846359	21857	1040	166	157133605	91412	550	1970	797933916	144509
34	Egypt	5	4996822	5426	1085	87	86560090	94537	1086	627	593890203	136400	217	3010	1529232076	2745119
35	Hong Kong	3	2997625	4722	1574	24	23866011	52575	2190	206	195421073	499193	2423	1669	748499400	4409864
36	Chile	1	999706	4152	4152	11	10946329	19157	1741	118	111926135	45667	387	1097	501557370	102865
37	Romania	1	999751	3829	3829	19	18872893	43122	2269	340	320728329	122081	359	4103	1782276066	291148
38	Ukraine	2	1998967	3454	1727	23	22865607	51990	2260	324	305744764	116304	358	4270	1836766203	694558
39	Israel	1	999531	3029	3029	15	14916470	23420	1561	152	143174716	51625	339	1810	787320950	145956
40	Sovakia	1	999278	2929	2929	9	8953340	9821	1091	121	114174358	43116	356	1195	506624478	121677

Table 29 Continued.

RANK	Country Long	total no. of webs at category 1 - 1.000	Σ(1.000.000 - positions at category 1 - 1.000)	**sum of links_in values at category 1 - 1.000**	average of links_in at category 1 - 1.000	total no. of webs at category 1.001 - 10.000	Σ(1.000.000 - positions at category 1.001 - 10.000)	sum of links_in values at category 1.001 - 10.000	average of links_in at category 1.001 - 10.000	total no. of webs at category 10.001 - 100.000	Σ(1.000.000 - positions at category 10.001 - 100.000)	sum of links_in values at category 10.001 - 100.000	average of links_in at category 10.001 - 100.000	total no. of webs at category 100.001 - 1.000.000	Σ(1.000.000 - positions at category 100.001 - 1.000.000)	sum of links_in values at category 100.001 - 1.000.000
41	Denmark	1	999609	2823	2823	11	10934956	16438	1494	190	179269493	48423	254	2883	1243335399	581178
42	Peru	1	999716	2702	2702	14	13913764	13651	975	96	90609243	36385	379	1284	570235334	963638
43	Ireland	1	999595	2602	2602	10	9933869	21768	2176	98	92199640	52308	533	1561	668838015	125289
44	Finland	1	999582	2578	2578	14	13923758	20176	1441	206	193963380	75301	365	2250	1025440499	1458478
45	Greece	1	999775	2456	2456	28	27844353	21816	779	348	328941076	91747	263	4079	1792813642	655573
46	New Zealand	1	999308	2392	2392	4	3983133	24024	6006	62	58609476	34482	556	1040	418610350	520846
47	Malaysia	1	999747	2357	2357	17	16908442	27742	1631	165	155786179	67509	409	2198	940800585	600575
48	Venezuella	1	999803	2198	2198	15	14911644	10490	699	136	128615860	41233	303	1122	510657867	63895
49	Colombia	1	999782	2176	2176	8	7953026	8018	1002	133	125955886	41891	314	1349	618929211	71863
50	Morocco	1	999273	2170	2170	3	2980859	1050	350	48	45311572	13749	286	641	287015291	49592
51	Bulgaria	1	999189	2157	2157	11	10927725	40196	3654	105	99208176	358495	3414	1224	538108441	222850
52	Singapore	1	999576	2132	2132	7	6953637	12851	1835	113	107417484	37905	335	980	430088032	943202
53	Norway	1	999536	2086	2086	18	17897768	29721	1651	210	198452581	79029	376	2484	1110219508	2304790
54	South Africa	1	999852	2035	2035	20	19886128	23532	1176	360	338611847	86010	238	4986	2197397076	1493647
55	Algeria	2	1998149	1872	936	18	17892243	9914	550	120	113382965	26539	221	884	424461919	1357736
56	United Arab Emir	1	999584	1846	1846	5	4966764	12695	2539	75	70902935	34084	454	601	260473041	35894
57	Croatia	1	999056	1771	1771	9	8950636	10862	1206	67	63105066	24708	368	814	365634377	83344
58	Pakistan	1	999850	1450	1450	15	14908900	20145	1343	433	406377675	82997	191	5085	2478641673	13676139
59	Ecuador	1	999171	1340	1340	1	991635	1702	1702	29	27415642	17075	588	409	187876606	28153
60	Dominican Repul	1	999101	1297	1297	1	993658	891	891	38	35750389	8566	225	429	193103418	30418
61	Sri Lanka	1	999125	954	954	5	4963224	1384	276	66	62098918	11326	171	755	349072710	2618094
62	Nigeria	1	999293	906	906	4	3973434	10365	2591	101	94747513	14041	139	608	363905965	459746
63	Bangladesh	1	999575	891	891	5	4975030	1074	214	155	145493441	17470	112	1436	741442791	3089836
64	Libya	1	999121	879	879	2	1987382	569	284	31	29234586	2549	82	341	165803795	10418
65	Kuwait	1	999231	564	564	3	2980781	2305	768	67	63824786	13164	196	449	226003658	20966
66	UnDefined	12	11992066	0	0	321	3,19E+08	0	0	7014	6617668914	0	0	171332	6,6356E+10	0
67	Tuvalu	1	999523	0	0	2	1989213	0	0	42	39609599	0	0	739	292885182	0

Ranking of Countries' Higher Education Institutions in the Top 500 of the Ranking Web of Webometrics.com is shown in Table 30 and Table 31.

Table 30: Strength Ranking

QS SAFE - National System Strength Rankings

Rank	Country	System		Access		Flagship		Economic		Overall Score
		Score	Rank	Score	Rank	Score	Rank	Score	Rank	
1	United States	100	1	100	1	100	1	100	1	100
2	United Kingdom	98	2	94	4	100	2	98	3	98
3	Australia	92	5	97	2	99	3	88	7	94
4	Germany	95	3	87	7	95	15	93	5	92
5	Canada	92	4	93	5	98	5	86	9	92
6	Japan	91	6	80	13	98	4	89	6	90
7	France	87	8	89	6	98	8	82	11	89
8	Netherlands	91	7	83	9	95	14	76	13	86
9	Korea, South	71	16	81	12	96	12	71	16	79
10	Sweden	75	13	82	11	94	16	64	19	79
11	Switzerland	83	9	71	20	98	6	62	21	79
12	Italy	73	14	96	3	76	30	64	18	77
13	Belgium	75	12	77	16	93	18	63	20	77
14	New Zealand	66	18	82	10	94	17	62	22	76
15	China	78	11	32	36	96	12	96	4	75
16	Hong Kong	80	10	61	24	98	7	49	30	72
17	Ireland	65	19	79	15	96	11	45	33	71
18	Finland	62	22	75	17	91	19	50	28	70
19	Taiwan	58	24	74	18	87	22	47	31	66
20	Austria	54	25	83	8	88	21	41	35	66
21	Denmark	66	17	59	27	96	10	45	34	66
22	Thailand	47	28	60	26	81	25	77	12	66
23	Israel	65	20	52	30	91	20	54	24	65
24	India	63	21	17	44	82	24	99	2	65
25	Singapore	73	15	61	23	97	9	22	40	63
26	Malaysia	51	26	60	25	68	34	66	17	61
27	Brazil	46	29	31	37	75	31	75	14	57
28	Spain	61	23	35	35	77	29	51	26	56
29	Greece	36	34	79	14	74	33	26	38	54
30	South Africa	34	36	44	32	78	26	59	23	54
31	Norway	47	27	65	22	78	26	23	39	53
32	Indonesia	35	35	38	34	54	38	84	10	53
33	Philippines	38	33	18	41	62	36	88	8	52
34	Mexico	31	38	45	31	83	23	45	32	51
35	Russia	42	30	22	40	77	28	54	25	49
36	Czech Republic	21	41	74	19	61	37	20	42	44
37	Poland	31	37	54	29	50	39	38	37	43
38	Chile	27	39	40	33	66	35	40	36	43
39	Argentina	40	31	2	48	75	32	49	29	41
40	**Turkey**	**39**	**32**	**26**	**39**	**26**	**41**	**50**	**27**	**35**

Table 30 Continued.

The Country Scoreboard has been designed following the model of the QS SAFE National System for evaluation of the countries' higher education system according to the presence of their Universities in the Top 500 of the Ranking Web.
Four normalized indicators are used with equal weighting as follows:

System: Number of universities in the Top 500 in the given country, divided by the mean position of those institutions.
Access: A score built according to ranks (5 points for a university in the top 100, 4 points for 101-200, 3 points for 201-300, 2 for 301-400 and 1 for 401-500) divided by the population size (root of the population in thousands) of the country (World Bank, 2007).
Flagship: A normalized score (100 for positions 1-20, 96 for 21-40, and so on) based on the leading university rank for countries with institutions among the Top 500.
Economic: Same score as the access defined before but divided by the GDP (PPP) per capita for the country in question (World Bank, 2007).

Table 31: Country Scoreboard

QS SAFE - National System Country Scoreboard						
RANK	**COUNTRY**	**SYSTEM**	**ACCESS**	**FLAGSHIP**	**ECONOMIC**	**OVERALL**
1	United States	100	100	100	100	**100**
2	Germany	82	94	80	91	**87**
3	United Kingdom	74	91	96	82	**86**
4	Canada	72	96	96	74	**85**
5	Spain	59	73	72	67	**68**
6	Australia	51	71	88	51	**65**
7	Sweden	47	77	90	46	**65**
8	Netherlands	50	73	88	47	**64**
9	Switzerland	45	72	92	42	**63**
10	Brazil	47	31	92	76	**62**
11	Japan	51	42	96	54	**61**
11	Taiwan	48	58	88	49	**61**
13	Italy	51	50	84	55	**60**
14	Norway	43	63	92	36	**59**
15	Finland	42	59	88	39	**57**
16	Belgium	44	57	84	41	**56**
17	Austria	42	53	88	38	**55**
18	Israel	42	52	76	41	**53**
19	Denmark	42	61	68	39	**52**
20	France	45	37	80	44	**51**
20	Czech Republic	41	43	80	40	**51**
20	Hong Kong	42	48	76	36	**51**
23	China	41	23	68	63	**49**
23	Mexico	39	24	92	39	**49**
25	Portugal	41	41	60	40	**46**
26	Slovenia	38	37	72	34	**45**
27	Thailand	40	28	56	53	**44**
27	Singapore	38	34	72	33	**44**
27	Ireland	40	40	60	34	**44**
30	South Korea	40	28	68	37	**43**
31	Saudi Arabia	39	28	64	36	**42**
32	Chile	38	26	64	36	**41**
33	Hungary	39	34	48	38	**40**
34	Estonia	38	36	44	34	**38**
34	Russia	38	23	56	35	**38**
36	New Zealand	38	32	44	34	**37**
37	Argentina	38	24	48	35	**36**
37	Iceland	37	42	32	32	**36**
39	Greece	39	28	40	34	**35**
39	Poland	39	26	36	37	**35**
41	South Africa	37	23	40	35	**34**
42	Serbia	37	25	24	36	**30**
43	Slovakia	37	23	16	32	**27**
44	India	37	21	8	37	**26**
45	Turkey	37	22	4	33	**24**

The Country Scoreboard has been designed following the model of the QS SAFE National System for evaluation of the countries' higher education system according to the presence of their Universities in the Top 500 of the Ranking Web.

The QS SAFE National System Strength evaluation is the first attempt to use rankings results, in concert with other indicators, not to evaluate the relative strength of individual institutions but of countries' higher education system

Table 31 Continued.

strengths as a whole.

QS has identified four key indicators which have been combined with equal weighting using standard statistical methods (the same standardisation approach used for the main QS World University Rankings™).

Four normalized indicators are used with equal weighting as follows:

System: Number of universities in the Top 500 in the given country, divided by the mean position of those institutions.
Access: A score built according to ranks (5 points for a university in the top 100, 4 points for 101-200, 3 points for 201-300, 2 for 301-400 and 1 for 401-500) divided by the population size (root of the population in thousands) of the country (World Bank, 2007).
Flagship: A normalized score (100 for positions 1-20, 96 for 21-40, and so on) based on the leading university rank for countries with institutions among the Top 500.

Economic: Same score as the access defined before but divided by the GDP (PPP) per capita for the country in question (World Bank, 2007).

Table 32: Ranking of Countries Based on Number of National Domains[23]

Webometrics - Number of National Domains		
RANK	**COUNTRY**	**NUMBER OF NATIONAL DOMAINS**
1	Germany (de)	80,100,000
2	United Kingdom (uk)	31,900,000
3	Japan (jp)	23,400,000
4	Netherlands (nl)	17,400,000
5	China (cn)	15,000,000
6	Russian Federation (ru)	14,900,000
7	Poland (pl)	12,300,000
8	Italy (it)	12,200,000
9	France (fr)	8,170,000
10	Brazil (br)	7,920,000
11	Australia (au)	7,680,000
12	Switzerland (ch)	7,230,000
13	Denmark (dk)	6,740,000
14	Canada (ca)	6,470,000
15	Czech Republic (cz)	6,230,000
16	Republic Of Korea (kr)	6,060,000
17	United States of America (us)	5,870,000
18	Austria (at)	5,190,000
19	Belgium (be)	4,970,000
20	Sweden (se)	4,670,000
21	European Union (eu)	3,890,000
22	Argentina (ar)	3,670,000
23	Hungary (hu)	3,140,000
24	Norway (no)	2,560,000
25	South Africa (za)	2,530,000
26	Taiwan (tw)	2,510,000
27	Spain (es)	2,470,000
28	New Zealand (nz)	2,250,000
29	Finland (fi)	1,800,000
30	Romania (ro)	1,780,000
31	Ukraine (ua)	1,760,000
32	Slovakia (sk)	1,410,000
33	Greece (gr)	1,320,000
34	Tuvalu (tv)	1,240,000
35	Mexico (mx)	1,160,000
36	**Turkey (tr)**	**1,160,000**
37	India (in)	1,070,000
38	Chile (cl)	1,010,000
39	Cocos (Keeling Islands) (cc)	1,000,000
40	Niue (nu)	917000
41	Israel (il)	916000
42	Ireland (ie)	780000

Table 33: General Features of Turkish Websites Derived from Three Different Search Algorithms

	result of inquiry with 19 keywords selected intuitively	result of inquiry with 50 keywords from Dr. İlyas Göz research	inquiry result of websites selected from Alexa.com search
count of repeated records which differs than the previous record	40.394	8422	25
Average caching period (in seconds)	1.786	1614	828
Average caching period (in minutes)	30	27	13,8
count of all URLs having size > 0	99.217	18975	12780
count of URLs having size = 0	648	56	682
total no. of repeated records having size > 0	65.367	9357	40
observation of same URL if it exists at least twice (unique count for each repeating URL)	16.109	3377	40
Average URL size	5.901.021	12.494.392	6.031.198
Average URL width	36	70	60
Average internal links	97	181	122
Average external links	162	482	92
Average HREF link info	6.590	13469	9191
Average IMG link	217	622	230
Average FORM link	2	5	0
Average search duration (in seconds)	96	247	164
Ratio of all URLs having size > 0 to all repeated records having size >0	1,52	2,03	319,50
Ratio of all repeated URLs having size > 0 to tot. no. of unique repeated records	4,06	2,77	1,00

Table 34: General Features of Turkish Websites Having com.tr Extension Derived from Three Different Search Algorithms

	result of inquiry with 19 keywords selected intuitively	result of inquiry with 50 keywords from Dr. İlyas Göz research	inquiry result of websites selected from Alexa search
count of repeated records which differs than the previous record	27.807	6664	20
Average caching period (in seconds)	1.787	1644	746
Average caching period (in minutes)	30	27	12
count of all URLs having size > 0	66.661	13913	8673
count of URLs having size = 0	219	26	352
total no. of repeated records having size > 0	43.142	7254	32
observation of same URL if it exists at least twice (unique count for each repeating URL)	10.683	2433	32
Average URL size	5.248.407	11.763.920	5.441.718
Average URL width	38	71	54
Average internal links	98	178	112
Average external links	186	547	88
Average HREF link info	6.946	13885	8487
Average IMG link	240	679	224
Average FORM link	3	6	1
Average search duration (in seconds)	95	251	150
Ratio of all URLs having size > 0 and all repeated records having size >0	1,55	1,92	271,03
Ratio of all repeated URLs having size > 0 and tot. no. of unique repeated records	4,04	2,98	1,00

Table 35: General Features of Turkish Websites with edu.tr Extension Derived from Three Different Search Algorithms

	result of inquiry with 19 keywords selected intuitively	result of inquiry with 50 keywords from Dr. İlyas Göz research	inquiry result of websites selected from Alexa search
count of repeated records which differs than the previous record	709	100	0
Average caching period (in seconds)	1.795	1086	0
Average caching period (in minutes)	30	18	0
count of all URLs having size > 0	3.539	526	109
count of URLs having size = 0	57	2	2
total no. of repeated records having size > 0	2.479	162	1
observation of same URL if it exists at least twice (unique count for each repeating URL)	608	99	0
Average URL size	7.113.806	18.950.240	7.584.441
Average URL width	14	26	23
Average internal links	32	98	19
Average external links	68	262	36
Average HREF link info	504	2302	391
Average IMG link	111	293	86
Average FORM link	0	1	0
Average search duration (in seconds)	49	186	30
Ratio of all URLs having size > 0 and all repeated records having size >0	1,43	3,25	109,00
Ratio of all repeated URLs having size > 0 and tot. no. of unique repeated records	4,08	1,64	

Table 36: General Features of Turkish Websites Having gov.tr Extension Derived from Three Different Search Algorithms

	result of inquiry with 19 keywords selected intuitively	result of inquiry with 50 keywords from Dr. İlyas Göz research	inquiry result of websites selected from Alexa search
count of repeated records which differs than the previous record	1.104	92	1
Average caching period (in seconds)	1.803	1252	2756
Average caching period (in minutes)	30	21	46
count of all URLs having size > 0	3.976	340	173
count of URLs having size = 0	85	9	11
total no. of repeated records having size > 0	2.739	136	1
observation of same URL if it exists at least twice (unique count for each repeating URL)	632	74	1
Average URL size	7.738.267	16.473.851	7.658.792
Average URL width	13	25	14
Average internal links	14	67	15
Average external links	19	170	20
Average HREF link info	548	2166	673
Average IMG link	43	173	45
Average FORM link	0	1	2
Average search duration (in seconds)	26	114	66
Ratio of all URLs having size > 0 and all repeated records having size >0	1,45	2,50	173,00
Ratio of all repeated URLs having size > 0 and tot. no. of unique repeated records	4,33	1,84	1,00

Table 37: General Features of Turkish Websites with net.tr Extension Derived from Three Different Search Algorithms

	result of inquiry with 19 keywords selected intuitively	result of inquiry with 50 keywords from Dr. İlyas Göz research	inquiry result of websites selected from Alexa search
count of repeated records which differs than the previous record	4.871	752	4
Average caching period (in seconds)	1.781	1669	1301
Average caching period (in minutes)	30	28	22
count of all URLs having size > 0	9.764	1708	1922
count of URLs having size = 0	5	0	142
total no. of repeated records having size > 0	6.703	802	5
observation of same URL if it exists at least twice (unique count for each repeating URL)	1.528	303	5
Average URL size	7.529.096	13731743	7549384
Average URL width	55	99	82
Average internal links	174	271	167
Average external links	102	302	106
Average HREF link info	10.833	18747	12579
Average IMG link	275	672	272
Average FORM link	0	0	0
Average search duration (in seconds)	153	297	218
Ratio of all URLs having size > 0 and all repeated records having size >0	1,46	2,13	384,40
Ratio of all repeated URLs having size > 0 and tot. no. of unique repeated records	4,39	2,65	1,00

Table 38: General Features of Turkish Websites Having org.tr Extension Derived from Three Different Search Algorithms

	result of inquiry with 19 keywords selected intuitively	result of inquiry with 50 keywords from Dr. İlyas Göz research	inquiry result of websites selected from Alexa search
count of repeated records which differs than the previous record	3.425	565	0
Average caching period (in seconds)	1.784	1487	0
Average caching period (in minutes)	30	25	0
count of all URLs having size > 0	9.135	1624	952
count of URLs having size = 0	92	1	66
total no. of repeated records having size > 0	6.291	694	1
observation of same URL if it exists at least twice (unique count for each repeating URL)	1.555	279	1
Average URL size	8.600.930	13.472.979	7.347.372
Average URL width	28	57	71
Average internal links	82	175	144
Average external links	202	429	124
Average HREF link info	5.902	11830	9994
Average IMG link	186	410	247
Average FORM link	0	0	0
Average search duration (in seconds)	97	227	199
Ratio of all URLs having size > 0 and all repeated records having size >0	1,45	2,34	952,00
Ratio of all repeated URLs having size > 0 and tot. no. of unique repeated records	4,05	2,49	1,00

Table 39: General Features of Turkish Websites With info.tr Extension Derived from Three Different Search Algorithms

	result of inquiry with 19 keywords selected intuitively	result of inquiry with 50 keywords from Dr. İlyas Göz research	inquiry result of websites selected from Alexa search
count of repeated records which differs than the previous record	278	24	0
Average caching period (in seconds)	1.723	1424	0
Average caching period (in minutes)	29	24	0
count of all URLs having size > 0	543	85	99
count of URLs having size = 0	0	0	8
total no. of repeated records having size > 0	350	29	1
observation of same URL if it exists at least twice (unique count for each repeating URL)	85	18	0
Average URL size	6.528.715	10.455.197	6.029.904
Average URL width	77	78	95
Average internal links	147	174	155
Average external links	272	150	90
Average HREF link info	12.624	12239	12692
Average IMG link	158	331	230
Average FORM link	0	0	0
Average search duration (in seconds)	198	271	212
Ratio of all URLs having size > 0 and all repeated records having size >0	1,55	2,93	99,00
Ratio of all repeated URLs having size > 0 and tot. no. of unique repeated records	4,12	1,61	

Table 40: General Features of Turkish Websites other than com.tr, edu.tr, gov.tr, net.tr, org.tr and info.tr Extensions Derived from Three Different Search Algorithms

	result of inquiry with 19 keywords selected intuitively	result of inquiry with 50 keywords from Dr. İlyas Göz research	inquiry result of websites selected from Alexa search
count of repeated records which differs than the previous record	2.200	225	0
Average caching period (in seconds)	1.789	1499	0
Average caching period (in minutes)	30	25	0
count of all URLs having size > 0	5.599	779	852
count of URLs having size = 0	190	18	101
total no. of repeated records having size > 0	3.663	280	1
observation of same URL if it exists at least twice (unique count for each repeating URL)	1.016	169	1
Average URL size	4.294.653	14.914.136	6.607.318
Average URL width	25	62	67
Average internal links	66	137	129
Average external links	55	141	85
Average HREF link info	3.629	10472	10263
Average IMG link	96	387	242
Average FORM link	0	1	1
Average search duration (in seconds)	68	206	182
Ratio of all URLs having size > 0 and all repeated records having size >0	1,53	2,78	852,00
Ratio of all repeated URLs having size > 0 and tot. no. of unique repeated records	3,61	1,66	1,00

Table 41: Word Frequency List Driven from Turkish Websites Found from Alexa.com

Order	Frequency of words found in our work	Words found in our work	Intersection of two list: order of matching words	Words from Dr. Göz's study	Frequency of words found in Dr. Göz's study	Order in Dr. Göz's study
1	432088	ve	16	bir	29296	1
2	363940	mesaj	6	olmak	20844	2
3	219300	resim	17	için	6886	3
4	217532	konu	49	ben	5829	4
5	201660	var		demek	5419	5
6	196651	ol		çok	5405	6
7	193182	son	51	yapmak	5189	7
8	188577	oyun		gibi	4994	8
9	184093	izle		daha	4683	9
10	182431	üye		almak	4422	10
11	171157	haber	5	var	4200	11
12	146103	yeni		kendi	4175	12
13	142670	filim		gelmek	4033	13
14	125653	göster	26	ile	3830	14
15	125242	ad		vermek	3827	15
16	120689	bir		ama	3668	16
17	106162	için		sonra	3639	17
18	103672	vb		kadar	3627	18
19	101687	indir		yer	3366	19
20	99219	yazan		insan	3352	20
21	98946	yazı		değil	3273	21
22	97169	bölüm		her	2924	22
23	97096	bu		istemek	2859	23
24	91652	git		yıl	2849	24
25	90134	sayfa		çıkmak	2777	25
26	72075	ile		görmek	2750	26
27	68982	türkce		gün	2633	27
28	66216	yok		biz	2621	28
29	63399	okunan	24	gitmek	2600	29
30	59415	tema		iş	2553	30
31	58239	bilgiler		şey	2549	31
32	57997	etiket	50	ara	2528	32
33	57050	cevap		bilmek	2448	33
34	55839	genel		zaman	2394	34
35	55172	programı		çocuk	2326	35
36	53064	yaz		iki	2294	36
37	52778	dünya		bakmak	2252	37
38	50390	hakkımızda		çalışmak	2184	38
39	49834	arşiv		içinde	2152	39
40	49683	isim		büyük	2133	40
41	49221	videolar	28	yok	2130	41
42	47102	paylaş		başlamak	2064	42
43	46966	kategori		yol	1994	43
44	46545	görüntü		kalmak	1947	44
45	45213	dizi		neden	1941	45

Table 41 Continued.

46	44763	gönder
47	44273	sezon
48	43106	türkiye
49	42837	ben
50	42533	arama
51	42061	yap

	siz	1939	46
4	konu	1934	47
51	yapılmak	1913	48
	iyi	1907	49
	kadın	1898	50

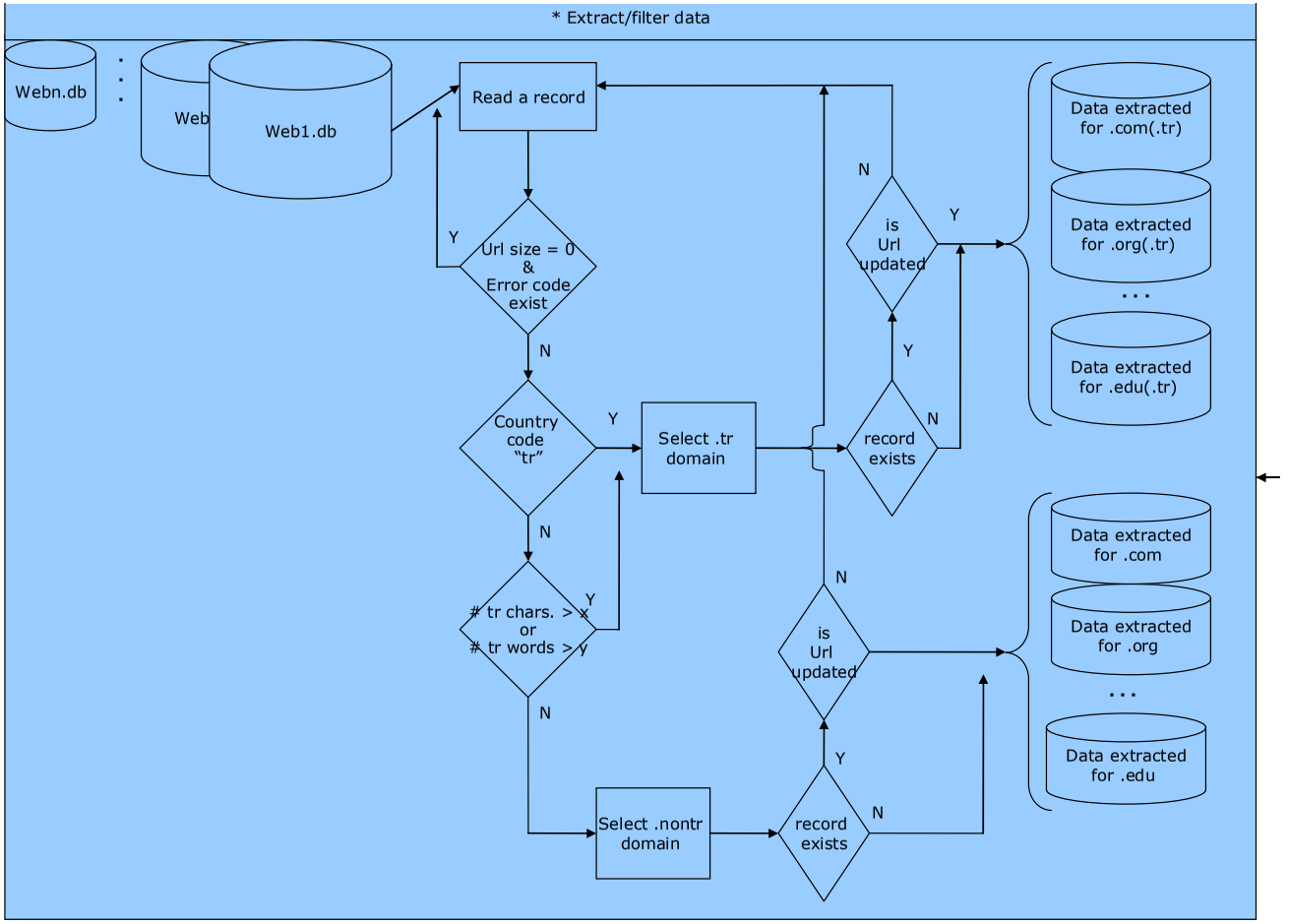

Figure 20: Data Extract/Filter from Web.db1 (…dbn) to the related domain Dbs.