CATEGORIZATION OF WEB SITES IN TURKEY WITH SVM

Kadir ŞİMŞEK

JULY, 2004

Categorization of Web Sites
In Turkey With SVM

By

Kadir ŞİMŞEK

A Dissertation Submitted to the

Graduate School in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Computer Engineering

Major: Computer Software

Izmir Institute of Technology

Izmir, Turkey

July 2004

We approve the thesis of **Kadir ŞİMŞEK**

Date of Signature

27.07.2004

.....
Prof. Dr. Halis PÜSKÜLCÜ
Supervisor
Department of Computer Engineering

27.07.2004

.....
Prof. Dr. Sıtkı AYTAÇ
Department of Computer Engineering

27.07.2004

.....
Prof. Dr. Fikret İKİZ
Ege University
Department of Computer Engineering

27.07.2004

.....
Prof. Dr. Sıtkı AYTAÇ
Head of Department

ACKNOWLEDGEMENTS

I would like to depict my deepest gratitude and respects to my advisor Prof. Dr. Halis Püskülcü for his enduring support, advice, suggestions, encouragement and supervision throughout this study.

I also owe my special thanks to my family, Mrs. Remziye ŞİMŞEK and Mr. Kutbeddin ŞİMŞEK, for their encouragements and patience.

ABSTRACT

In this study of topic "Categorization of Web Sites in Turkey with SVM" after a brief introduction to what the World Wide Web is and a more detailed description of text categorization and web site categorization concepts, categorization of web sites including all prerequisites for classification task takes part. As an information resource the web has an undeniable importance in human life. However the huge structure of the web and its uncontrolled growth led to new information retrieval research areas to be risen in last years. Web mining, the general name of these studies, investigates activities and structures on the web to automatically discover and gather meaningful information from the web documents. It consists of three subfields: "Web Structure Mining", "Web Content Mining" and "Web Usage Mining". In this project, web content mining concept was applied on the web sites in Turkey during the categorization process. Support Vector Machine, a supervised learning method based on statistics and principle of structural risk minimization is used as the machine learning technique for web site categorization.

This thesis is intended to draw a conclusion about web site distributions with respect to thematic categorization based on text. The popular web directory Yahoo's 12 top level categories were used in this project. Beside of the main purpose, we gathered several statistical descriptive informations about web sites and contents used in html pages. Metatag usage percentages, html design structures and plug-in usage are some of these information. The processes taken through solution, start with employing a web downloader which downloads web page contents and other information such as frame content from each web site. Next, manipulating, parsing and simplifying the downloaded documents takes place. At this point, preperations for categorization task are completed. Then, by applying Support Vector Machine (SVM) package SVM^{Light} developed by Thorsten Joachims, web sites are classified under given categories. The classification results obtained in the last section show that there are some over-lapping categories exist and accuracy and precision values are between 60%-80%. In addition to categorization results, we saw that almost 17% of web sites utilize html frames and 9367 web sites include meta-keywords.

ÖZ

Bu çalışmada, Türkiye'deki *".tr"* uzantılı Web sitelerinin SVM (Support Vector Machine) ile sınıflandırılması yapılmıştır. Web'in kısa bir tanımı yapıldıktan sonra metin sınıflandırması ve web sitesi sınıflandırılması konuları anlatılmıştır. Sınıflandırma işlemi için gerekli kelime ayıklama, kelimelerin ağırlıklarını bulma gibi tüm önkoşullar yerine getirildikten sonra sınıflandırma işlemi tamamlanmıştır. Web'in devasa yapısı ve kontrol edilemeyen genişlemesi son yıllarda yeni araştırma alanlarının ortaya çıkmasını sağlamıştır. Bu çalışmaların genel tanımı olarak bilinen *Web madenciliği* web üzerindeki yapı ve hareketleri inceleyerek Web'den anlamlı bilgilerin otomatik bir biçimde alınmasını sağlar. Web madenciliği üç alt daldan oluşmaktadır: "Web Yapı Madenciliği", "Web İçerik Madenciliği" ve "Web Kullanım Madenciliği". Bu projede, "Web İçerik Madenciliği" yapılarak Türkiye'deki Web siteleri sınıflandırılmıştır. Sınıflandırma esnasında, yapısal risk minimizasyonu ve istatistik tabanlı denetlemeli öğrenme yöntemi olarak tanımlanan "Support Vector Machine" (SVM) algoritması kullanılmıştır.

Bu tezle metin tabanlı bir sınıflandırma yöntemi ile web sitelerinin temalarına göre dağılışları elde edilecek ve aynı zamanda SVM gibi istatistiksel bir programın kullanım sürecinin hangi aşamalardan oluştuğu görülecektir. Sınıflandırma işlemi için Yahoo'nun üst katmanda yeralan 12 sınıfı kullanılmıştır. Tez sonuçları ayrıca web siteleri ve içerikleri hakkında özellikle html tasarımı ve sayfa yapısı ile ilgili birtakım bilgileri de içermektedir. Metatag kullanım yüzdeleri ve html tasarım yapıları gibi çıkarımlar bu bilgiler içinde yeralmaktadır. Sınıflandırma, sayfaların gövde metni bölümünde yeralan bilgiler ve Thorsten Joachims'in geliştirdiği SVM^{light} paketi kullanılarak yapılmıştır.

Sınıflandırma sonuçları bazı sınıfların çakıştığını göstermektedir. Sonuçların doğruluk ve kesinlik değerlerinin 60%-80% aralığında olduğu gözlenmiştir. Sınıflandırma sonuçlarına göre html sayfa içeriklerinin homojen olmadığı ortaya çıkmış, bu nedenle sınıflandırma işleminin olumsuz yönde etkilendiği gözlenmiştir. Sınıflandırma sonuçlarının yanısıra, web sitelerinin yaklaşık 17% 'si html çerçevelerini ve 9367 web sitesinin *meta-keyword* etiketlerini kullandığı sonucuna varılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 MOTIVATION

World Wide Web (the "Web" or "WWW" for short) is the large and single set of public web sites which is also defined as a network structure of hyperlinked environment. In the span of a decade, the Web has grown rapidly and become a complicated network system that serves as a huge repository of information. It has been adopted all over the world with a high degree of acceptance.

The Web does not have a structured architecture like other networks. It is now getting more complicated whenever a new web site is created. This leads to an uncontrolled growth in size of the Web. Therefore, most precious resources can not be located with just randomly surfing the Web. In this situation a user generally can not see the actual content that he/she wants to get from the Web. In order to overcome the lack of usage, researchers have been proposing ideas about how the users can optimally benefit from the Web. There are a number of research problems for which web studies have investigate the solutions.

Web site categorization is a task of research on the web. It deals with text data reside on the web documents. Text contents in web pages are valuable for organizing the data on the web. As the number of documents on digital devices increases, web site categorization studies have focused on automatic knowledge discovery tasks that employ statistical machine learning algorithms for generating a classifier. Vapnik's Support Vector Machine [Vapnik, 1995] is such a technique for pattern recognition, regression problem and learning of a classification function. Its effectiveness is proofed by many researches about the topic. Yiming Yang and Xin Liu, for instance, [Yand and Liu, 1999] compared performance of a number of machine learning algorithm in an automated text categorization process in which SVM showed the best accuracy performance. In this study, therefore, we preferred to use SVM in categorization task of the web sites in Turkey.

Web domain names are consisted of .com, .org, .net, .edu, .biz, .info domain names which are called gTLD (generic Top Level Domain) in international area and almost 200

country codes of .uk, .us, .dc, .tr, etc. which are called ccTLD (Country Code Top Level Domain) having ISO-3166 standards. Method and coordination of ".tr" up-level domain name and secondary level domain names under ".tr" have been maintained by Middle East Technical University (METU) since 1990 when it was recorded to Internic for the first time. Turkey has adopted "secondary level domain names model" following the decision taken by TÜVAKA (Türkiye Üniversiteler ve Araştırma Kurumları Ağı) in 1990. "DNS Çalışma Grubu" which works under "Internet Kurulu" and consisted of 11 company representatives determines the domain names strategies and rules. List of secondary level domain names that are deployed by *".tr" Domain Name Administration* is given below:

".com.tr", ".info.tr", ".biz.tr" sub-domain names can be taken by companies and people in business. These sub-domain names were opened in order to increase the number of internet based services (web pages, e-mail, etc...) and domain names of companies under the up-level domain name ".tr", by the way spreading the use of internet in business.

The domain names can only be used by companies and people in business. However, ".com.tr" sub-domain name can be used by municipalities supplying that it is limited to city names and it is used to establish a city portal.

".net.tr" is a sub-domain name which is established for various internet service supplying companies. It is given to companies that provide net connection service, portal over internet, search engine, e-mail, wide range of services such as Web and application service. In addition to companies that provide connection service, it is aimed to contain projects that transmit the services to internet and projects that improve public benefit and public participations.

".org.tr" sub-domain name was established for foundations, associations, non-profit organizations and such kind of public organizations.

".web.tr", ".gen.tr" sub-domain names are established for formation, company and enterprises that will give public and/or business service on the web, but there is no limitation for assigning these names provided that there is no violation of general rules. It is aimed that persons, companies and formations would be present in the net. Personal and foundational applications can be accepted for these names. There are general rules like "no necessity for relationship between domain name and foundation" and "first comes, gets".

They were established in order to increase the number of personal and company domain names under the ".tr" domain name, by the way spreading the use of internet.

".bbs.tr" was established for the use of persons and/or companies giving BBS (Bulletin Board System) service.

"name.tr" sub-domain name was established for personal use of T.C. citizens and foreigners living in Turkey. It aims to increase the number of personal domains and services (web pages, e-mail etc…) under the ".tr" domain name.

".tel.tr" sub-domain name is set up for persons and companies staying in Turkey. The aim is to establish a unique sub-domain name for the purpose of publishing all personal or corporate phone numbers.

".gov.tr" sub-domain name is established to serve for institutions of Republic of Turkey (T.C.) government. Its usage is limited to only institutions of T.C. government.

".mil.tr" sub-domain name is established to serve for institutions of T.C. military.

".k12.tr" sub-domain name is established for primary and secondary schools approved by T.C. national education ministry.

".edu.tr" sub-domain name is setup for being used by universities and institutions accepted by T.C. Yüksek Öğretim Kurumu (YÖK).

".av.tr" sub-domain name is defined for being used by lawyers who are member of "Türkiye Barolar Birliği" (TBB). Self-employed lawyers, law offices and advocacy corporations can apply this sub-doman name. It aims to spread the professional use of internet and make it possible that lawyers present their social contributions and improve their professional progress. TBB is authorized to approve or refuse the requests.

".bel.tr" sub-domain name is established for being used by T.C. municipality organizations. Its usage is limited to only T.C. municipalities.

".pol.tr" sub-domain name is established for being used by T.C. police organizations. Its usage is limited to only T.C. police organizations.

1.2 OBJECTIVES

Web sites in Turkey are logically grouped under certain sub-domain names that are mentioned above. The purpose of the naming scheme of domain names is to define a set of

classes that distinguish web sites of the same domain from others. Because of the huge web structure that increases rapidly and growing complexity of the web, Internet users mostly need some kind of classification upon web sites to see their ways on the net until they reach desired pages. The classes that arise from the structure of domain names are few and not flexible enough for covering users' needs.

Some web directories like *http://www.netbul.com* serve categorization services to internet users for better navigation on the net. However categorization of web sites is done by hand or traditional methods like hard-coded conditional statements that decide the category of a web site by use of keywords. In last years, statistical machine learning algorithms have become popular for categorization of text content. With this issue in mind, we decided to categorize the web sites of which domain names end up with .tr by using a machine learning approach. During our research about the thesis topic, we have not encountered a study that categorizes web sites in Turkey by using statistical methods.

In this study, we classified web sites in Turkey under predefined categories. We employed a package called SVM^{Light} that is an implementation of widely accepted effective machine learning algorithm "Support Vector Machines". We selected a subset of categories from well known internet directory service, Yahoo, for this study.

In addition to categorization, valuable information was extracted from the web documents. Web sites were examined according to number of words for each type of metatag included in site. There are four types of tags that would be considered: *Html Page Title, Meta-Description, Meta-Keywords* and *Body Text***.** Web sites were grouped into four groups for each type of tags listed above. These groups are sites that contain 0 words, 1-10 words, 11-50 words and lastly 50 and more words. Moreover, html frame usage and plug-in usage ratio were investigated besides to classification task.

1.3 STRUCTURE OF THE STUDY

Web mining is a great research area which branches into a number of sub-fields. In Chapter 2 we introduced knowledge discovery on the web and scope of web mining techniques including web structure mining, web content mining and web usage mining.

Chapter 3 sets the focus from general web mining studies in Chapter 2 to a more specific topic, *text categorization*. After a brief introduction to text categorization applications, text categorization process is described in detail. Both traditional knowledge engineering approach and machine learning approach is mentioned stressing advantages of the machine learning. This chapter also covers all preprocessing tasks needed before learning a classifier and all performance calculations needed after classification in machine learning approach.

In chapter 4, we introduced SVM algorithm followed by a description of well known SVM implementation SVM^{Light}.

Chapter 5 is concerned with all steps from project analysis to obtaining the classification results. It also covers the presentation and discussion of all the statistical findings gathered from structures of the web sites.

Chapter 6 summarizes the results and mentions some possible future workings on the topic.

Chapter 2

WEB'S STRUCTURE AND KNOWLEDGE DISCOVERY

World Wide Web is the large and single set of public web sites which is also defined as a network structure of hyperlinked environment. In this large network, while the nodes are represented by web pages, the hyperlinks between pages constitute the connection. The Web is a *virtual* network of pages and hyperlinks, with over a billion interlinked "documents" created by tens of millions of individuals not knowing each other [Kleinberg and Lawrence 2001].

In the span of a decade, the Web has become a huge repository of information. According to [Levene and Poulovassilis 2001] one way of measuring the size of the Web is to estimate the number of pages that can be indexed by well known public search engines such as Google and Altavista. This measure is often referred to the *publicly indexable Web* [Lawrence and Giles 1999]. The term "deep" Web is the name given to the invisible part of the web that reside in searchable databases [Levene and Poulovassilis 2001]. On the "deep" Web, pages are dynamically generated by web servers with server side scripting languages such as .jsp (Java Server Pages) and .asp (Active Server Pages). Besides, these databases are only available to and reachable from only the web server that generates pages. Therefore it is not possible to crawl, analyze and index those pages. The "deep" Web and searchable databases topic is a completely different topic, therefore this hidden part of the web is not included in scope of this thesis. In the rest of the research, we will imply the *publicly indexable Web* with the term "Web".

There are a number of research problems for which web studies have investigate the solutions. A study by Broder et al. about the graph structure of web was performed in 2000 [Broder et al. 2000]. We obtained the results of this study from the articles in [Levene and Poulovassilis 2001, Kleinberg and Lawrence 2001, FürnKranz 2002] and made a summary as follows. The most interesting result is that the web structure looks like a huge bow tie with a strongly connected core component in which every page can reach every other by a path of hyperlinks. This core contains most of the famous sites. The remaining pages can be characterized by their relation to the core: Upstream nodes can reach the core but cannot be

reached from it, downstream nodes can be reached from the core but cannot reach it, and "tendrils" contain nodes that can neither reach nor be reached from the core.

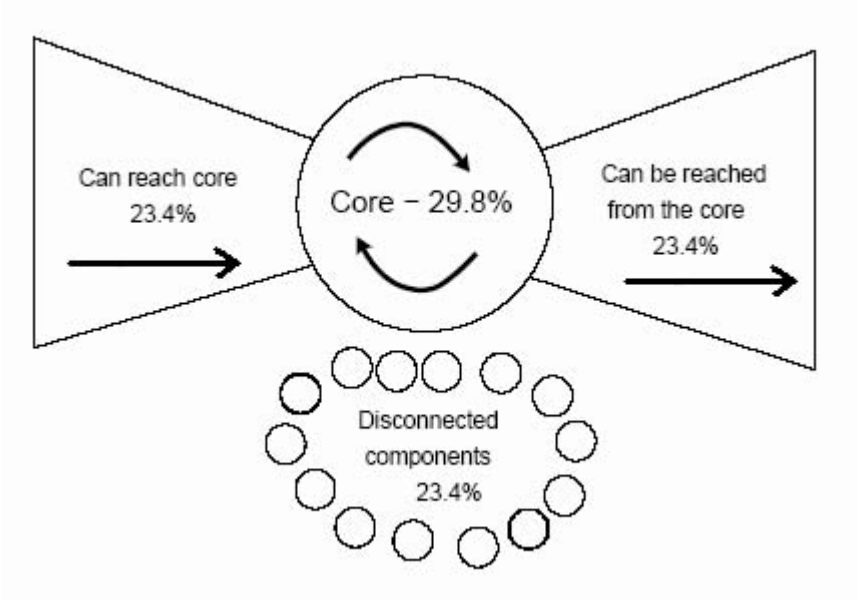

Figure 2-1: The bow-tie shape of the Web [Broder et al. 2000]

The core covers nearly 30% of the Web. While the left bow contains the "upstream" nodes, the right bow includes the "downstream" nodes. According to the results 75% of the time there is no directed path between randomly selected two web pages and the average path distance is 16 hyperlinks.

2.1 KNOWLEDGE DISCOVERY ON THE WEB

The Web provides lots of information to the internet users. However growth in size of the Web makes it increasingly difficult to find out relevant information on the Web. This section introduces the information retrieval techniques on the web, *Web Mining,* in general.

The term *Web Mining* was originally thought by Etzioni [Etzioni 1996]. *Web Mining* applies data mining[1] and machine learning[2] to web data and link structure of the

[1] the technology used to discover non-obvious, potentially useful and previously unknown information from data sources [www.mineit.com]
[2] learning approach that learn from experimence E with respect to some class of tasks T and performance measures P [Keller 2000]

Web [FürnKranz 2002]. It investigates the content and activities on the Web and processes this data to automatically discover and gather meaningful information from the Web documents. It covers all studies that are related with web research and analysis. However, searching and using semi-structured information stored on the web is more difficult than the information that proprietary database systems store. To partially reduce the complexity of web mining studies, the research areas are grouped into three categories as *Web Structure Mining*, *Web Usage Mining* and *Web Content Mining* in general.

Web Structure Mining defines the Web as a graph in which each web document forms a graph vertex and each hyperlink between pages forms an edge of the graph. It covers analysis of the logical structure of the Web that consists of web pages and links. *Web Content Mining* is actually *mining of text*. It deals with web document contents such as textual, image, audio and video to discover knowledge. The task of content mining covers applications like web page categorization, clustering, filtering and ranking. *Web Page Content Mining* and *Search Result Mining* are the two research areas of *Web Content Mining* [Zaiane 1999]. Lastly, *Web Usage Mining* is the mining of log files and associated data from a particular web site to discover knowledge on browser and buyer behaviour on that site [www.mineit.com]. While Web structure mining and Web content mining focus on the global network, Web usage mining concentrates on a specific web site to improve it for better navigation, easy usage and so on.

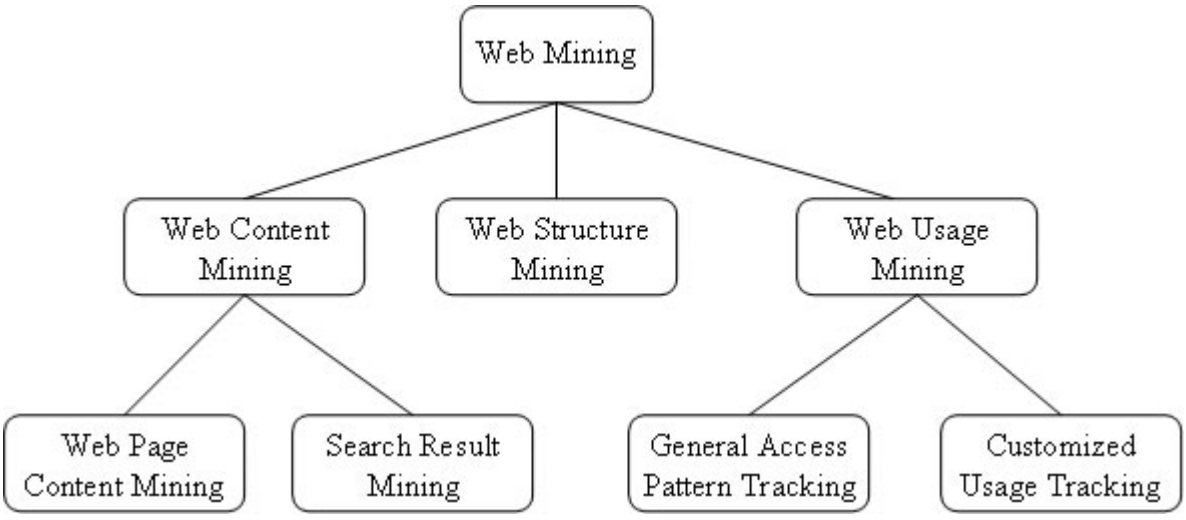

Figure 2-2: Taxonomy of web mining techniques [Zaiane 1999]

2.1.1 Web Content Mining

Basically, the Web content consists of several types of data such as textual, image, audio, video, metadata as well as hyperlinks. Recent research on mining multi types of data is termed multimedia data mining [Zaiane et al. 1998]. Thus multimedia data mining can be considered as an instance of Web content mining. However this line of research still receives less attention than the research on the text or hypertext contents [Zaiane et al. 1998]. The research around applying data mining techniques to unstructured text is termed knowledge discovery in texts [Feldman and Dagan 1995], or text data mining [Hearst 1999], or text mining [Tan 1999]. Hence we could consider text mining as an instance of Web content mining [Kosala and Blockeel 2000].

Web Content Mining is a subcategory of Web mining which deals with text data reside on the web documents. Text contents in web pages are valuable for organizing the data on the web. It is generally used for a number of applications such as web page categorization, clustering of web documents and web content filtering. There are two groups of web content mining strategies: Those that directly mine the content of documents and those that improve on the content search of other tools like search engines [Zaiane 1999]. The latter, *Search Result Mining*, uses brief descriptions of search results that generated after a query is performed on a search engine as an input to make further operations. Classifying the results into predefined categories and narrowing the search results are the examples of search result mining applications.

Several applications of knowledge discovery which uses the web content as a resource exist. Text classification and clustering applications are the most interesting application types in this domain. One example is a hierarchical classification of Web content by S. Dumais and H. Chen [Dumais and Chen 1999] which uses SVM for text categorization process. The purpose of this study was to improve the ranked list view of results by organizing web search results into hierarchical categories. They used a large heterogeneous collection of pages from LookSmart's web directory which consisted of 370597 unique pages at the time of experiment (1999). Search results was classified under 13 top-level categories which consist of a total of 150 second-level categories.

In another study, Berkant Barla Cambazoğlu from Bilkent University [Cambazoğlu 2001], applied some well-known classification techniques to text categorization. The study is mainly concentrates on application of text categorization to documents having content written or spoken Turkish. k-NN and naïve Bayesian algorithms are compared on two different Turkish datasets, each from different domains.

2.1.2 Web Structure Mining

Web Structure Mining defines the Web as a graph in which each web document forms a graph vertex and each hyperlink between pages forms an edge of the graph. The aim of Web structure mining is to extract knowledge from the interconnections of hypertext documents in the Web. There are two famous algorithms that mine structure of the Web. Next subsections describe these studies namely HITS and PageRank. A more detail analysis of HITS and PageRank is placed in *Appendix.*

2.1.2.1 Authoritative Pages and HITS Algorithm

Search engines such as Altavista return thousands of relevant pages on the Web when one enters a query like "computers". Kleinberg defines this abundance problem of search engine results as "The number of pages that could reasonably be returned as relevant is far too large for a human user to digest" [Kleinberg 1998]. According to Kleinberg, one needs a way to filter among a huge collection of relevant pages, a small set of the most "authoritative" and "definitive" ones to provide effective search methods [Kleinberg 1998]. In his study, Kleinberg identifies two kinds of pages by analyzing web page links:

- pages that are very important and authorities in a *special* topic (autorities),
- pages that have great number of links to *multiple* relevant authority pages (hubs).

Kleinberg proposed an algorithm to identify these pages on the web.

Hyperlink-Induced Topic Search (HITS) algorithm was introduced to filter the most relevant resources, namely good authorities and good hubs, from enormous amout of information returned from search queries. HITS is a purely *link structure* based

computation that ignores the textual content entirely. The text that surrounds hyperlink definitions (href's) in Web pages is often referred to as anchor text.

The main drawback of the HITS algorithm is that the hub and authority score must be computed from the query result, which does not meet the real-time constraints of an on-line search engine [Fürnkranz 2002]. It generates a different root set depending on the search results of given query string for each execution. Another limitation of HITS is that it filters the results of broad-topic queries to narrow the result set and neglects specific queries [Jing et al. 2001]. each filters The next subsection describes the implementation of a similar algorithm, namely *PageRank* that comes with a major advantages over HITS algorithm in respect to real time working status.

2.1.2.2 PageRank Algorithm

Web pages vary on a wide range in respect to their quality and importance. In order to measure the relative importance of web pages Sergey Brin and Lawrence Page proposed PageRank, a method for computing rank of every web page based on the graph of the web. PageRank is a numeric value that represents how important a page is on the web [Craven 2003]. It is an important part of the ranking function of the Google search engine. PageRank has applications in search, browsing and traffic estimation [Brin and Page 1998a]. It ranks all web pages based on just the link structure of the Web. Google uses this technique to order search results so that more important and central Web pages have preference over other pages. It is an efficient and effective way of ranking web sites.

2.1.3 Web Usage Mining

J. Fürnkranz defines *Web usage mining* as an analysis of user interactions with a Web server [Fürnkranz 2002]. Web servers logs an entry to log files for every access they get. Log entries includes but not limited to IP address of the user, time of every access, duration of every visit and URLs of accessed pages [Romanko 2002]. Romanko summarizes applications of *Web usage mining* as follows:

- Grouping together visitors that have same behavior.

- Association of frequently visited pages.
- Display of customer comments on e-commerce web sites.
- Explicit feedback to get recommendations.
- Reorganizing web pages according to users behavior.

Web usage mining applications collect valuable, statistical information about a web site's usage and investigate ways of improving web site design and quality for best navigation and performance.

Web usage mining, thus, differs from *Web content mining* and *Web structure mining* in that its application domain is limited to a single web site. Other two types of mining can be applied anywhere from a small portion of the Web to huge snapshots of the Web.

2.2 SEARCH ENGINES AND WEB DIRECTORIES

Searching on the Web is one of the most interesting topics. By uncontrolled growth of the Web size and unsystematic structure of the Web, it has become a difficult and disappointing research topic. *Search engines* and W*eb directory services* are the two well known techniques that introduced as outcome of web mining studies for optimal usage of the web. *Search engines* have a usage based on keywords. User inputs one or more keywords and sends the query to the search engine. Engine searches its database for these keywords and returns all related documents that contain at least one keyword. There are a variety of researches to improve the search on the Web. In this part of the thesis we will discuss and categorize the Web search technologies.

A *Web directory* uses heavily human labour to review web sites and make decisions for classifying web sites into predefined categories or subjects. A web directory differs from a search engine in that, it present information in an organized way by using categorization. Keyword usage does not take place in directories rather user selects a category, then a sub-category until he/she reaches desired web sites and resources. In the last years machine learning techniques have been used to group web sites under predefined categories.

Search engines, however, are complicated than directory services. A search engine consists of three components: the *crawler*, the *indexing software* and the *search and*

ranking software [Yuwono and Lee 1996, Greenberg and Garber 1999]. Figure 2-9 shows the system structure of a search engine.

A **crawler** is a program that automatically traverses the Web page by page, read and store every web document. It follows hyperlinks on the visited site to find other relevant pages. Crawler is a typical application of web structure mining which views the Web as a huge graph with URLs treated as the nodes of the graph. The objects reside on the URLs could be HTTPs(Hypertext Transfer Protocols), FTPs(File Transfer Protocols), mailto(e-mal), news, telnet, etc [Nansi and Murthy 2002]. Crawlers use two common graph algorithms, breadth-first searches and depth-first searches, to find out relevant pages.

Chakrabarti introduced a new learning system called a *focused crawler* [Chakrabarti 1999], which is a specialised crawler with the objective of seeking out Web pages on a predefined set of topics. *Focused crawlers* are different from general-purpose crawlers utilized by search engines to create their offline databases of Web pages. The focused crawler starts from a small set of relevant URLs on the same topic. While crawling, system uses a classifier that decides which links to follow from each visited page according to their relevance to the query.

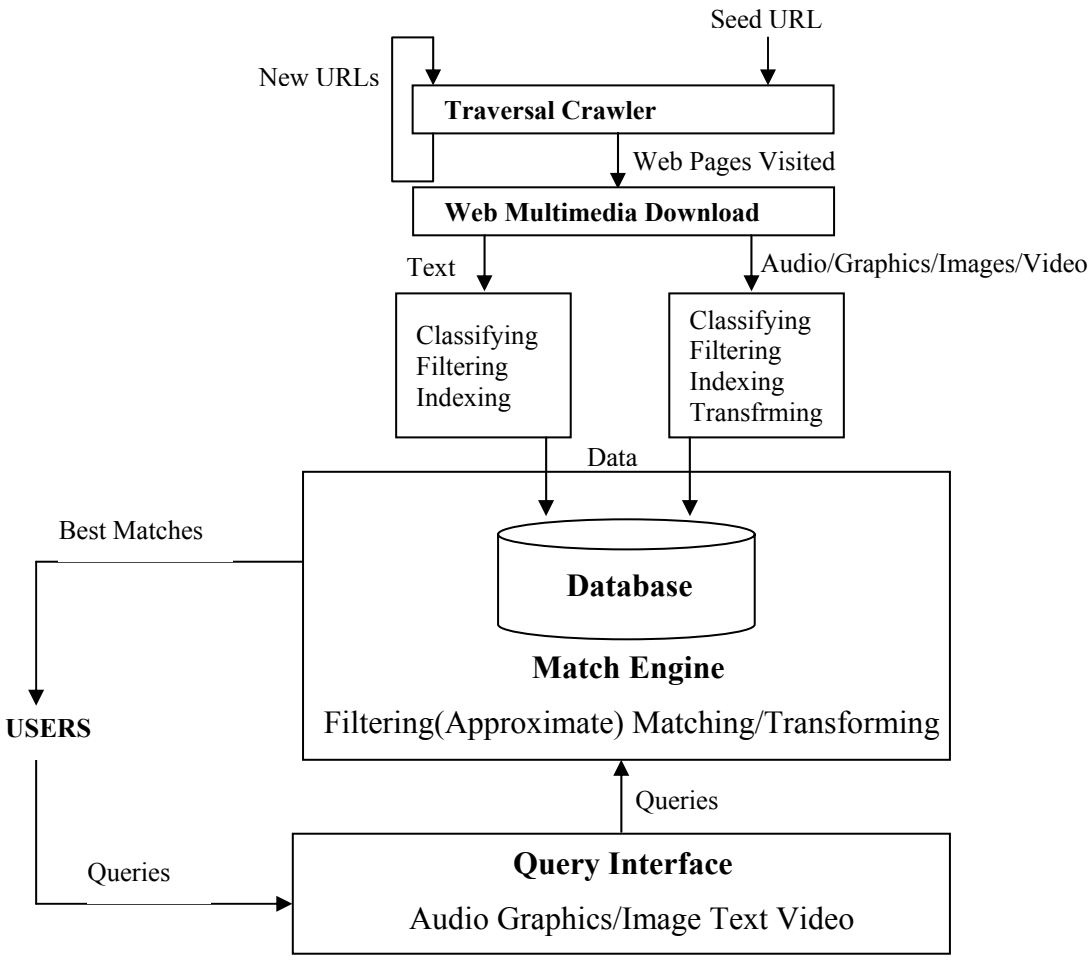

Figure 2-3: System structure of a search engine [Nansi and Murthy 2002]

The information collected by a crawler is not appropriate for quick keyword search. Thus **indexing software** that constructs a data structure from collected items needed inside of the search engine system. Traditional search engines utilize the following information, provided by HTML files, to locate the desired Web pages:

- *Content:* Most accurate and full-text information
- *Descriptions:* Constructed from metatags or submitted by Webmasters
- *Hyperlinks:* Contain high-quality semantic clues to a page's topic
- *Hyperlink Text:* A title or brief summary of the target page
- *Keywords:* Can be extracted from content or metatags
- *Page title:* Defines the title of an HTML document
- *Text with a different font:* Shows the importance of the emphasized text

- *The first sentence:* Likely to give crucial information related to the document

During query processing **search and ranking software** searches indexed database and ranks the results of query according to importance of the web sites retrieved. The ranking can be dynamically performed just after scanning the indexed database or precedence values of each retrieved site can be statically stored in the database. The latter case which is the efficient approach is the most popular one.

There are a number of search engines on the Web. The problems with search engines have never end. Thus, search engines have renovated themselves to keep up with accelerated growth of the Web since early 1990's. Here is a quotient from a survey by M.Levene and A.Poulovassilis about search engine generations.

> First-generation search engines were based on classical information retrieval models, such as the vector space model, and support HTML parsing and weighting. Second-generation search engines employ link analysis, such as the Google pagerank algorithm, and utilise anchor text in order to provide some navigational support. Third-generation search engines aim to reflect users' needs by detecting the context of a query. This context could be spatial, textual, the user profile, or previous user queries based on data mining analysis. Third-generation search engines also have to deal with dynamic Web-page content.
>
> [Levene and Poulovassilis 2001]

Search engines, now, are categorized in different classes according to their aspects and services they offer. Primary search engines are the best engines and more preferable than the others. Google, MSN Search, Teoma and Yahoo! get in this group. Secondary search engines group contains smaller search engines such as Gigablast, Hotbot, iWon and WiseNut. In the third group there are dead search engines. Following list shows a number of dead search engines which have abandoned their place in searching domain, although they still may have some kind of search functionality.

- AlltheWeb (Switched to Yahoo! database in March 2004)
- AltaVista (Switched to Yahoo! database in March 2004)
- Deja.com (Dead Usenet search, bought by Google and became Google Groups)
- Direct Hit (Dead, redirecting to Teoma)
- Excite (Dead as a separate database, now uses an InfoSpace meta search)
- Excite News (NewsTracker) (Dead)
- Go (Dead as a separate database, took over Infoseek, but now just uses Overture)

- Go (Infoseek) News (Dead)
- Infoseek (abandoned)
- InvisibleWeb.com (a hidden Web directory, dead by 2003)
- Flipper (Hidden Web databases from Quigo, dead by Fall 2003)
- iWon (Old Inktomi version dead. Now uses Google "sponsored" ads and Web and image databases)
- Lycos (Switched to Yahoo!/Inktomi database in April 2004, still using 10 LookSmart directory results.)
- Magellan (Dead, redirects to WebCrawler)
- MessageKing (Dead Web forum search engine as of Fall 2003)
- NBCi (formerly Snap) (Dead, now uses metasearch engine Dogpile)
- NBCi Live Directory (formerly Snap) (Dead directory)
- Northern Light (Dead as a Web search engine as of 2002.)
- Northern Light Current News (Dead. Updates ceased as of Feb. 28, 2003.)
- Openfind (Under "reconstruction" as of 2003)
- WebCrawler (Dead as a separate database, Same as Excite)
- WebTop (Dead)

Furthermore, there are web directories that are completely different from the search engines. The information is organized by topic in web directories. A web directory organizes the sites by categories to help the user in browsing and finding relevant pages. Most of the directory services are maintained by human. The directories are constructed according to the web pages downloaded by using registered site lists. Sites are registered by web site owners who want to be listed on a specific web directory. In addition, the web topics are also populated with some interesting web sites by fully using human effort in randomly searching and browsing the web. Due to the different approaches for presenting web content, directories and search engines have different uses, as the difference between a book's table of contents and index [www.yahoo.com]:

- When looking for a specific site (something known by name), a directory is used
- When looking for limited lists of the best sites in the given subject, a directory is used

- When looking for a wide list of results related to keywords, a search engine is used

The commonly used web directories are Britannica, LookSmart, The Open Directory (Dmoz) and Yahoo!. More detailed search engine categorization and statistics can be found at http://searchengineshowdown.com/reviews/ which we take search engine categorization data from.

Chapter 3

MACHINE LEARNING AND TEXT CATEGORIZATION

In the last years number of documents in digital form increased rapidly. Management of these documents takes an important place in the information systems area. Text categorization (or text classification) is one of the studies to organize documents according to their contents. Text categorization is being utilized in many circumstances such as document indexing, document filtering, any application that requires document organization, web page categorization and so forth.

There are two approaches used in text categorization. The first one is the conventional *knowledge engineering* that was popular in the text categorization field before machine learning techniques were introduced. In this approach, domain experts define a set of rules manually to classify documents under certain categories. *Machine learning* approach on the other hand, applies a general inductive process and a set of preclassified documents under predefined categories to automatically construct an automatic text classifier. The advantages of machine learning are an acceptable accuracy and a considerable savings in terms of expert manpower, since no intervention from either knowledge engineers or domain experts is needed for the construction of the classifier or for its porting to a different set of categories [Sebastiani 2001].

In *machine learning* we have two primary approaches namely *supervised learning* and *unsupervised learning*. In *supervised learning* approach, machine learning algorithm builds the classifier from a collection of precategorized documents. Text categorization process is a good example for supervised learning. However there is no any precategorized data in unsupervised learning approach. All categories are dynamically defined. Algorithms use document similarity techniques to discover similarities and automatically cluster documents to dynamically generated groups. Text clustering, for instance, is an unsupervised learning application domain that clusters incoming documents into categories dynamically by a function defined based on document similarity. There are two types of clustering algorithms, namely hierarchical clustering and non-hierarchical clustering. The

k-nearest neighbor method is the most widely used hierarchical clustering method. For non-hierarchical clustering, one of the most common approaches is the K-means algorithm.

In order to categorize web sites into predefined categories we will use a successful supervised learning algorithm, Support Vector Machines (SVM) after collecting some web sites into predefined categories. In this chapter we will discuss the text categorization process in general excluding any machine learning algorithm.

3.1 TEXT CATEGORIZATION

In his survey Fabrizio Sebastiani (2001) reports how a text categorization process can be performed automatically by using the machine learning techniques. He defines text categorization and its applications. He also makes a summary of preprocessing steps involved in this process. In this chapter we will benefit from the study written by Fabrizio Sebastiani and some other workings in text categorization literature.

Sebastiani defines text categorization as follows: "Text categorization is the task of assigning a Boolean value to each pair $\langle d_j, c_i \rangle \in DxC$, where D is a domain of documents and $C = \{c_1, c_2, c_3, ..., c_{|c|}\}$ is a set of predefined *categories*." A false value F assigned to the pair $\langle d_j, c_i \rangle$ shows that d_j does not belong among c_i whereas a true value T shows that d_j belongs among c_i. Machine learning techniques generate and utilize a classifier function $\breve{\Phi} : \mathrm{D} \times \mathrm{C} \rightarrow \{\mathrm{T}, \mathrm{F}\}$ that decides how documents could be classified. This function estimates real results of the unknown target function $\Phi : \mathrm{D} \times \mathrm{C} \rightarrow \{\mathrm{T}, \mathrm{F}\}$. The measure of closeness in results is called the effectiveness of the classifier utilized. Performance measures can be considered by calculating the *precision* and *recall* values of results. Descriptions of both concepts will be covered later in this chapter.

According to the application different constraints can be applied in text categorization. In a multi-label categorization process a document $d_j \in D$ may be put into the zero or more categories while in a single-label categorization a document $d_j \in D$ should be located in exactly 1 category. *Binary* text categorization is a special type of single-label in which each $d_j \in D$ should be located in category $\mathrm{c_i}$ (i = 1,….,|c|) or in its complement $\overline{\mathrm{c}}_\mathrm{i}$. Because binary categorization is more general it can be used in applications instead of

multi-label categorization. Binary text categorization splits the classification problem under categories $C = \{c_1, c_2, c_3, ..., c_{|c|}\}$, into |C| independent smaller problems. Therefore there is a classifier function $\breve{\Phi}_i : D \times C \rightarrow \{T, F\}$ for each c_i. It applies each function sequentially to obtain boolean results T and F. Binary case solves a wide range of problems thus most of the machine learning algorithms support and include binary categorization.

In some critical applications automated text categorization is not desired. Final decision of assigning documents to categories should be done manually by human experts. Thus instead of full automation of text categorization a different task that ranks categories according to documents should be considered. This approach is called *ranking* categorization whereas the full automation of text categorization is called *hard* categorization. In ranking categorization, for a document $d_j \in D$ the function ranks the categories $C = \{c_1, c_2, c_3, ..., c_{|c|}\}$ according to their closeness with the document d_j. *Ranking* categorization is not covered in this study. We will make a *hard* categorization of web sites in Turkey.

3.2 APPLICATIONS OF TEXT CATEGORIZATION

Text categorization is used in a range of applications. We will briefly list the most important applications in this section. The applications covered here are document organization, text filtering, word sense disambiguation and hierarchical categorization of Web pages.

3.2.1 Document Organization

Many issues related to document organization can be addressed by text categorization techniques. For instance, at the offices of a newspaper incoming documents must be, prior to publication, categorized under categories such as Personals, Cars for Sale, Real Estate, etc [Sebastiani 2001]. It may be also used for grouping academic papers by topic in universities and research institutions.

3.2.2 Text Filtering

Text filtering is the activity of classifying a stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer [Belkin and Croft 1992]. A good example is a situation where the producer is a news agency and the consumer is a newspaper [Hayes et al. 1990]. Filtering system should block the documents the consumer is not interested in (e.g. all news not related with sports, in the case of a sports newspaper).

3.2.3 Word Sense Disambiguation

Word sense disambiguation (WSD) is the activity of finding the sense of an ambiguous (i.e. polysemous or homonymous) word in a given text. WSD may be seen as a text categorization task when we view word occurrence contexts as documents and word senses as categories.

3.2.4 Hierarchical Categorization of Web Pages

Classifying Web pages or web sites under the hierarchical categories is a recent interest that for automatic text classification. When Web documents are catalogued in this way, rather than issuing a query to a general-purpose Web search engine a searcher may find it easier to first navigate in the hierarchy of categories and then restrict her search to a particular category of interest [Sebastiani 2001]. Since the traditional manual categorization of the huge subsets of the Web is infeasible, automation of the classification task is needed by issuing machine learning techiniques. Sebastiani states two important peculiarities of automatic web page categorization as follows:

1. *The hypertextual nature of the documents*: links are a rich source of information, as they may be understood as stating the relevance of the linked page to the linking page.
2. *The hierarchical structure of the category set*: this may be used e.g. by decomposing the classification problem into a number of smaller classification

problems, each corresponding to a branching decision at an internal node. Techniques exploiting this intuition in a TC context have been presented in [Dumais and Chen 2000].

3.3 MACHINE LEARNING APPROACH

In the ’80s the most popular approach of automatic document classifiers was the manually builded *knowledge engineering* techniques, an expert system capable of taking text categorization decisions. An expert system consists of a set of manually defined logical rules for each category. CONSTRUE system is such a system built by Carnegie Group for the Reuters news agency. This system is described by an example from Sebastiani (2001) in Figure 3-1 :

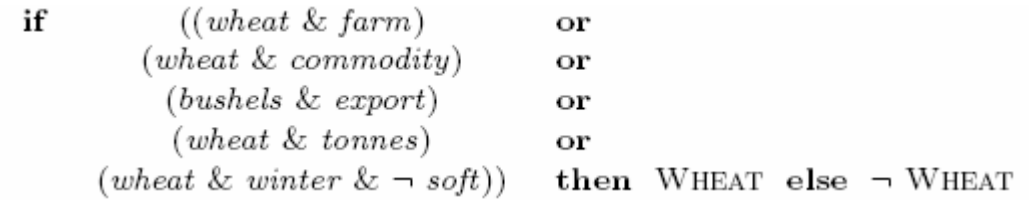

Figure 3-1: Rule-based classifier for the WHEAT category [Sebastiani 2001]

General form of the the formula is given below:

if ⟨*DNF formula*⟩ **then** ⟨*category*⟩

A DNF (disjunctive normal form) formula is a disjunction of conjunctive clauses; the document is classified under *“category”* if and only if it satisfies the formula.

Knowledge acquisition bottleneck is a problem in this approach in which the rules must be manually defined by a knowledge engineer with the help of a domain expert. If the set of categories is updated, then these two professionals must work togather again. In fact, when the classifier is tranferred to a completely different domain (set of categories) a different domain expert needs to be employed and the work has to be done from the beginning.

Machine learning approach to text categorization has been used since the early ’90s, and has become the dominant one, at least in the research groups. In this approach a general inductive process (called the *learner*) automatically builds a classifier for a category c_i by

observing the characteristics of a set of documents manually classified under c_i or c_i by a domain expert; from these characteristics, the inductive process gleans the characteristics that a new unseen document should have in order to be classified under c_i [Sebastiani 2001].

The advantages of the machine learning approach on the knowledge engineering approach are apparent. The engineering effort is not used in construction of such a classifier, rather, it is used in constructing automatic builder of classifiers (the *learner*). This means that once a learner is available, all that is needed is the inductive, *automatic* construction of a classifier from a set of manually classified documents. The learner program is a generic program that is not changed when original set of categories is updated, or if the classifier is ported to a completely different domain.

Classifiers are created for each category independently. For each category there must be a set of preclassified documents for construction of a classifier for each category. Therefore selection of the base document set is very important in machine learning. Even if there is no any manually classified documents available, machine learning is still a better choice than the knowledge engineering approach because it is easier to manually classify a set of documents than to create and test a full set of rules for categorization process.

3.3.1 Training Set & Test Set

The ML approach relies on the availability of an *initial corpus* $\Omega = \{d_1, \ldots, d_{|\Omega|}\} \subset D$ of documents preclassified under $C = \{c_1, \ldots, c_{|C|}\}$ [Sebastiani 2001]. This is the base initial set of documents that includes training and test sets. The initial corpus is divided into two subsets not necessarily be equal in size. While one subset is used for training documents, other subset is used as the test set.

In machine learning, construction of each classifier is binary, this means that for each category a classifier is constructed. A classifier determines whether incoming documents should be categorized under the category c_i or not. Training set and test sets, each contains same category labels $C = \{c_1, \ldots, c_{|C|}\}$ under which positive and negative document examples take place. A document d_j is a *positive example* of c_i if the result of

function $\Phi : D \times C \rightarrow \{T, F\}$ for the pair $< d_j, c_i > \in \Omega \times C$ is true (T), whereas it is a negative example when result is false (F).

The classifier for each category is inductively built by observing documents in training set. Test set is used for testing the effectiveness of the classifiers. If the classifier for category c_i, after testing, is not satisfactory to be generalized for remaining whole document set, then first the training set should be revised and be determined whether it is a good sample of the category c_i or not. The documents in test set cannot participate in the inductive construction of the classifiers; if this condition were not satisfied the experimental results obtained would be unrealistically good, and the evaluation would thus have no scientific character [Mitchell 1996]. This is called the *train-and-test* approach.

3.3.2 Document Indexing and Dimensionality Reduction

A classifier building algorithm cannot interpret texts directly. Because of this, an *indexing* procedure that maps a text d_j into a different representation is needed for training and test documents. There are two issues considered in this convertion:

1. Different ways to understand what a term is
2. Different ways to compute term weights

The most popular choice in defining a term is that, accepting individual words as terms. This is called the *bag of words* approach. When it comes to the second issue, there are a number of methods to compute term weights.

Weights usually range between 0 and 1 (non-binary). As a special case, binary weights may be used (1 denoting presence and 0 absence of the term). Binary or non-binary weights usage depends on the classifier learning algorithm used. Support Vector Machine algorithm uses non-binary style. A simple scaling on the data must be performed to avoid attributes in greater numeric ranges dominate those in smaller numeric ranges in non-binary form. Formulation of non-binary term weighting is included in Chapter 5.

3.3.3 Feature Selection Methods

Feature selection should be done before feature weighting because weighting is directly dependent on the features on the data set. There are four popular feature selection methods exist in the information retrieval domain. Next sub-sections define these methods.

After feature selection and weighting procedures data set is transformed to a new format that represents words as *attributes* and frequencies (or occurrences) of words as *values*. Namely, each text document will be converted to an attribute vector that contains *attribute:value* pairs. SVM requires that each document is represented as a vector of real numbers.

3.3.3.1 Information Gain (IG)

Information gain [Yang Pedersen, 1997] of a term measures the number of bits of information obtained for category prediction by the presence or absence of the term in a document. Let m be the number of classes. The information gain of a term t is defined as:

$$\begin{aligned} IG(t) = & -\sum_{i=1}^{m} p(c_i) \log p(c_i) \\ & + p(t) \sum_{i=1}^{m} p(c_i \mid t) \log p(c_i \mid t) \\ & + p(\bar{t}) \sum_{i=1}^{m} p(c_i \mid \bar{t}) \log p(c_i \mid \bar{t}) \end{aligned}$$

where *c* ranges over the classes (where a document might belong to several categories and we treat each category as an individual two-class problem, *c* can be either *positive* or *negative*). It measures how much we learn about *c* by knowing *t*, hence the term is called "information gain".

3.3.3.2 Chi-square Statistic (CHI)

The statistic measures the association between the term and the category [Yang Pedersen, 1997]. It is defined to be :

$$\chi^2(t,c) = \frac{N \times (p(t,c) \times p(\overline{t},\overline{c}) - p(t,\overline{c}) \times p(\overline{t},c))^2}{p(t) \times p(\overline{t}) \times p(c) \times p(\overline{c})}$$

$$\chi^2(t) = \underset{i=1}{\overset{m}{avg}} \{\chi^2(t,c_i)\}$$

The first equation procudes the chi-square of each term for every category. However a general value for each term is needed in the whole dataset. Therefore in the second equation average value of all results produced in the first equation is calculated for every term in the dataset. *m* is the number of categories and *i* is the sequence number between 1 and *m*.

3.3.3.3 Document Frequency (DF)

Document frequency is the number of documents in which a term occurs in a dataset. It is the simplest criterion for term selection and easily scales to a large dataset. It is a simple but effective feature selection method for text categorization [Yang, Pedersen 1997]. We used Document Frequency method in this study.

3.3.3.4 Term Strength (TS)

Term strength is computed based on the conditional probability that a term occurs in the second half of a pair of related documents given that it occurs in the first half [Yang, Pedersen 1997]:

$$TS(t) = p(t \in d_j \mid t \in d_i), d_i, d_j \in D \bigcap sim(d_i, d_j) > \beta$$

where d_i and d_j are similar documents, D is the document set and β is the parameter to determine the related pairs. Since we need to calculate the similarity for each document pair, the time complexity of TS is quadratic to the number of documents.

Chapter 4

SUPPORT VECTOR MACHINES

Support Vector Machines (SVM's) are a new learning method used for binary classication. It tries to find a hyperplane which separates the d-dimensional data into its two classes. SVM's are well-founded, and have shown to be practically successful. SVM's have also been extended to solve regression tasks (where the system is trained to output a numerical value, rather than yes or no in classification). Support Vector Machines were introduced by Vladimir Vapnik and colleagues. The earliest mention was in [Vapnik, 1979], but the first main paper is published in [Vapnik, 1995].

Support Vector Machines (SVM) solves a binary classification problem (+ vs. -) in two dimensions. The hyperplane h* separates positive and negative training examples with maximum margin δ. The examples closest to the hyperplane are called support vectors (marked with circles). Figure 4-1 demonstrates the SVM hyperplane.

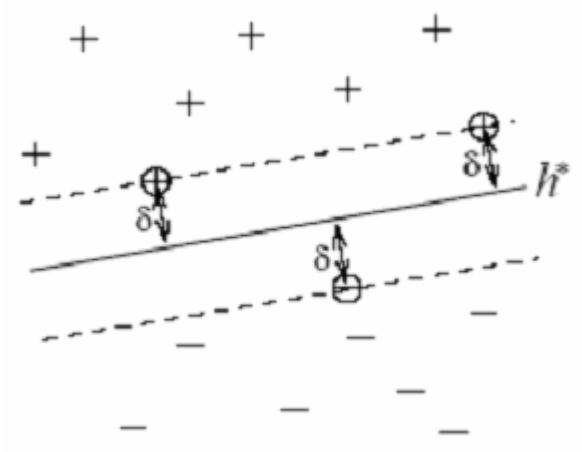

Figure 4-1: SVM hyperplane [Joachims 2001]

We are given *l* training examples {xi; yi} i = 1,.....*l* , where each example has d inputs and a class label with one of two values ($y_i \in \{-1,1\}$). Now, all hyperplanes in R^d are parameterized by a vector (w), and a constant (b), expressed in the equation

$$w \cdot x + b = 0 \quad (4\text{-}1)$$

(Recall that w is in fact the vector orthogonal to the hyperplane.) Given such a hyperplane (w,b) that separates the data, this gives the function

$$f(x) = \text{sign}(w \cdot x + b) \quad (4\text{-}2)$$

which correctly classifies the training data (and other testing data it hasn't seen yet). However, a given hyperplane represented by (w,b) is equally expressed by all pairs $\{\lambda w, \lambda b\}$ for $\lambda \in R^+$. So we define the canonical hyperplane to be that which separates the data from the hyperplane by a "distance" of at least 1. That is, we consider those that satisfy:

$$x_i \cdot w + b \geq +1 \quad \text{when} \quad y_i = +1 \quad (4\text{-}3)$$

$$x_i \cdot w + b \leq -1 \quad \text{when} \quad y_i = -1 \quad (4\text{-}4)$$

or more compactly:

$$y_i(x_i \cdot w + b) \geq 1 \qquad \forall i \quad (4\text{-}5)$$

All such hyperplanes have a "functional distance" ≥ 1 (quite literally, the function's value is ≥ 1). This shouldn't be confused with the "geometric" or "Euclidean distance" (also known as the margin). For a given hyperplane (w,b), all pairs $\{\lambda w, \lambda b\}$ define the exact same hyperplane, but each has a different functional distance to a given data point. To obtain the geometric distance from the hyperplane to a data point, we must normalize by the magnitude of w. This distance is simply:

$$d\big((w,b),\ x_i\big) = \frac{y_i(x_i \cdot w + b)}{\| w \|} \geq \frac{1}{\| w \|} \quad (4\text{-}6)$$

Intuitively, we want the hyperplane that maximizes the geometric distance to the closest data points. Figure 4-2 shows hyperplane selection.

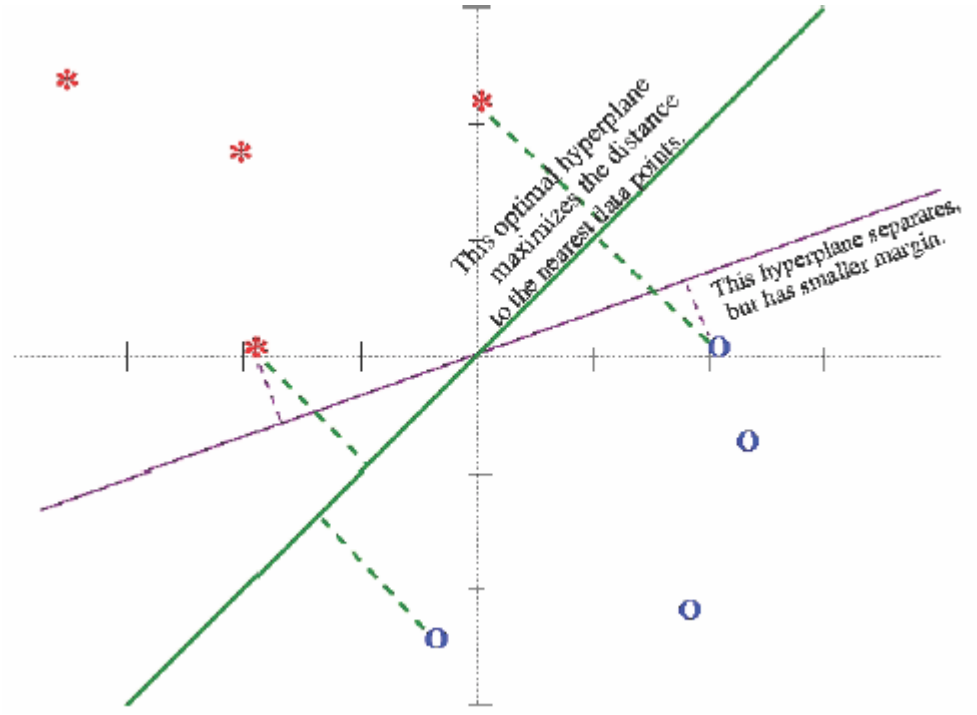

Figure 4-2: Choosing the hyperplane that maximizes the margin [Boswell, 2002]

From the equation we see this is accomplished by minimizing $\|w\|$ (subject to the distance constraints). The main method of doing this is with Lagrange multipliers. (See (Vapnik, 1995), or (Burges, 1998) for derivation details.) The problem is eventually transformed into following formula where α is the vector of l non-negative Lagrange multipliers to be determined, and C is a constant:

$$\begin{aligned} \text{minimize:} \quad & W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \tfrac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (\mathrm{x_i} \cdot \mathrm{x_j}) \\ \text{subject to:} \quad & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & 0 \le \alpha_i \le C \quad (\forall i) \end{aligned}$$

Two common methods in SVM exist. These are transductive and inductive. Transductive Support Vector Machines has advantages then inductive SVM in text categorization process. The following two figures summarize the difference between transductive and inductive learning.

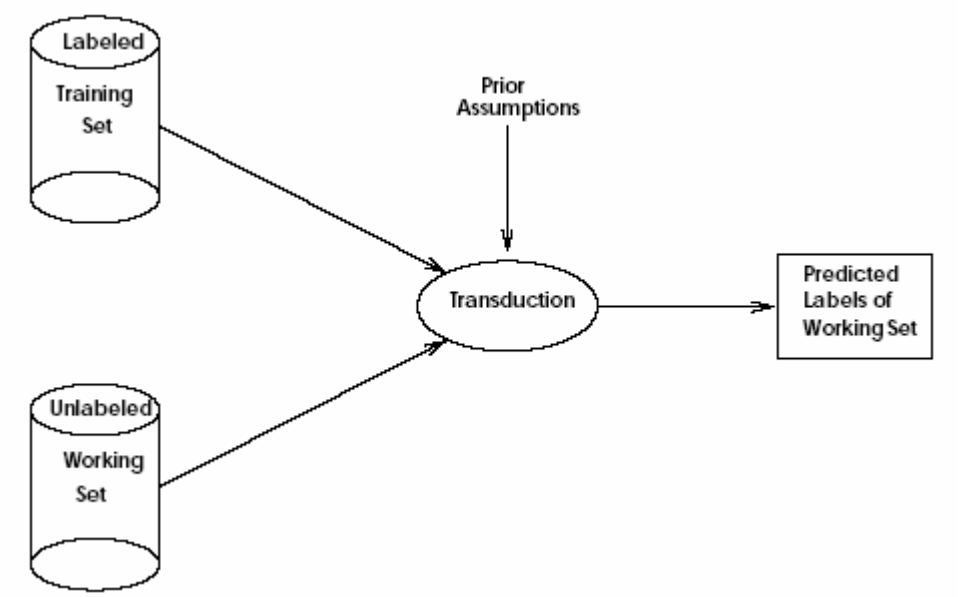

Figure 4-3: Transductive Learning

The text classification task is characterized by a special set of properties. They are independent of whether text classification is used for information filtering relevance feedback or for assigning semantic categories to news articles [Joachims 1999]. In such applications there are lots of unlabeled documents whereas a small number of training documents exists. When transductive approach is applied every unlabeled document orderly included in the learning process. While learning process, unlabeled documents are also classified getting the a negative or positive value if classification mode is active.

Performance of SVM is evident by many studies performed all over the world. In a study from Y. Yang and X. Liu (1999), five automatic text categorization methods, Support Vector Machines (SVM), a k-Nearest Neighbor (kNN) classifier, a neural network (NNet) approach, the Linear Least-squares Fit (LLSF) mapping and a Naive Bayes (NB) classifier were examined. For the performance on category assignments, both a sign test and an error-

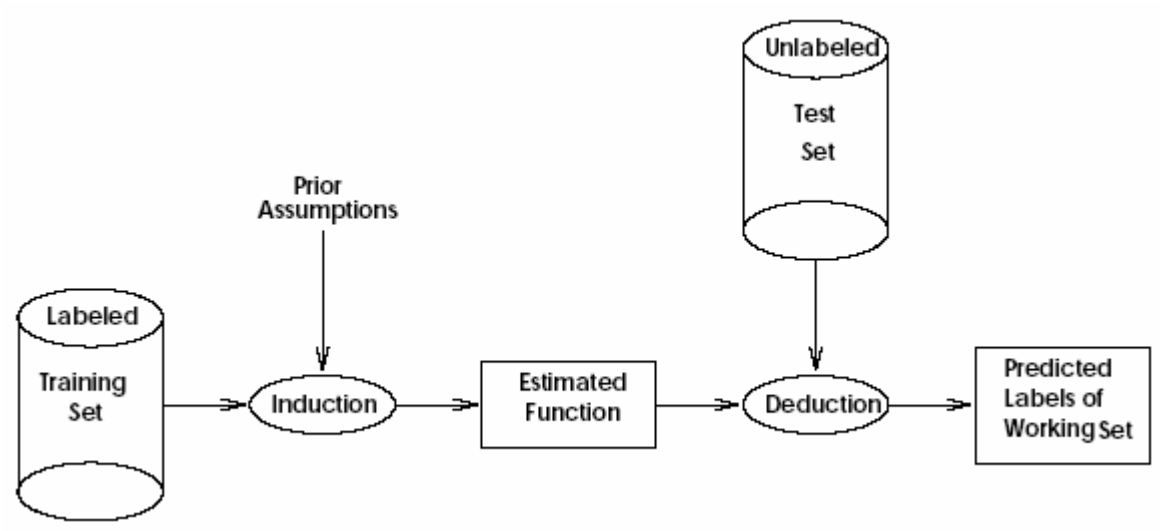

Figure 4-4: Inductive Learning

based proportion test suggest that SVM and kNN significantly outperform the other classifiers.

Chapter 5

CATEGORIZATION OF WEB SITES IN TURKEY

Categorization of web sites in Turkey is a multi-step study that categorizes the web sites of which domain names end up with *".tr"* by using a machine learning approach. This chapter covers all tasks performed from beginning of the study. A supervised learning technique is used in classification process. The main requirement of *supervised learning* approach is that, there should be an initial precategorized set of documents for each category in order to classify remaining documents. Categories also should be defined before any other task in this approach. Therefore, as an initial work, we decided the categories that would be used in the study. Predetermined categories are selected from web's popular directory Yahoo. Selected subset of Yahoo's top categories is listed below:

1. Business & Economy (Finance, Organizations etc.)
2. Education (college and universities)
3. Computers & Internet (www, software, games, hardware etc.)
4. Commercial (corporate web sites)
5. News & Media (newspapers, tv, radio etc.)
6. Entertainment (movies, music, cinema etc.)
7. Recreation & Sports (sports, travel, outdoor, indoor, autos etc.)
8. Health (diseases, drugs, fitness etc.)
9. Government (elections, military, law, taxes, municipalities etc.)
10. Society & Culture (people, environment etc.)
11. Reference (dictionaries, indexes etc.)
12. Online Shopping

There are over 50,000 web sites registered to the public DNS servers of Middle East Technical University (METU). Our main topic, as mentioned earlier, is classifying web sites in Turkey into 12 predefined thematic categories. We started with collecting contents of these web sites first, then parsed each web content to clear any html tag, stop words, punctuation marks etc. All contents were carried to a database for later steps. After a number of preprocessing operations such as stemming, feature selection, feature weighting performed on the data, the data is converted to a numeric format that represents web

documents as a vector of attribute : value pairs. Next step was to create a preclassified set of documents for each category manually, perior to classification process. Then, SVM^{light} is employed whenever all documents are ready to be processed by the software.

In addition to this, we gathered several important information about web sites such as type of plug-in contents (use of flash, applet, activeX), html design methods (frames vs. non-frame), metatag usage, title statistics and link information.

Support Vector Machine relies on good quality texts, especially for training. Unlike well known collections like Routers, the web pages are not homogeneous and consistent. Moreover web page content is usually established on images and plug-ins not based on text. Besides, metadata tags that contain valuable information about page content are not included in most web pages. Before starting the classification study we should be aware of the limitations and difficulties of dealing with web content during web site classification. Since the classification itself is text based, it is important to know the amount and quality of text based features that typically appear in web sites.

Following sections focus on the details of tasks which are shortly touched on above.

5.1 STRUCTURE OF THE DATABASE

The thesis project has had many steps that need different representations of data and so storage requirements. While we used file system structure for raw data, log files, temporary files, training sets, test sets and results, we employ Microsoft's MS Access database to store, query and manipulate structural data. The database, *master.mdb,* includes 10 tables, several queries, forms and a Visual Basic module for accomplishing several tasks during the study. Structure of each table in our database is described below.

"DOMAINLIST table": Created for storing the web sites' domain names, downloaded web document's size in byte and availability status of the web sites. It is employed during data collection. Fields of the table are showned in Table 5-1. The NAME, LEVEL1, LEVEL2 fields constitute domain name of the web site. For example, when storing the domain name "iyte.edu.tr", "iyte" is stored in NAME, "edu" is stored in LEVEL1 and "tr" is stored in LEVEL2 fields. AVAILABLE field describes the status of web site in terms of availability.

Table 5-1: domainlist table fields

Field Name	Data Type	Description
ID	AutoNumber	Primary key, id of the site
NAME	Text	domain name portion-1
LEVEL1	Text	domain name portion-2
LEVEL2	Text	domain name portion-3
AVAILABLE	Number	Availability
SIZE	Number	size in bytes

Figure 5-1 shows number of the sites that are ready to manipulate and other sites that are either not found or reserved for future use. Reserved sites are recognized by looking home pages. Every reserved site has a home page of unique content and same document size (3883 bytes).

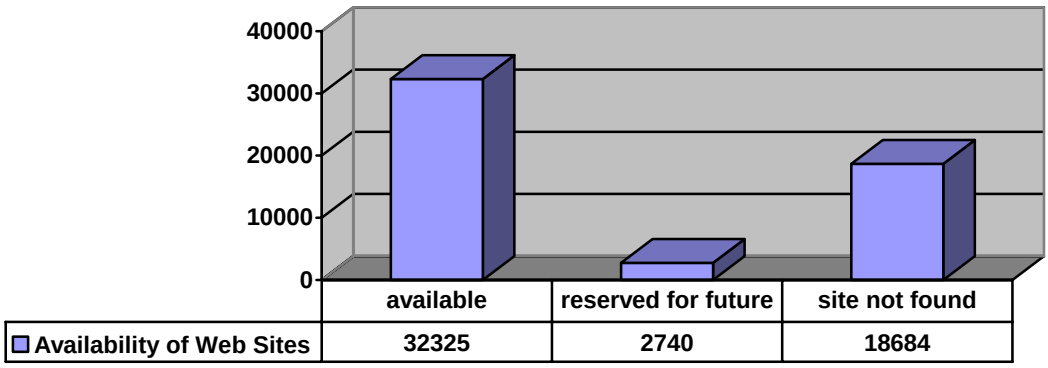

Figure 5-1: Status of Web Sites

"HTML_FRAME table": Created for storing the name and link information of target frames in a framed home page. Table 5-2 defines field names of table HTML_FRAME. In order to download framed home page contents we looked at this table from *SaveURL* program.

"HTML_TITLE table": Created for storing the html page title information. It is used as a resource for statistical analysis of web sites. It is not included in actual work of categorization. It gives an idea about title employment in web sites of *.tr* domain. DOMAIN_ID, TITLE, WORDCOUNT are fields of this table.

Table 5-2: Table structure of HTML_FRAME

Field Name	Data Type	Description
DOMAIN_ID	Number	Primary key, id of the site
NO	Number	Primary key, frame id
FRAME_NAME	Text	Name of the frame written in frameset tag
FRAME_SRC	Text	A href, destination address of the frame
TYPE	Number	internal vs. external frame (0:internal;1:external)

"HTML_LINK table": Created for storing the html link information obtained from downloaded home page. It is used as a resource for statistical analysis of web sites. It can also be useful for future work on web categorization by using anchor text that surrounds the link information rather than the body text. It is not included in actual work of categorization. Table 5-3 shows the name, data type and description of fields. TYPE field entries are composed of two logical values. First logical value is the type of link based on address. A link is either an internal link that refers to a page just in the same web site or an external link points to a different web site's page. Other logical value is obtained from the decision whether it is an image link or a text based "a href" link. Link code descriptions are listed below: Internal link:1, External link:2 Image:4 Href(text):8. Possible values of TYPE field are 1+4=5, 1+8=9, 2+4=6, 2+8=10. Besides 105,106,109,110 are for javascript created links.

Table 5-3: Table structure of HTML_LINK

Field Name	Data Type	Description
DOMAIN_ID	Number	Primary key, id of the site
NO	Number	Up to 255 links is collected per site
LINK	Text	Address of the destination page
LINK_TEXT	Text	Anchor text of the link
TYPE	Number	Values(5,6,9,10)

"META_INIT table": Created for storing the html meta information obtained from downloaded home page. It is used as a resource for statistical analysis of web sites. It can also be useful for future work on web categorization by using meta keywords as an alternative to the body text. Table 5-4 shows the details about META_INIT table. NO defines the metatag id. It has a default value of 0 and incremented for each metatag encountered in

the page. META stores the tag content. When a tag content is greater than 255 characters PARTITION is incremented and additional record is created for remaining content. It is incremented until remaining content is not greater than 255.

For statistical and future workings we also stored link, title and metatag related information that referred frame pages include. META_INIT_F, HTML_LINK_F, HTML_TITLE_F tables contain metatag, link and title knowledge in frames, respectively.

Table 5-4: Table structure of META_INIT

Field Name	Data Type	Description
DOMAIN_ID	Number	PK, id of the site
NO	Number	PK, Meta id, sites may contain more than 1 meta tag
META	Text	Meta tag contents
PARTITION	Number	PK, default:0, incremented when meta > 255 characters.

"DOCUMENTVECTOR table": Collected web sites are, after parsing and removal of all html tags and punctuation marks, read into this table from text files. It is used as the base table that represents all of the data collection before and during stop-word clearing operations. It contains records for distinct words that occur in a web document. It has no any PK constraint. FEATURE stores a single word. The number of occurrences is stored in VAL field. *Documentvector* is the initial table that represents the collection in a structural way. Its structure is displayed in Table 5-5 below. Mission of this table is completed after removal of stop-words (conjunctions, prepositions etc.). DOCUMENTVECTOR_STEMMED table substitutes for DOCUMENTVECTOR table. DOCUMENTVECTOR_STEMMED has an additional field named FLAG which was used in and feature stemming process.

Table 5-5: Table structure of DOCUMENTVECTOR

Field Name	Data Type	Description
DOMAIN_ID	Number	id of the site
FEATURE	Text	distinct word
VAL	Number	Number of occurrances of FEATURE in DOMAIN_ID

"**STOPWORDS table":** Stores stop-word list that is used with a query to delete the words from DOCUMENTVECTOR table. It as one field called FEATURE of text data type. We found 191 stop-words and removed all of them from data collection.

"**SOZLUK table":** Stores a dictionary in its single field. It contains nearly 43000 word stems. We used this table in stemming process. It as one field called FEATURE of text data type.

"**STEMMED_FINAL table":** This table is utilized for two reasons. First, text based feature representation should be converted to numeric representation after stemming process. Secondly, additional fields are defined for feature weighting which is the last process before generating input files of SVM. The table structure is shown in Table 5-6.

Table 5-6: Table structure of STEMMED_FINAL

Field Name	Data Type	Description
DOMAIN_ID	Number	PK, id of the site
FEATURE_ID	Number	PK, id of the feature
VAL	Number	Number of occurrances of FEATURE_ID in DOMAIN_ID (val>0)
TFTKDJ	Number	1 + log(VAL)
TRTK	Number	Number of documents in which feature_id occurs at least once
TFIDFTKDJ	Number	TFTKDJ*LOG(20225/TRTK)
WKJ	Number	Normalized weights obtained from tfidftkdj

Its data is collected from DOCUMENTVECTOR_STEMMED, FEATURE and TRTK tables by issuing a joined query. TJTKDJ, TRTK, TFIDFTKDJ, WKJ fields take part in weighting of features. We will discuss the feature weighting process later in this chapter.

"**FEATURE table":** We defined FEATURE (id as number; feature as text) table to give a distinct number to every feature in the collection and represent features by integer numbers.

"**TRTK table":** TRTK (feature as text; val as number) table stores number of documents in which feature occurs at least once. These values are copied to TRTK field of STEMMED_FINAL table.

"TRAINING_SET table": Contains training document id's for all categories. A column is defined for each category. With STEMMED_FINAL table, is used in construction of training input files.

5.2 COLLECTING THE DATA SET

We communicated with authorized members of *Internet Committee* which works under *Ministry of Communications* for getting the whole list of domain names in Turkey. After receiving the list of 53,172 web site domain names we transmit the list to DOMAINLIST table in MS Access database from text file for use in data collection procedure.

We implemented a multithreaded program with Java that is responsible for saving home page contents. The *SaveURL* program takes the site's domain name from the DOMAINLIST table and attempts to connect the URL. Once the connection is successfully established, the web site's home page is downloaded and saved to local disk. To do so, the program creates a file named with <*web-site-name*>.html on the disk and writes all source code of the home page to the file object. If downloaded file's text content is not adequate for text mining because of "*html frame*" usage then the *SaveURL* is later employed for downloading frame file contents by following links that inside the framesets of html frames.

In order to determine the available web sites, we examined all downloaded files. We grouped sites into three categories namely "available","reserved for future" and "not found". We assigned codes in the DOMAINLIST database to distinguish these groups. After this step, we went on our study ignoring web sites other than "available" web sites.

5.3 PARSING HTML DOCUMENTS

Data collection process is completed with a file on the disk for each web site. Because SVM, as other supervised learning approaches, needs relevant text without any noisy content, now, we should analyze every file syntactically to remove any irrelevant symbols, html tags, javascript codes etc.

To filter the raw data first the original document corpora have been parsed and the terms that will be used as the attributes were selected. Spaces and punctuation were used as delimiters for detecting the terms. All terms found were converted to lower-case.

We developed a parser for this task with Java. The parser performs lots of work in a single run. Flow of this procedure is shortly described as follows:

- It reads the initial <*web-site-name*>.html file for every available web site.
- Locates on "<metatag…>" part of the file and writes all tags to META_INIT table.
- Locates on "<title…>" part of the file and writes title content to HTML_TITLE table.
- Locates on "links" on the page, iterate on "links", determines each link's type, and writes gathered information to HTML_LINK table.
- Detects frames by looking for <frame…>" tags, parse the tags, determines each frame name, type and address, writes gathered information to HTML_FRAME table.
- Excludes any content that is part of the html source until only the natural language text remains.
- Creates a new file for every group of 50 web sites to store natural text specified in previous step. Figure 5-2 shows a part from file created.

```
SITE-ID:46119
SITE-DOMAIN:adana-bld.gov.tr
kutuları
kullanarak
belediye
otobüslerimizin
güzergahlarını
öğrenebilirsiniz
detaylı
yapmak
yardımı
güzergah
adı
.......
SITE-TOKENS:48
**********
SITE-ID:46120
SITE-DOMAIN:adana-aski.gov.tr
haberler
pazar
.......
```

Figure 5-2: A part from file created by parser program

At this point, data is ready to be ported to MS Access database for applying preprocessing tasks on data. We implemented a procedure to transform data from text file

to DOCUMENTVECTOR table in the master.mdb database. We did not include any other content other than Turkish while transmitting the data because of performance issues.

5.4 PREPROCESSING OF DATA SET

In order to transform a document into a feature vector, preprocessing is needed. This includes stop-word removal, stemming, feature selection, and feature weighting calculations.

Feature selection has been studied by [Yang, 1999], where information gain and chi-square methods are found most effective methods. Term selection based on document frequency is simple but has similar performance to information gain and chi-square methods. We benefit from document frequency for term selection process.

5.4.1 Removal of Stop-words

There are lots of words in documents that occur very often and are so general that their usage does not effect the text categorization. In fact, if these words are not removed effectiveness of categorization process will be reduced. The goal of stop-word removal process is to decide which words can be thought as stop-word, make a list of them and remove all stop-words that occur in the dataset. For this purpose a stop-word list containing the most common stop-words in Turkish has been used. The list was entered into the STOPWORDS table. A simple operation was done to remove these words from the dataset.

5.4.2 Stemming of Data Set

Stemming is the activity that groups words that share the same stem. The main goal of this process is to reduce diversity of words in common. “The word stem is derived from the occurrence form of a word by removing case and affix information [Porter, 1980]. For example “okul", “okulda" and “okullar" are all mapped to the same stem “okul".

We used a Turkish dictionary which contains 42989 distinct words for stemming operation. We implement a Visual Basic module in MS Access database that reads each

item the dictionary in descending order and replaces the words in the DOCUMENTVECTOR_STEMMED table that begin with this item.

5.4.3 Feature Selection

There are a number of feature selection strategies in text categorization task. To facilitate efficient and effective feature selection we used a simple Document-Frequency criterion for selecting features to use in categorization. This is very efficient, and has been shown [Yang, 1999] to be competitive with more sophisticated methods. We included some critical issues from Yang's study, comparison of feature selection methods, in the Chapter 3. Document frequency of a feature is the number of documents in which that feature takes place at least once. In automatic text categorization, features that have frequency of 3 or less treated as too specific to use in categorization process. Therefore we eliminate those features that match this criterion. To do so, we created a query which deletes related rows in STEMMED_FINAL table.

By applying feature selection on dataset, preprocessing of the data is completed. The average number of words per document is 55 after stop-word removal, stemming and feature selection.

There are 1117752 words, 22253 of which are unique in the dataset after preprocessing operations.

5.4.4 Feature Weighting

Selected features must be associated with a numerical weighted value to be ready for classification. Most types of feature weighting methods in text categorization are taken from the field of information retrieval. The most frequently used weight is TFIDF [Salton and Buckley, 1988]. The original TFIDF is displayed in Eq. 5-1.

There are three assumptions that appear in all weighting methods:

(i) "rare terms are no less important than frequent terms" (the (inverse document frequency, *IDF, assumption*);

(ii) "multiple appearances of a term in a document are no less important than single appearances" (the term frequency, *TF, assumption*);

(iii) (iii) "for the same quantity of term matching, long documents are no more important than short documents" (the *normalization assumption*).

These assumptions are well exemplified by the *tfidf* function [Debole,Sebastiani 2002].

$\#T_r(t_k)$ is the number of documents in Tr (training set) in which t_k occurs at least once. $\#(t_k,d_j)$ is the number of occurrances of term t_k in document d_j.

$tf(t_k, d_j)$ which is defined in Eq. 5-2 enforces the TF assumption.

$log\frac{|T_r|}{\#T_r(t_k)}$ component of Eq.5-1 enforces IDF assumption [Debole, Sebastiani 2002] where the expression $|T_r|$ is the total number of documents in training set.

The result of Eq. 5-1, *tfidf,* stands for *"term frequency" * "inverse document frequency"*

$$tfidf(t_k,d_j) = tf(t_k,d_j) \cdot \log \frac{|T_r|}{\#T_r(t_k)} \qquad 5\text{-}1$$

$$tf(t_k, d_j) = \begin{cases} 1 + \log \#(t_k,d_j) & \text{if } \#(t_k,d_j) > 0 \\ 0 & \text{otherwise} \end{cases} \qquad 5\text{-}2$$

Weights obtained by Eq. 5-1 are usually normalized by cosine normalization that shown in Eq. 5-3 to enforce normalization assumption [Debole, Sebastiani 2002].

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} tfidf(t_s, d_j)^2}} \qquad 5\text{-}3$$

First, we calculated $\#(t_k,d_j)$ and $tf(t_k, d_j)$. Then, $\#T_r(t_k)$, number of documents that the term is occurred, is calculated for each word. Next, we assigned the result of Eq. 3-1 to the $tfidf(t_k,d_j)$. This result is a feature based weighting that may be different for each feature. Finally we normalized the results to be in the range of [0,1] by the cosine normalization formula that shown in Eq. 3-3.

5.4.5 Constructing Initial Corpus Set

Initial corpus set consists of 328 randomly selected web documents from the dataset. To facilitate a good selection we implemented a Java program module that generates random numbers. Java's random number generator is well suited to uniform distribution. Therefore we certainly trust the results as a good sample of the collection.
Later we manually categorized 328 documents into selected categories and training set and test set is formed for each category. While the *commercial* and *computer* categories are populated with 117 documents which constitutes one third of the initial corpus set, "*society & culture"* contains a total of 12 pages for both training and test sets. We used 210 documents for constructing the training set and remaining 118 documents as test set to validate the effectiveness of classifiers. All work is performed with unique document ids defined in the beginning of the project. After assigning documents to categories we inserted classified document ids to the TRAINING_SET table of which column names are equal to the given categories.

Table 5-7 shows the training and test sets of categories. Given values were used as positive examples of related categories. For each category, negative examples were selected from remaining portion of the initial corpus randomly. Number of positives and number of negatives were equal because of the automatic text classification input requirements. For example when constructing training set for *commercial* category, we used 40 positive examples that manually classified under *commercial* category and randomly selected 40 negative examples that classified under the other categories. Test set is not contributed in any way in the construction of training set and classifier. Otherwise test results would be mistaken.

Table 5-7: Initial corpus set, consists of training set and test set

Category	Training Set	Test Set	Total
Business & Economy	20	5	25
Commercial	40	27	67
Computers	30	20	50
Education	20	10	30
Entertainment	10	6	16
Government	20	8	28
Health	16	8	24
News & Media	15	10	25
References	10	5	15
Shopping	12	7	19
Society & Culture	6	6	12
Sports	11	6	17
Total	**210**	**118**	**328**

5.5 SUMMARY OF THE DATA SET

This section summarizes the statistics about data set used in this study. A distribution of documents according to distinct feature counts that they include is given below in Figure 5-3. Most of the documents contain less than 20 features. Actually it is an advantage to decrease the feature number in preprocessing phase of the study because long vectors of features would lead to a significant decrease in performance.

A summary of the data set used is also given in Table 5-8. As it is seen, after preprocessing steps, the feature space considerably reduced from 46,091 to 22,253 features. We eliminated the documents that included English words and had insufficient content for text categorization. As a result nearly 20,000 web documents left.

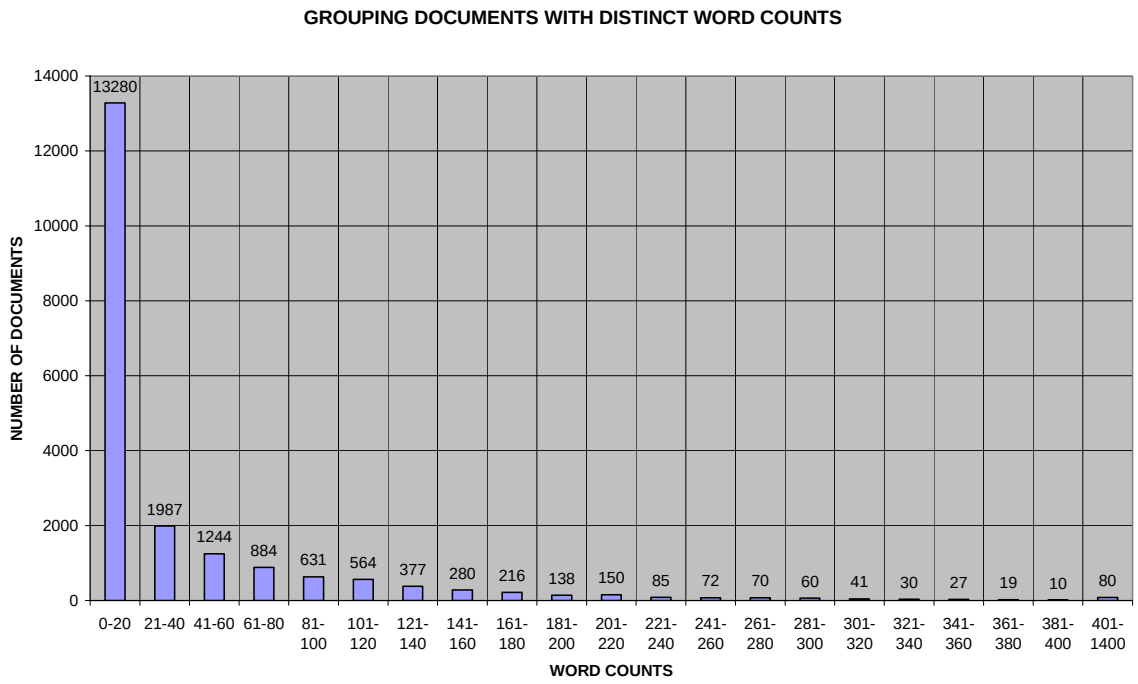

Figure 5-3: Distribution of documents depending on distinct word counts contained

Table 5-8: Data Set

Corpus Name	“.tr” web sites
Number of Documents	reduced from 53,172 to 20,225
Number of Categories	12

Number of Words Before Preprocessing	1,149,665
Number of Words After Preprocessing	1,117,752
Distinct Words Before Preprocessing	46,091
Distinct Words After Preprocessing	22,253

5.6 CATEGORIZING WEB SITES BY USING SVM*light* PACKAGE

This section explains how to use the SVM*light* software. Information is taken from SVM*light* 's corporate web site. SVM*light* package source code when compiled, creates the two executables:

```
svm_learn (learning module)
svm_classify (classification module)
```

SVM*light* consists of a learning module (`svm_learn`) and a classification module (`svm_classify`). The classification module can be used to apply the learned model to new examples.

```
svm_learn [options] example_file model_file
```

The input file `example_file` contains the training examples. The first lines may contain comments and are ignored if they start with #. Each of the following lines represents one training example and is of the following format:

<line> .=. <target> <feature>:<value> <feature>:<value> ... <feature>:<value>

<target> .=. +1 | -1 | 0 | <float>

```
<feature> .=. <integer> | "qid"
<value> .=. <float>
```

The target value and each of the feature/value pairs are separated by a space character. Feature/value pairs MUST be ordered by increasing feature number. Features with value zero can be skipped.

The target value denotes the class of the example. +1 as the target value marks a positive example, -1 a negative example respectively. So, for example, the line

```
-1 1:0.43 3:0.12 9284:0.2
```

specifies a negative example for which feature number 1 has the value 0.43, feature number 3 has the value 0.12, feature number 9284 has the value 0.2, and all the other features have value 0. A class (category) label of 0 indicates that this example should be classified using transduction. The predictions for the examples classified by transduction are written to the file specified through the -l option. The order of the predictions is the same as in the training data.

The result of `svm_learn` is the model which is learned from the training data in `example_file`. The model is written to `model_file`. To make predictions on test examples, `svm_classify` reads this file. `svm_classify` is called with the following parameters:

```
svm_classify [options] example_file model_file output_file
```

SVM^{light} supports both inductive and transductive learning from examples. Transductive learning is much better than inductive especially in cases where training data is small. Our training set is not large enough to use inductive approach. Therefore in this study we preferred to use transductive approach.

5.6.1 Creating Input Files

Now, we have document ids classified under given categories in the database. From now on, we should prepare the data set step by step to meet requirements of Support Vector Machine classifier learner. This process was quite complicated process that involves several steps. To clarify the final preparation steps before SVM usage, we described each step in a different sub-section.

5.6.1.1 Creating Training and Test Files

Training files were created sequentially by employing a Java procedure, TRAINING_SET table and STEMMED_FINAL table inside of the *master.mdb.* We created 12 *train.dat* file under certain directories. Initially all these files included just the positive samples read from database. Once training files created conforming to the input file formatting rules of SVM^{light} , we then, manually created every *test.dat* by cutting some of lines from each *train.dat* pasting to the corresponding *test.dat.* This manual process splits the initial corpus set into two subsets namely "*train.dat*"s and "*test.dat*"s. Now, test data is completely distinguished from train data.

At this point, we have positive examples for both train and test set. SVM^{light} requires in the train and test files, same number of negative examples following the positive ones. We had to include the negative samples to every test and train files. We assumed that each category is a mutually exclusive category. Therefore we randomly select positive examples of the other categories among *train.dat* files as negative examples to complete each category's *train.dat* file. The same thing was also applied to the *test.dat* files in test.
A partial fraction of *train.dat* from *health* category is shown in Figure 5-4.

5.6.1.2 Creating Unlabeled Entries for Remaining Documents

As we preferred to use transductive approach rather than inductive, we included unlabeled documents inside of each train.dat file. The final appearance of a train.dat file would consist of from top to bottom as:

- positive examples (lines starting with 1)
- negative examples (lines starting with -1)
- unlabeled examples (lines starting with 0)

At this step, we have no any unlabeled documents inside of train.dat files. In order to complete formation above, we employed a Java module that reads documents and feature weights from STEMMED_FINAL table with a criterion that documents read should not be in the TRAINING_SET table. A file named *alltest.dat* was created when this step is completed. This file includes only the unlabeled documents. In fact, our main purpose is to categorize these unlabeled documents under given categories.

```
#15 positive/15 negative examples in trainingset
1 666:0.11781962894866 967:0.11025309313764 1053:0.06355833659787 152
1 911:0.06562953440871 959:0.11510087620928 1003:0.06500871259649 106
1 119:0.11614011778081 349:0.13507560057053 362:0.04415607671648 412:
21979:0.09905747233003 21980:0.10063398770576 21981:0.12874276569971
.
..
-1 146:0.10128125308642 148:0.06416143592517 163:0.16468845889400 213
14737:0.13507560057053 14749:0.04775481356256 14753:0.10726407535547
-1 146:0.05981833962770 150:0.07741963524863 163:0.09726765681383 185
13541:0.10642954877238 13876:0.14691613700661 14216:0.09082726162139
-1 7072:0.05322742283509 8489:0.06767991999296 9002:0.03102055448612
```

Figure 5-4: A partial fraction from *train.dat* file of *HEALTH* category

5.6.1.3 Combining Labeled Entries with Unlabeled Entries

Labeled training files needed to be combined with unlabeled entries for transductive learning of classifiers. Because of the big size of the unlabeled entry file, *alltest.dat,* we utilized a Java module to merge each *train.dat* with *alltest.dat*. When all files are ready, we tried *svm_learn* module of SVM^{light} . After a waiting time of a whole day, we observed that the program is still running not producing any result. Thereafter, we splitted the unlabeled set of 19,897 documents into fixed sized pieces each of which contains 100 unlabeled documents. This operation breaks the problem into 199 smaller problems since 19897 ≅ 199x100. Then, by the help of a Java module, we appended labeled *train.dat* file of "*business & economy"* to the first lines of these 199 files. This action was repeated for every category in the study.

At this point we have small 199 training files for every category in the set. Every file contains some positive examples, some negative examples and 100 unlabeled documents. Figure 5-5 symbolizes the final situation in which each category has 199 files. The difference between three *train1.dat* is that, each file contains different documents in (1) and (-1) sections. On the other hand, the common feature of these files is that, they contain the same first group of 100 unlabeled documents in (0) section. Within a single category, all *trainX.dat (0<X<200)* files contain same documents in (1) and (-1). Files consecutively

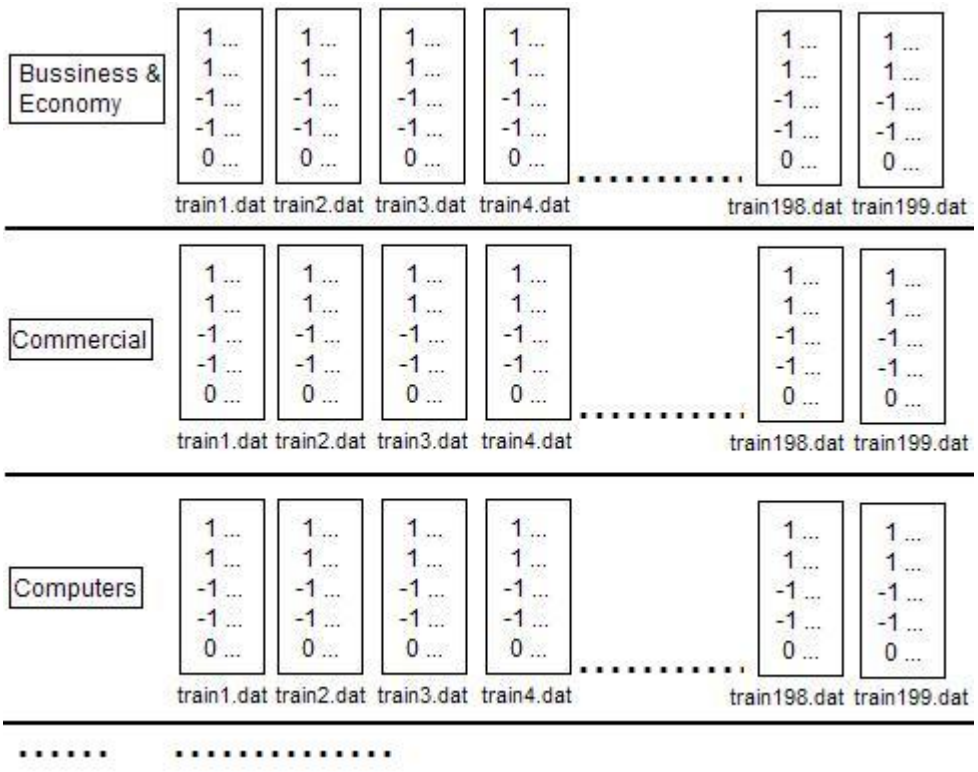

Figure 5-5: Symbolized display of splitted data set under categories.

store groups of 100 documents from first document to 19,897th document in the set.

5.6.2 Creating Batch Files

As described before, SVM^{light} software has two executables. One for learning classifier (svm_learn), other (svm_classify) for classifying documents utilizing the model file created by svm_learn. We had used *svm_learn* in the following way for a single category, say *computers*, before splitting the data set. Here, *train.dat* contains 30 positive, 30 negative and 19,897 unlabeled document entries. r*esult.dat* is output file in transductive mode which contains classifier results about unlabeled data. m*odel* file is the learned classifier that can be used in future without retraining the data.

```
svm_learn -l results/result.dat train.dat model/model
```

After splitting the data set we created 20 batch files each of which contains 10 commands in a form given above. The first batch file categorizes first 10 splitted *trainX.dat (1<X<10)*, second batch file categorizes 2nd 10 *trainX.dat (11<X<20)* files …… twentieth batch file categorizes 20th 10 *trainX.dat (191<X<199)* files. Batch files were given names of *svm_learnX.dat (0<X<21)*. A sample portion from *svm_learn1.bat* is given below:

```
svm_learn -l results/result1.dat train1.dat model/model1
svm_learn -l results/result2.dat train2.dat model/model2
svm_learn -l results/result3.dat train3.dat model/model3
….
svm_learn -l results/result10.dat train10.dat model/model10
```

After all batch files were executed successfully 199 result files and 199 model files were created. For readability concerns, classification results are converted to a simple binary form from original result formats after combining result files into a single *final* result file. For this conversion we again used Java to implement a program module. Figure 5-6 shows the original result file format (left side) and converted binary result format (right side). (+1) indicates a positive result, whereas (-1) refers to a negative result.

```
0.97438931:+1 -0.97438931:-1                    +1
0.96752834:-1 -0.96752834:+1                    -1
0.96373809:-1 -0.96373809:+1                    -1
0.7297526:-1 -0.7297526:+1                      -1
0.52661304:-1 -0.52661304:+1                    -1
0.77725443:+1 -0.77725443:-1                    +1
```

Figure 5-6:Original output of transductive svm_learn(left) and converted final output(right)

Till now, test data has not been participated in any step. We used *test.dat* file of a category to measure categorization performance of the classifier. We created *svm_classify_batch.bat* batch file for *svm_classify* command. A sample portion of *svm_classify_batch.bat* is given below:

```
svm_classify test/test.dat model/model1 output.dat
svm_classify test/test.dat model/model2 output.dat
svm_classify test/test.dat model/model3 output.dat
……..
svm_classify test/test.dat model/model199 output.dat
```

Since we employed transductive SVM we actually did not need to run *svm_classify* for classification task. The *transductive svm_learn* did the learning and classification in a single step. However *svm_classify* is still useful when we are looking for performance measures such as accuracy, precision and recall percentages. Besides of output.dat file, *svm_classify* command produces log entries about performance measures. Therefore we run *svm_classify_batch.bat > log/logtest.txt* program in command line and then parse the log file with a Java module for calculating average values for 199 accuracy, 199 precision and 199 recall values.

5.6.3 Running the Program

When we first ran the *svm_learn* on some categories we observed very low accuracy and precision. For example *society & culture* category had a 3% precision in first run. We revised training set used for this category and re-constructed it. This led to an impressive improvement on the classifier decisions. The precision increased to 50%. We saw that quality of the training data is undeniably important in automatic text categorization. Besides we tried some different kernel options like radial basis function and polynomial kernels but the improvement on the results was not clear. Therefore we finalized with linear kernel which is recommended in text categorization tasks.

5.6.4 Converting Results to a Readable Form

We sequentially read all result files of a category to a single file and then parse each line of the created merged file to capture (+1)'s and (-1)'s. Next, captured values were written to a new text file. The order of results is same as the order of unlabeled documents located in *train.dat* files. Both orders were tied to the document ids sorted by ascending order. Therefore, we collected document ids, and domain names from master.mdb in ascending order and results from the new text file into a MS Microsoft Excel sheet to analyze the final picture. Finally after collecting results of all categories, we export the excel sheet to master.mdb database for applying queries on results.

5.7 RESULTS

Contingency table shown in Table 5-9 defines the relationship between expert judgments and classifier judgments. Performance measures, namely *precision* and *recall*, of the classifier are derived from this table.

Table 5-9: Contingency Table

Category c_i		**Expert Judgments**	
		YES	**NO**
Classifier Judgments	**YES**	TP	FP
	NO	FN	TN

TP (true positives): Number of cases in which classifier assigned the category c_i to a document and the correct category is c_i

FP (false positives): Number of cases in which classifier assigned the category c_i to a document but the correct category is not c_i

FN (false negatives): Number of cases in which classifier did not assign the category c_i to a document but the correct category is c_i

TN (true negatives): Number of cases in which classifier did not assign the category c_i to a document and the correct category is not c_i

Two important measures in information retrieval literature, called recall and precision are used for measuring the categorization performance. Precision(c_i) is the conditional probability, that is, as the probability that if a random document d_j is classified under c_i , this decision is correct. Recall(c_i) is defined as the probability that, if a random document d_j ought to be classified under c_i, this decision is taken.

The formula for recall and precision is given below that uses information in the contingency table above:

```
Recall = TP / (TP + FN)
Precision = TP / (TP + FP)
```

5.7.1 Performance Values of Classifiers for Each Category

We run *svm_classify_batch.bat > log/logtest.txt* program in command line. This program utilizes our test.dat file for measuring the performance of the classifier. We parsed

the *logtest.txt* file with a Java module for calculating average values for 199 accuracy, 199 precision and 199 recall values.

Table 5-10: Performance values

Category	Accuracy	Precision	Recall
Business & Economy	68.64%	82.30%	51.15%
Commercial	52.30%	52.87%	41.61%
Computers	67.48%	73.01%	57.36%
Education	38.64%	36.92%	32.81%
Entertainment	64.07%	63.95%	66.83%
Government	69.81%	68.41%	74.12%
Health	64.07%	63.95%	66.83%
News & Media	73.46%	72.78%	76.03%
References	45.00%	46.57%	50.00%
Shopping	66.00%	69.57%	58.74%
Society & Culture	54.23%	50.25%	92.96%
Sports	65.70%	62.38%	84.00%
Average	**60.78%**	**61.91%**	**62.70%**

5.7.2 Categorization Results

Categorization of web sites ended with formating the results. We get following outputs after analyzing the results. Table 5-11 lists number of web sites according to number categories they assigned.

We included the number of web sites that categorized under 13 categories (1 for the *other* category) in Table 5-12.

Table 5-11: Number of web sites that assigned under different number of categories

number of documents	number of categories
7022	0 (no category)
3372	Only 1
2762	Two
1688	Three
1015	Four
493	Five
656	Six
971	Seven
1444	Eight
435	Nine
39	Ten
0	Eleven
0	Twelve
Total = 19897	

Table 5-12: General categorization results in respect of number of sites

Category	Number of Sites
Bussiness & Economy	3231
Commercial	2849
Computers	2731
Education	4153
Entertainment	4193
Government	3446
Health	3980
News & Media	3680
References	5710
Shopping	3682
Society & Culture	4312
Sports	5108
Other	7022

5.8 OTHER FINDINGS

In addition to the categorization task, we collected some useful statistics about web page design of .tr sub-domain web sites. The following table is a compact representation of meta-tag and title usage ratio statistics. There are 30878 web sites that utilizes a <TITLE> tag in their HTML source. Almost all web sites' titles contain word counts in interval of [1,10]. Moreover 7298 web sites include a <meta-description> tag, which briefly describes the web site, in their source code. In addition, 9367 sites have used <meta-keywords> tag which is meaningful for search engines that index the keywords for identifying a web page.

Another statistical analysis is about *HTML FRAMEs*. Findings of mining the information in the *master.mdb* base tables will be discussed in the following paragraphs.

4155 sites are designed with frames. Number of frames in a page is ranged from 1 to 23 with a mean value of 2. 4155 sites have used a total number of 10603 html frames 9772 of which are inside the same web server. Usage of 2 frames is the most popular way since it is preferred in 1745 web sites. 92% of frames are on the same web server with the home pages that refer to these frames. A small portion of the frames, nearly 8%, are outer

Table 5-13: Tag usage statistics

Tag Type	0 words	1-10 words	11-50 words	51+ words	Web Sites
Title	666 (2.2 %)	28886 (93.5%)	1270 (4.1%)	56 (0.2%)	30878 (100%)
Meta-Description	19 (0.2%)	4375 (60%)	2864 (39.3%)	40 (0.5%)	7298 (100%)
Meta-Keywords	18 (0.02%)	2380 (25.4%)	6837 (72.9%)	132 (1.6%)	9367 (100%)

frames that reside on a different web server. These frames are generally used for redirecting web address to a new location. Home pages generally use a single frame for this purpose. Figure 5-7 and Table 5-14 show details about html frames.

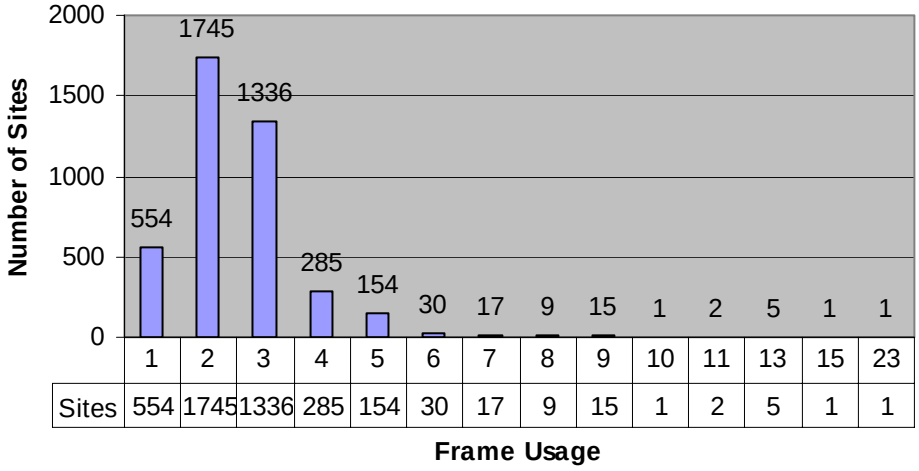

Figure 5-7: Number of frames in 4155 web sites' home pages that use frames

Table 5-14: HTML Frame Usage Information

Number of Frames	Web sites
1	554
2	1745
3	1336
4	285
5	154
6	30
7	17
8	9
9	15
10	1
11	2
13	5
15	1
23	1
Total Web Sites	4155

Median of the Frame Distribution
2

Frame Type	Count
inner (in the web server)	9772
outher (different web s.)	831
Total Frame Count	10603

Chapter 6

RESULTS AND CONCLUSION

The principal purpose of this study is to categorize web sites under .tr sub-domain name into predetermined categories taken from Yahoo by using a supervised machine learning approach with SVM. Thorsten Joachims' popular SVM^{light} program is chosen for this study. For the data set, we used home pages of web sites that registered to domain name servers of the Middle East Technical University (METU). From 53,172 web sites we could download home pages of 32,325. Then we analyzed all downloaded pages through several preprocessing steps before the actual categorization process. Eventually the set has reduced to a size of 20,225 pages. After tasks like tokenizing document terms, stop-word removal, stemming, feature selection and feature weighting, preprocessing of data set was completed. Critical issues such as construction of training and test data sets were performed. We constructed those sets from an initial corpus set of 328 documents. Unlabeled data was prepared according to requirements of SVM. When all needed processes finished we employed svm_learn and svm_classify modules of SVM^{light} for learning and testing issues respectively. We used inductive approach because of the generalization problem of inductive approach especially with a small number of traing set as in our study.

6.1 ANALYSIS OF RESULTS

In this study an automatic text categorization approach was selected for classification of web sites. Body texts of the home pages were considered as main source for automatic text classification process. We computed precision and recall performance measures for each 12 classifiers. We observed that "*business & economy"* has the highest precision (82%) and the *"society & economy"* has the highest recall (92%) value in the category set. The average precision over categories is 61.91%. In some categories like

"business & economy" and *computers* the precision value is somewhat satisfactory and so is the results. As we stated earlier, HTML documents may lack in homogenity and consistency issues. This nature of HTML documents would lead to a low precision in some categories like *education (36.92%)*. Although average precision is decreased because of these categories, classification results still give an idea about distribution of the web sites under 12 categories issued.

6.2 OTHER STATISTICAL INFERENCES

In addition to the classification task, we had aimed to collect useful information about web page designers' attitudes during page design. We stored frame, title, link and meta tag structures of web sites' home pages in the *master.mdb* and study on these data for converting to an important statistical information about web page design. Findings include:

- HTML <Title> tag usage, grouping the sites according to word counts in <title> tag
- Most useful meta tags, meta-description and meta-keyword tags, are collected and a similar grouping like in th <title> tag is performed.
- Utilized HTML FRAME statistics

6.3 CONCLUSION

To conclude, this study points out an automated classification of web pages and more generally refers to text classification by using a common machine learning algorithm , Support Vector Machine. Web site distribution is figured out under 12 categories and additional knowledge is discovered from web documents by the completion of this study.

In addition to any categorization results and important statistical inferences, hands on experience are accomplished in all steps of applying a machine learning approach to text classification. It would be beneficial for future studies on the classification problem of large document sets with machine learning approach.

6.4 FUTURE WORK

We dealt with too many information that lead to a complex work in some points of the study. All needed actions were performed depending on the needs at that time. Because of the existence of several complex preprocessing tasks, it was a long-termed job to prepare everything until employing the SVM^{light} software. This study does not use any easy to use framework for preprocessing tasks on the data set. In future studies a framework can be implemented that offer many features from term representation techniques, such as single word as a term or word phrases as a single term, to a number of feature selection and feature weighting methods. A generic framework can be designed to fully automate a wide range of text categorization tasks just in a single program.

Specifically for web page classification tasks a framework that decides what portion of web page to use in categorization process could be implemented as a future work. We used body text in this study, alternatively “meta tag” keywords or title or even anchor text that surrounds the hyperlink could be used as the data set. By existence of a framework, when data set is changed, all preprocessing tasks should be performed by the framework until issuing machine learning algorithm such as SVM.

In addition, information stored on web pages received from the internet for this study, would be used in different workings in future.

BIBLIOGRAPHY

[Barfourosh et al. 2002] A.Barfourosh, M.L.Anderson, H.R. Motahary Nezhad, D. Perlis, *Information Retrieval to the World Wide Web and Active Logic*, 2002

[Belkin and Croft 1992] N.J.Belkin and W.B.Croft, "Information filtering and information retrieval: two sides of the same coin?" *Commun. ACM 35*, 1992

[Broder et al. 2000] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, A. Rajagopalan, R. Stata, A. Tomkins and J. Wiener, "Graph structure in the web" *Proceedings of International World Wide Web Conference*, Amsterdam, 2000

[Cambazoğlu 2001] Berkant Barla Cambazoğlu, *Text categorization of Turkish documents*, Bilkent University, 2001

[Chakrabarti et al. 1999] S.Chakrabarti, M.V.Berg and B.Dom, "Focused crawling: A new approach to topic-specific web resource discovery", *Proceedings of International World Wide Web Conference*, pages 1623–1640, Montreal, 1999.

[Craven 2003] P.Craven, *Google's PageRank Explained*, 2003

[Dumais and Chen 2000] S.T.Dumais and H.Chen, "Hierarchical classification of Web content", *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval,* 256–263, 2000

[Etzioni 1996] Etzioni, "The World Wide Web: Quagmire or gold mine", *Communications of the ACM*, 39(11), 65–68, 1996

[Feldman and Dagan 1995] R. Feldman, I. Dagan, *Knowledge Discovery in textual databases*, pages 112-117, 1995

[FürnKranz 2002] J.FürnKranz, "Web Structure Mining Exploiting the Graph Structure of the Web", *Austrian Research Institute For AI*, 2002

[Greenberg and Garber 1999] I.Greenberg and L.Garber, "Searching for new search technologies", *IEEE computer*, 1999

[Hearst 1999] M.A.Hearst, *Untangling text data mining*, 1999

[Heydon and Najork] A.Heydon and M.Najork, "Mercator: A scalable, extensible web crawler", *World Wide Web*, pages 219–229, 1999.

[Jing et al. 2001] D.Jing, W.Yan,Zhou Xuan, *Web Structure Mining*, 2001

[Joachims 1999] T.Joachims, "Text categorization with support vector machines: Learning with many relevant features". *Tenth European Conference on Machine Learning ECML 98*, pages 137–142, 1999

[Joachims 2001] T.Joachims, "A Statistical Learning Model of Text Classification for Support Vector Machines", *SIGIR'01,* New Orleans, Louisiana, USA, 2001

[Keller 2000] Frank Keller, "Definition of Learning", *Introduction to Machine Learning,* 2000

[Kleinberg 1998] J. Kleinberg,"Authoritative sources in a hyperlinked environment", *Proceedings of the Ninth ACM-SIAM Symposium on Discrete Algorithms (SODA),* San Francisco, CA, 1998

[**Kleinberg and Lawrence 2001]** J. Kleinberg, S. Lawrence, "The structure of the Web", *Science's Compass*, 2001

[Kosala and Blockeel 2000] R. Kosala and H. Blockeel, "Web mining research: A survey", *SIGKDD Explorations*, 2:1–15, 2000

[Lawrence and Giles 1999] S.Lawrence and C.L.Giles, "Accessibility of information on the web", *Nature*, 400:107–109,1999

[Levene and A.Poulovassilis 2001] M.Levene and A.Poulovassilis, "The Size of the Web Graph", *Web Dynamics*, 2001

[Mitchell 1996] T.M.Mitchell, *Machine Learning.* McGraw Hill, New York, NY, 1996

[Page and Brin 1998] L.Page, S.Brin, "The anatomy of a large-scale hypertext web search engine", *Proceeding of the seventh International World Wide Web Conference*, 1998

[Page and Brin 1998a] L.Page, S.Brin, *The PageRank Citation Ranking: Bringing Order to the Web*, 1998

[Porter, 1980] M.Porter, *An algorithm for suffix stripping*, 1980

[Romanko 2002] O.Romanko, *Web Mining,* McMaster University, 2002

[Salton and Buckley, 1988] G.Salton and C.Buckley, "Term-weighting approaches in automatic text retrieval", *Inform. Process. Man. 24*, pages 513–523, 1988

[Sebastiani 2001] F.Sebastiani, *Machine Learning in Automated Text Categorization,* Italy, 2001

[Tan 1999] A.H.Tan, *Text mining: The state of art and the challenges*, pages 65-70, 1999

[Vapnik 1995] V. Vapnik, *The Nature of Statistical Learning Theory*, New York, 1995

[Yang and Liu 1999] Y.Yang and X.Liu, "A comparison of machine learning techniques in automated text categorization", *Proceedings of ICML-99, 14th International Conference on Machine Learning,* 1999

[Yang and Pedersen 1997] Y.Yang and J.Pedersen, "A comparative study on feature selection in text categorization", *Proceedings of ICML-97, 14th International Conference on Machine Learning,* 412-420, 1997

[Yuwono and Lee 1996] B.Yuwono and D.L.Lee, "WISE: A world wide web resource database system", *IEEE Transactions Knowledge and Database Engineering*, 1996

[Zaiane 1999] Osmar Rachid Zaiane, "Web Mining", *Resource and Knowledge Discovery from The Internet and Multimedia Repositories,* 1999

[Zaiane et al. 1998] Osmar Rachid Zaiane et al., "Multimediaminer: a system prototype for multimedia data mining", *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, page 581, 1998

APPENDIX

DETAIL ANALYSIS OF HITS AND PAGERANK ALGORITHMS

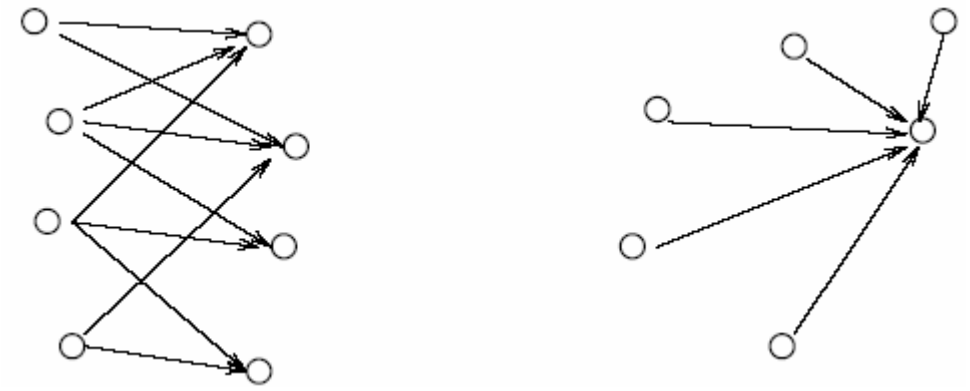

Figure A-1: A densely linked set of hubs and authorities

The algorithm consists of two main steps. The first step involves constructing a focused subgraph of the Web.

> Any collection of hyperlinked pages can be viewed as a directed graph $G = (V,E)$. From a graph G, we can isolate small regions, or *subgraphs*, in the following way. If $W \subseteq V$ is a subset of the pages, we use $G[W]$ to denote the graph *induced* on W: its nodes are the pages in W, and its edges correspond to all the links between pages in W. Given a broad-topic query, specified by a query string σ. Properties of subgraph S_σ are:
>
> (i) S_σ is relatively small.
> (ii) S_σ is rich in relevant pages.
> (iii) S_σ contains most of the strongest authorities.
>
> [Kleinberg 1998]

To construct the subgraph, Kleinberg first created the *root set* R_σ by collecting the t (set to about 200) highest-ranked pages for the query σ from a text-based search engine such as Altavista or Hotbot. Properties (i) and (ii) of subgraph S_σ were covered by this root set. However (iii) was not covered because links between pages contained by this root set were so few. Strongest authorities are pages that are pointed to by many good hubs. Therefore, R_σ to satisfy (iii), had to be expanded to S_σ. Kleinberg, then obtained S_σ by growing R_σ to

include any page pointed to by a page in R_σ and any page that points to a page in R_σ. Figure A-2 [Kleinberg 1998] shows the procedure of the expansion.

Subgraph(σ,$\mathcal{E}$,t,d)
  σ: a query string.
  $\mathcal{E}$: a text-based search engine.
  t, d: natural numbers.
  Let R_σ denote the top t results of $\mathcal{E}$ on σ.
  Set $S_\sigma := R_\sigma$
  For each page $p \in R_\sigma$
    Let $\Gamma^+(p)$ denote the set of all pages p points to.
    Let $\Gamma^-(p)$ denote the set of all pages pointing to p.
    Add all pages in $\Gamma^+(p)$ to S_σ.
    If $|\Gamma^-(p)| \leq d$ then
      Add all pages in $\Gamma^-(p)$ to S_σ.
    Else
      Add an arbitrary set of d pages from $\Gamma^-(p)$ to S_σ.
  End
  Return S_σ

Figure A-2: Procedure of expanding R_σ to S_σ

S_σ is the base set for σ. He used search engine Altavista, t = 200 and d = 50 for construction of the subgraph. Kleinberg also observed that by expanding the root set the size is generally rised up to the range 1000-5000 from initial set of t = 200.

The second step of HITS algorithm involves computing hubs and authorities. Hubs and authorities exhibit what could be called a *mutually reinforcing relationship*: a good *hub* is a page that points to many good authorities; a good *authority* is a page that is pointed to by many good hubs [Kleinberg 1998]. Kleinberg implemented an iterative algorithm that maintains and updates numerical weights for each page.

$x^{\langle p \rangle}$: non-negative *authority weight* for each page p

$y^{\langle p \rangle}$: non-negative *hub weight* for each page p

The goal of the algorithm is to view the pages with larger x and y-values as good authorities and hubs respectively. Weights of each type are normalized so their squares sum to 1:

$\sum_{p \in S_\sigma} (x^{\langle p \rangle})^2 = 1$ and $\sum_{p \in S_\sigma} (y^{\langle p \rangle})^2 = 1$.

Figure A-3 shows two basic operations that calculates weights: $\mathcal{I}$ operation which is displayed on left side of the figure, updates the *x*-weights and $\mathcal{O}$ operation that is symbolized on right side of figure, updates *y*-weights.

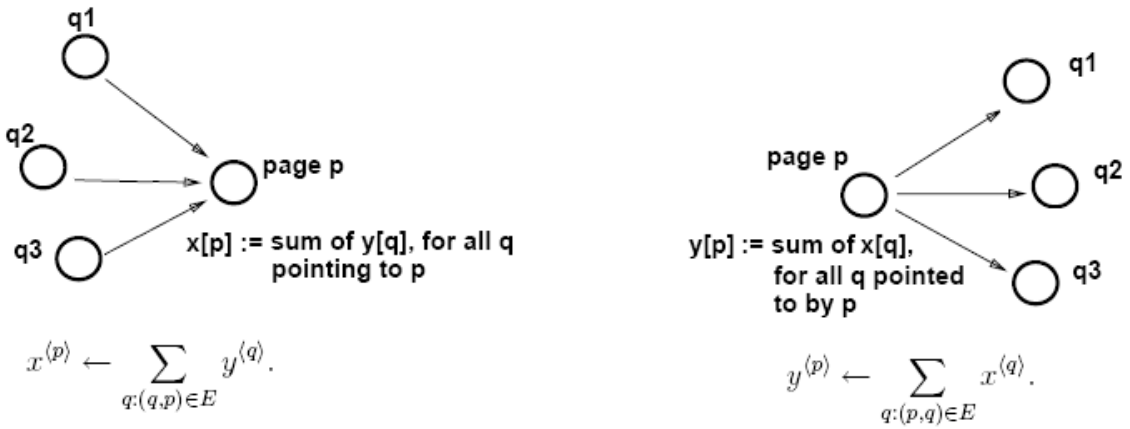

Figure A-3: The basic operations of $\mathcal{I}$ (left side) and $\mathcal{O}$ (right side)

These operations yield to a mutually reinforcing relationship between hubs and authorities. To find the desired values for $x^{\langle p \rangle}$ and $y^{\langle p \rangle}$ Kleinberg suggests the *Filter(G,k,c)* and *Iterate(G,k)* procedures in Figure A-4. *Filter* procedure uses *Iterate* procedure to filter out the top c authorities and top c hubs in the G_σ with c ≈ 5-10. *Iterate* procedure returns good results with arbitrarily large values of k. Pages with the highest x-values are viewed as the best authorities, while pages with the highest y-values are viewed as the best hubs.

Iterate(G,k)

G: a collection of n linked pages

k: a natural number

Let z denote the vector $(1, 1, 1, \ldots, 1) \in \mathbf{R}^n$.

Set $x_0 := z$.

Set $y_0 := z$.

For $i = 1, 2, \ldots, k$

Apply the $\mathcal{I}$ operation to (x_{i-1}, y_{i-1}), obtaining new x-weights x'_i.

Apply the $\mathcal{O}$ operation to (x'_i, y_{i-1}), obtaining new y-weights y'_i.

Normalize x'_i, obtaining x_i.

Normalize y'_i, obtaining y_i.

End

Return (x_k, y_k).

Filter(G,k,c)

G: a collection of n linked pages

k,c: natural numbers

$(x_k, y_k) :=$ `Iterate`(G, k).

Report the pages with the c largest coordinates in x_k as authorities.

Report the pages with the c largest coordinates in y_k as hubs.

Figure A-4: Procedures that compute hubs and autorities in the subgraph *G*

For example in Figure A-5 each of A and B has one forward link, C has two backlinks and one forward link and D has a backlink that comes from C.

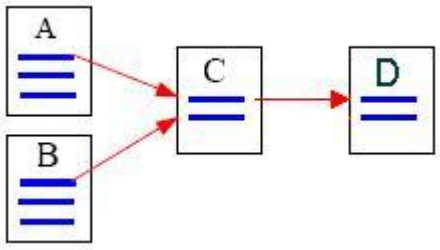

Figure A-5: Backlinks and forward links

Web pages vary on a wide range in respect to their quality and importance. In order to measure the relative importance of web pages Sergey Brin and Lawrence Page proposed PageRank, a method for computing rank of every web page based on the graph of the web. PageRank is defined like this in the original Google paper:

> We assume page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. Also C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:
>
> $$PR(A) = (1-d) + d\ (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$
>
> Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one. PR(A) can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.
>
> [Brin and Page 1998]

PR of each page depends on the PR of the pages pointing to it. But we won't know what PR those pages have until the pages pointing to them have their PR calculated and so on says Ian Rogers [Rogers 2002]. Thus, theoretically the final PR values of pages impossible but Brin and Page say that the results converge to 1 by using a simple iterative algorithm. We will explain this approach by a simple example from [Rogers 2002] that contains a network of 4 pages. Figure A-6 shows the network with backlinks and forward links of pages.

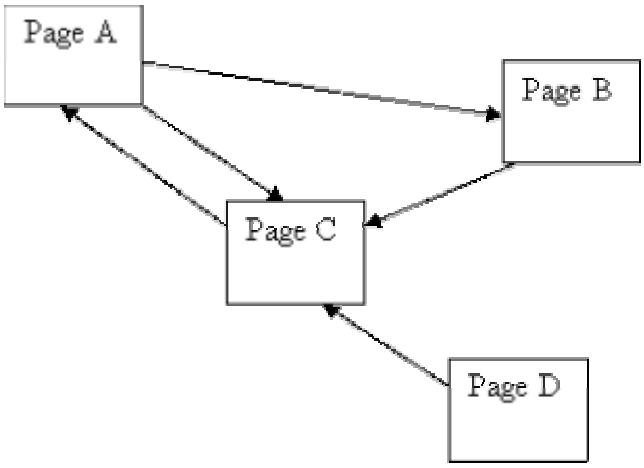

Figure A-6: Network of 4 nodes [Rogers 2002]

The damping factor, *d* is always constant and equals to 0.85. To calculate the PR of Page A PR of C must be known. However, PR(C) can not be known until learning PR(B) and PR(D). General PageRank formulas of each page:

$$PR(A) = (1-d) + d\,[PR(C)/C(C)]$$

$$PR(B) = (1-d) + d\,[PR(A)/C(A)]$$

$$PR(C) = (1-d) + d\,[PR(A)/C(A) + PR(B)/C(B) + PR(D)/C(D)]$$

$$PR(D) = (1-d) + d\,[0]$$

In the iterative approach, an initial value is given such as 0 (zero) to PR of all the pages to start the iteration process. Forward links of pages, namely C(A), C(B), C(C) and C(D) are also constant. C(A) = 2, C(B) = 1, C(C) = 1, C(D) = 1. Now we will iterate over PR and see what is happening.

Iteration I : PR(A) = 0.15 + 0.85*0 → 0.15

PR(B) = 0.15 + 0.85*(0.15/2) → 0.2137

PR(C) = 0.15 + 0.85*(0.15/2 + 0.2137 + 0) → 0.3954

PR(D) = 0.15

Average(PR) = 0.2272

Iteration II : PR(A) = 0.15 + 0.85*0.3954 → 0.3477

PR(B) = 0.15 + 0.85*(0.3477/2) → 0.2977

PR(C) = 0.15 + 0.85*(0.3477/2 + 0.2977 + 0.15) → 0.6783

PR(D) = 0.15

Average(PR) = 0.3684

Iteration III : PR(A) = 0.15 + 0.85*0.6783 → 0.7265

PR(B) = 0.15 + 0.85*(0.7265/2) → 0.4587

PR(C) = 0.15 + 0.85*(0.7265/2 + 0.4587 + 0.15) → 1.1219

PR(D) = 0.15

Average(PR) = 0.6142

It is clear that the average of PageRanks converges to 1. After 20 iterations the PR(A) becomes 1.49, PR(B) reaches to 0.78, PR(C) results with 1.58 and PR(D) remains same at 0.15. The iteration number is also reasonable for large networks.

> PageRank on a large 322 million link database converges to a reasonable tolerance in roughly 52 iterations. The convergence on half the data takes roughly 45 iterations. This graph suggests that PageRank will scale very well even for extremely large collections as the scaling factor is roughly linear in log n.
>
> [Brin and Page 1998a]