

**COLLECTION AND CLASSIFICATION OF
JAVASCRIPT LIBRARIES INCLUDED IN
WEBSITE**

**A Thesis Submitted to
the Graduate School of Engineering and Sciences of
Izmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of**

**MASTER OF SCIENCE
in Computer Engineering**

**by
İrem ATICI**

**June 2015
IZMIR**

We approve the thesis of İrem ATICI

Examining Committee Members:

Assist. Prof Dr. Tuğkan TUĞLULAR

Department of Computer Engineering, Izmir Institute of Technology

Assist. Prof Dr. Selma TEKİR

Department of Computer Engineering, Izmir Institute of Technology

Assist. Prof Dr. Semih UTKU

Department of Computer Engineering, Dokuz Eylul University

25 June 2015

Assist. Prof Dr. Tuğkan TUĞLULAR

Supervisor, Department of Computer Engineering, Izmir Institute of Technology

Prof Dr. Halis PÜSKÜLCÜ

Head of Department of Computer
Engineering

Prof Dr. Bilge KARAÇALI

Dean of Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

I would like to thank Asst.Doç.Dr.Tuğkan TUĞLULAR for his extraordinary support and valuable feedbacks in this thesis process.

Secondly, I should also thank to my manager Selim ÖZOKAN from Doğanata Elektronik A.Ş. and my colleague Gökhan ÇİMEN for giving me their support and encouragement to do my thesis.

Finally, I would like to thank my parents for their unlimited patience, support and love during this thesis.

ABSTRACT

COLLECTION AND CLASSIFICATION OF JAVASCRIPT LIBRARIES INCLUDED IN WEBSITES

Over the past years, web development processes have been improved incredibly with the fact that websites and web applications became extremely useful and beneficial in several different areas such as business, education, e-commerce, entertainment etc. At the same time, web development tools and technologies have been improved to be able to develop more creative and interactive websites and web applications. Therefore, a wide variety of methods, tools and languages for developing websites and web applications are available for the web developers nowadays. In this thesis, JavaScript among all of the web development technologies is analyzed with the hope of contributing web development processes.

A prototype named JS_Librarian has been developed for this thesis with the purpose of collection and classification of JavaScript libraries included in websites. The application accepts website URLs as input. After it receives URLs, it makes request to the webpages of the website and retrieves the HTML code. Then JavaScript code is extracted from the HTML code, so the basic process of information retrieval is achieved in this way. After the information retrieval process, JavaScript codes are analyzed and selected classification methods are applied to the URLs that are residing in the system.

At the end of the study, JavaScript libraries have been classified based on specified website categories and the retrieval reports from the application represents that class of JavaScript libraries used in websites may vary according to categories of websites.

ÖZET

WEB SİTELERİNDE YER ALAN JAVASCRIPT KÜTÜPHANELERİNİN TOPLANMASI VE SINIFLANDIRILMASI

Yıllar ilerledikçe, web sitelerinin ve uygulamalarının çok çeşitli alanlarda, iş, eğitim, e-ticaret, eğlence gibi, oldukça kullanışlı hale gelmesi gerçeği ile web geliştirme süreçleri hızlı bir şekilde ilerleme göstermiştir. Aynı zamanda, daha yaratıcı ve dinamik web siteleri ve uygulamaları geliştirebilmek adına, web geliştirme araçları ve teknolojileri de ilerleme göstermektedir. Bu sebep ile günümüzde, web geliştiriciler için çok fazla sayıda yöntem, araç ve yazılım dili bulunmaktadır. Bu tez çalışmasında, web geliştirme süreçlerine katkıda bulunabilmek umudu ile bu teknolojilerden biri olan JavaScript incelenmektedir.

Bu tez için JS_Librarian adında bir uygulama, web sitelerinde yer alan JavaScript kütüphanelerinin toplanması ve sınıflandırılması amacıyla hizmet etmek üzere geliştirilmiştir. Uygulama girdi olarak web site URL'lerini kabul eder. URL'ler alındıktan sonra, web sitelerinde bulunan sayfalara istek gönderilir ve HTML kodları çağırılır. Toplanan HTML kodları arasından JavaScript kodları bulunur. Bilgi alma işleminin önemli bir kısmı bu şekilde tamamlanır. Bu aşamadan sonra, JavaScript kodları analiz edilerek, seçilen sınıflandırma metotları sistemde bulunan URL'lere uygulanır.

Çalışmanın sonunda JavaScript kütüphaneleri, belirlenen web site kategorilerine göre sınıflandırılmıştır ve uygulamadan elde edilen rapor sonuçlarına göre JavaScript kütüphaneleri sınıfları web sitelerinin kategorilerine bağlı olarak çeşitlilik gösterebilmektedir.

TABLE OF CONTENTS

LIST OF FIGURES	ix
LIST OF TABLES	x
CHAPTER 1. INTRODUCTION	11
CHAPTER 2. BACKGROUND	13
2.1. Overview of Websites	13
2.2. Languages used in Websites	18
2.2.1. HTML and CSS	19
2.2.2. JavaScript	20
2.3. Information Retrieval	24
2.4. Classification Algorithms	27
CHAPTER 3. METHODOLOGY	29
3.1. Classification through Machine Learning	30
3.2. Naïve Bayes Classification	32
CHAPTER 4. TECHNOLOGIES USED	36
4.1. Jsoup	37
4.2. Solr	38
4.3. Lucene	39
4.4. Java Machine Learning Library (Java-ML)	42
4.5. WEKA	43
CHAPTER 5. PROTOTYPE DEVELOPED	46
5.1. Overview of the JS_Librarian Application	47

5.2. JS_Librarian Application Specification	48
5.3. JS_Librarian Application Architecture	50
5.3.1. Solr Architecture in JS_Librarian Application	51
5.3.2. Lucene Architecture in JS_Librarian Application	52
5.3.3. Database Structure in JS_Librarian Application.....	53
5.3.4. Information Retrieval in JS_Librarian Application	54
5.4. Website Categories for JS_Librarian Application	55
5.5. JavaScript Library Categories for JS_Librarian Application	56
5.6. JS_Librarian Application Implementation.....	57
5.6.1. Input Data for the Application	57
5.6.2. Storing Basic Input Data in Solr	58
5.6.3. Parsing HTML and JavaScript Codes for the Input	60
5.6.4. Indexing HTML and JavaScript Codes with Lucene	62
5.6.5. Finding Frequency of Specified Keywords.....	63
5.6.6. Implementation of Naïve Bayes Algorithm	64
5.6.7. Implementation of WEKA	65
 CHAPTER 6. EXPERIMENTAL RESULTS	 69
6.1. Experiment Description	69
6.2. Experiment with JS_Librarian Application	70
6.3. Experiment with WEKA.....	72
 CHAPTER 7. CONCLUSION	 74
REFERENCES	75
 APPENDICES	
APPENDIX A. CATEGORIES OF JAVASCRIPT LIBRARIES	78

APPENDIX B. GRAPHICAL REPRESENTATION OF RESULTS TAKEN FROM JS_LIBRARIAN APPLICATION	84
--	----

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
Figure 2. 1. A map of ARPANET in December 1970 [2]	13
Figure 2. 2. A web page structure [3]	14
Figure 2. 3. Interaction of web components	15
Figure 2. 4. Dynamic website structure	17
Figure 2. 5. Layers of a web page [5]	19
Figure 2. 6. Tree of objects	22
Figure 2. 7. The structure of HTML DOM	23
Figure 2. 8. Processes of information retrieval [14]	25
Figure 2. 9. Flowchart of classification process [16].....	27
Figure 2. 10. Classification types [17]	28
Figure 4. 1. The structure of Lucene Index.....	39
Figure 4. 2. Graphical user interface of WEKA	44
Figure 5. 1. Software architecture diagram of JS_Librarian application.....	50
Figure 5. 2. Architecture of Solr insertion processes.....	52
Figure 5. 3. Lucene indexing and searching architecture	52
Figure 5. 4. Tables and views	53
Figure 5. 5. Crawling process	54
Figure 5. 6. Single URL as input	57
Figure 5. 7. Websites as input by crawling Alexa	58
Figure 5. 8. Solr schema xml	58
Figure 5. 9. The main page of the application	60
Figure 5. 10. Search a keyword in HTML codes.....	63
Figure 5. 11. Search a keyword in JavaScript codes	63
Figure 5. 12. Table that holds keyword frequency	64
Figure 6. 1. An example of WEKA result	733

LIST OF TABLES

<u>Table</u>	<u>Page</u>
Table 4. 1. Java-ML algorithms and features	42
Table 6. 1. Class of each website category	71

CHAPTER 1

INTRODUCTION

In today's world, internet is spreading all over the world and number of internet users is increasing continuously. 40% of the world population has an opportunity to connect internet today. It was less than 1% in 1995. The number of internet users reached to first billion in 2005 and with this year it is more than third billion.

Additionally, internet is accessible from everywhere. Beside the improvements in internet, portable device usage has also become increasingly common. These devices provide almost all functionality of the computers to the users. For this reason, people have opportunity to access internet by using these devices from anywhere in the world. This kind of advantage increases the interest in web-based applications because people can complete their different tasks via web-based applications from their online devices, such as webmail, social networking, product request forms, login pages, shopping carts, interactive mapping software, CMS etc.

In a world that websites are significantly important and inevitable in business and daily life, we wanted to navigate deeply through the structure of websites and studied for the basis of websites in detail. For these reasons, we decided on JavaScript technology, which has a critical role in today's applications. When we investigate deeply we found out that numerous JavaScript libraries have been developed rapidly and they all available for usage today and it can be said that each library goes ahead with their specific functionalities. Thus, they can be separated from each other and classified into categories.

One of the basic difficulties in developing websites is deciding which JavaScript library is more convenient for the content and category of the application. There is not enough literature on this area and there are not any helpful applications that will guide developers through using the best libraries in their development phases.

The main obstacle for this study is the enormous number of JavaScript libraries in web world. Trying to categorize them was difficult. We have used machine learning for this study. Naïve Bayes is the one of core method used in this study.

The main goal of this thesis is to give a viewpoint to its audience about which libraries are used in which kind of websites. Web developers want to select the most convenient JavaScript libraries for their applications. This thesis provides statistical information about the usage of JavaScript libraries. Thus, people who are about to develop website can benefit from this theses.

Our hypothesis is that websites that are belonging to a specific category are using specific JavaScript libraries. We have developed the application named JS_Librarian in this study for testing this hypothesis. JS_Librarian application and its background are explained in the following Chapters. Organization of this thesis is as in the following:

CHAPTER 2 has been organized for the explanation of the background work of this study. Basic concepts related to this thesis have been mentioned in this section. Thus, the audiences can feel more comfortable with the core ideas of this study.

CHAPTER 3 contains the methodology explanations. Classification methods used in this study are explained in this chapter.

CHAPTER 4 represents the technologies used in this thesis. Jsoup, Solr, Lucene and Java Machine Learning Library are explained in this chapter.

CHAPTER 5 describes details of the JSLibrarian application. Every aspect of the application is explained in detail.

CHAPTER 6 discusses the experimental results based on JS_Librarian application.

CHAPTER 7 contains conclusion and future work of this study.

CHAPTER 2

BACKGROUND

2.1. Overview of Websites

The history of websites based on ARPANET that stands for Advanced Research Projects Agency Network. It is a wide-area network developed by United States Defense Advanced Research Project Agency (ARPA) in 1969. Universities and research centers were linked by ARPANET and new networking technologies were tested with it. UCLA and the Stanford Research Institute were first two nodes of this network [1]. Then it reached over 60000 nodes all over the world as research network. A map of the ARPANET can be seen in Figure 2.1.

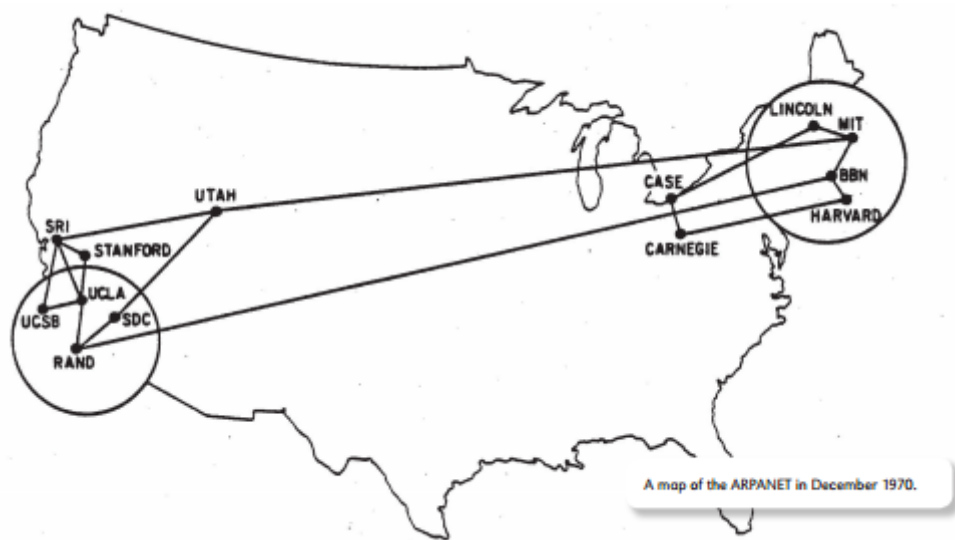


Figure 2. 1. A map of ARPANET in December 1970 [2]

Subsequent to the development of ARPANET, the internet searched out in 1970's. In the following period World Wide Web was developed by Tim Berners-Lee and Robert Cailliau at the European Particle Physics Laboratory (CERN). It was introduced to the world in 1989. Development of the Internet was an important factor for the revolution of 20th century. It transformed all the ways of doing affairs in the world.

WWW includes enormous number of computers that play role in files to be available through the Internet by using the HyperText transfer Protocol, HTTP. Thus, files have been started to be shared among the computer users. Therefore, it became very popular in a short time and today, more than 250 million people in the world are using the web.

As mentioned above WWW is consisting of web pages and websites. Both of them are commonly used concepts in the world of web. These concepts are seemed as they have the same meaning with each other. However, they are slightly different. While web page represents an individual page, a website consists of collection of web pages. Namely there are several web pages belonging to a website. A web page is the visible part of web file and it is exclusively an index to several different files such as images, PDFs, hyperlinks, sounds, videos etc. As several different files are linked to a web page as seen in Figure 2.2, the number of these files can be very large for a website [3]. In this chapter history, design, structure, types based on structure, categories based on content of websites are mentioned.

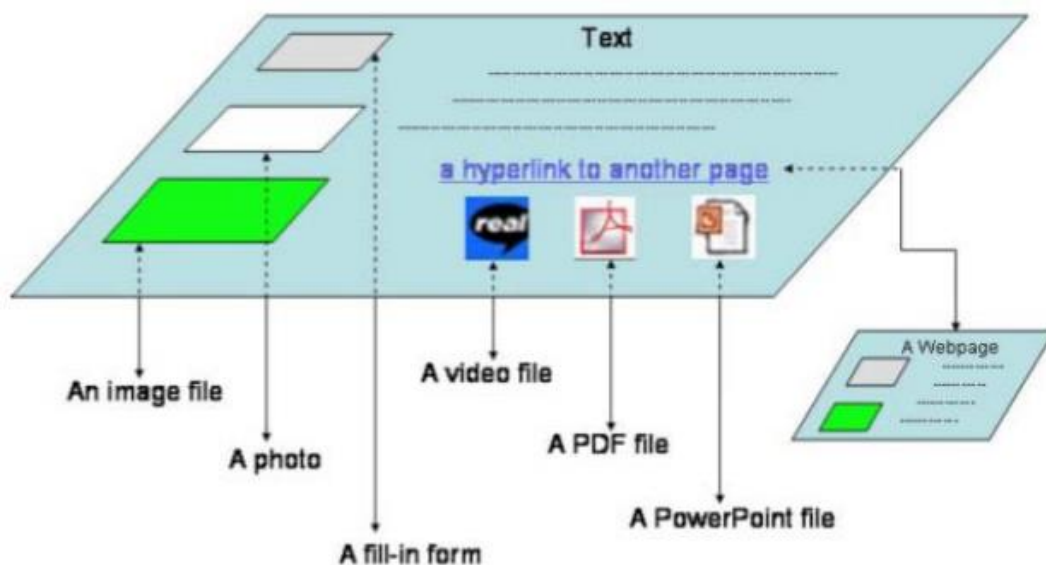


Figure 2. 2. A web page structure [3]

As mentioned in above, websites are created from web pages and they have three components for their implementations. These components are host, web server and client. The host is a computer that is connected to a TCP/IP network and it is a repository for the services such as email, FTP and World Wide Web. It can be accessed by a user who resides in remote location. On the other hand, the web server is a

program. It uses the client/server architecture and HTTP to serve the web pages to web users. Most common web servers are Apache and IIS. Finally, the client is a software application that recognize the services that server supplies. Client mostly resides on desktop computers or on a host. Their interaction can be seen in Figure 2.3.

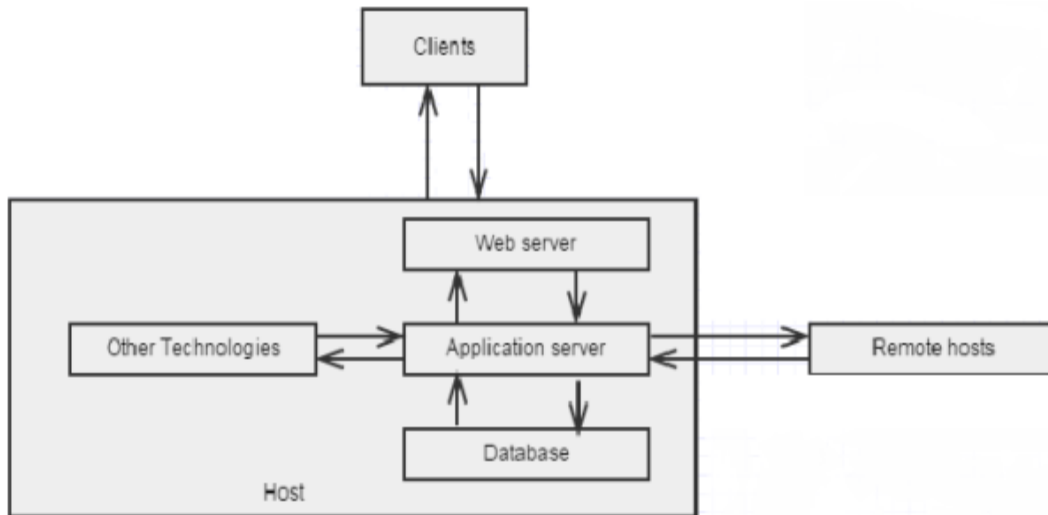


Figure 2. 3. Interaction of web components

A website consists of the documents, style sheets, and scripts. Therefore, development phases of a website include authoring, styling, and scripting/programming. Authoring phase is the content preparation for representation on the Web. The content is marked up with the HTML tags. The roles of HTML in websites are explained in the following. Styling phase is related with the appearance of the page in the browser. It is organized by style rules that are created in CSS (Cascading Style Sheets). CSS is separated from HTML. This separation has advantages like improving accessibility of content, sharing a style by multiple HTML pages and controlling specification of representation characteristics more easily. The other phase is scripting/programming that enables elements of a web page to perform functions. Scripting is commonly used to add different functionalities and behaviors to the web pages. JavaScript is a popular scripting language and there are also web related languages such as Python, PHP, Ruby, and ASP.NET. These languages run on the server while JavaScript runs on client when it is used for client side scripting. The codes run on server and the codes run on client can be grouped into front-end and back-end programming. Front-end design refers to the design process that is related directly to the browser and scripting with JavaScript

language is a process belonging to front-end design. Front-end design also includes graphical design, interface design, information design on the web page, HTML document, and style sheet development. On the other hand back-end development refers to the programs and scripts that run on server behind. Languages such as Python, PHP, Ruby, and ASP.NET are used in back-end processes. Back-end development includes information design on the server, forms processing, database programming, content management system. In this thesis the focus is on the HTML and JavaScript languages from the front-end design.

As known semantic web is a study that aims to make information available to the machines not people. Web 2.0 concept has come up with the semantic web studies. Although the recent developments on the semantic web and XML formats are improved, achieving the interpretation of the information by the machines is difficult technically. Therefore, classic web is still widely used platform [4]. Internet users are benefiting from the websites in all part of their life. Today especially in business world companies in variety scale need a website because the websites are the best way to keep in touch with the existing customers and to inform potential customers about products of a company. Additionally companies can give opportunity to buy a product via dynamic websites to their customer. Companies become accessible in this way and accessibility is an important factor in success. Thus, it cannot be imagined that a company without a website can participate in competition in business world.

Day by day developments in web technologies are reflected to the websites. As a result websites are started to be developed as dynamic to meet user requirements successfully. There are two types of dynamic website that can be analyzed. One of them is server-side dynamic website and the other one is client-side dynamic website. The scripts of server-side dynamic website are run on a web server. The structures of this kind of websites are controlled by an application server. On the other hand, the scripts of client-side dynamic website are run on a client browser. HTML scripts are executed firstly when the page is loaded. For the interaction of user and the website, JavaScript and other scripting languages like Action Script, VBScript, and Typescript are used. The scope of this thesis includes client-side dynamic websites and one of the interaction languages JavaScript. Client-side dynamic website structure can be viewed in Figure 2.4.

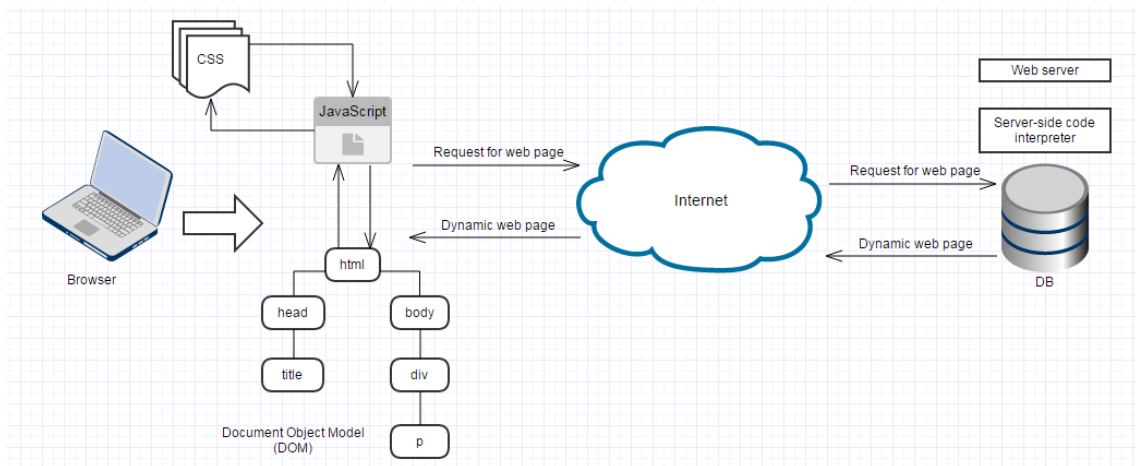


Figure 2. 4. Dynamic website structure

The basic structure of a website is formed by HTML (Hypertext Markup Language) which is a standard markup language for developing websites. First steps of developing HTML were taken by Tim-Berners-Lee in 1980 with the idea of sharing document for CERN researchers as mentioned at the beginning of this section. Today web browsers are using HTML to interpret texts, images, videos and other materials and to depict the web content in a user friendly form.

JavaScript also plays an important role in dynamic websites structure. Actually it can be thought as an extension to HTML. Objects are created by HTML tags and manipulated by JavaScript. While HTML is used only to display static websites, dynamic functionalities are given to the websites by the help of JavaScript. JavaScript has enormous types of functions for the advanced programming. JavaScript libraries help programmers by representing pre-written JavaScript codes. These libraries have been developed to accomplish common JavaScript tasks. Most popular JavaScript libraries are listed in the following:

- JQuery
- MooTools
- Prototype
- Dojo Toolkit
- Ext Core
- Ext Js

2.2. Languages used in Websites

Today it is inference that the most popular websites are dynamic websites and dynamic website development consists of client-side coding, server-side coding and database operations. Dynamic website can be developed with several different programming languages. These languages can vary according to content. There are several specific languages used in web development. However, other languages are also used in client-side or server-side scripting.

Basically HTML and CSS are core languages for the construction of the backbone of a website. Therefore, nearly all websites are built by applying these two languages. While HTML constructs the main structure of a website, CSS deals with style and makeup of a website. However with these two languages only a static website can be built. HTML and CSS is adequate for creating a website that looks same to its every visitor. However, today a great number of websites are dynamic websites that are well cut to his visitors. Thus, for the development of dynamic websites more advanced client-side and server-side scripting languages are necessary.

Client-side scripting languages used to create scripts that run on browsers, so HTML and CSS are client-side languages. Additionally, JavaScript and ActionScript are commonly used languages for the client-side. On the other hand, server-side scripting is used for the transfer of the data between webserver and browser. Server-side scripts run on the web server. While source codes of client-side scripts are located on the client, server-side scripts are located on a web server. PHP, Java, Python, Ruby, Rails, Scala, C, C++, ASP.NET, Hack, FBML, Erlang, D, Xhp,Go are the examples of server-side scripting languages.

Most popular websites today have been developed by using JavaScript in their front-end. For example Google, Twitter, Bing, Blogger, Wikipedia, MSN, Outlook,

Yahoo, YouTube, and Facebook are using variety of server-side scripting languages. However, all of them are using JavaScript as scripting language. As seen JavaScript is the common language for all popular websites. It was the reason that initiates this study. In the following sections, HTML, CSS and JavaScript languages among all these mentioned languages are handled.

2.2.1. HTML and CSS

In the case of construction of a web page is considered, it can be determined that a web page contains three layers as depicted in Figure 2.5.

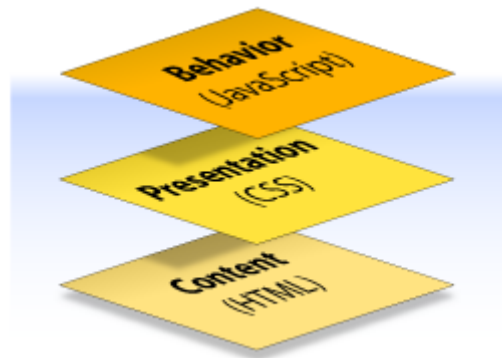


Figure 2. 5. Layers of a web page [5]

The content layer is the visible part of a web page, so HTML or XHTML construct this layer. Visible part of a web page is embedded in these markup languages. Visible content consists of text, images, videos, sounds etc. The presentation layer deals with the appearance of a web page. CSS gives shapes to a web page while it is viewed by user in a browser. On the other hand, the behavior layer includes the user interaction processes which are created by JavaScript. These interactions may be the validation of inputs while filling an membership form.

HTML is the standard hypertext markup language used for constructing the backbone of websites. HyperText provides a way to move around the web, by clicking the hyperlinks that navigate user to another page. The word hyper expresses that it is not linear. User can navigate any page by using these hyperlinks. These actions are not in an order. HTML consists of markup tags. Markup is used to mark text in a certain type. A

web page is described by HTML tags, so each HTML tag describes distinct document content. HTML tags and scripts are not visible on browser. Browsers read and render HTML files for the representation of web pages. HTML can embed scripts and CSS codes.

XHTML is a type of HTML which uses the syntax of XML, the Extensible Markup Language. XHTML is an XML application. Therefore, elements of XHTML are same with the HTML, but their syntaxes are not totally same. It is also stricter than HTML and it is supported by nearly all browsers. DOCTYPE declaration is compulsory for XHTML.

CSS stands for Cascading Style Sheets. It is used to for the makeup of a website. Elements like color, layout, and font are determined and coded in a file separately from base codes. Thus, CSS file can be used by multiple webpages. The separation of HTML and CSS codes is referred to the separation of structure from presentation. Web pages can be viewed from different kinds of devices with the help of the CSS. Today there are different types of technological devices at different dimensions. CSS provide for web pages the best size in order them to be viewed on this devices.

2.2.2. JavaScript

JavaScript is a scripting language that is commonly used as part of the browsers. Brendan Eich, co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation is the creator of this language. It was developed as an improvement on Netscape browser in 1995. Its development was started as simple scripting language with the aim of saving HTML pages from being always static. Now it plays dramatic role in several different types of software applications [6]. Today it is one of the most popular languages in the world. ECMAScript is the untrademarked name of JavaScript [7]. JavaScript language can be used in client-side and server-side network programming. JQuery and Dojo libraries are the most used JavaScript libraries on client-side programming. As an example of server-side network programming Named Data Networking can be given and it can be implemented with NDN.js client library [8].

JavaScript is an object-oriented scripting language. Objects can be connected to JavaScript and can be controlled programmatically by JavaScript. One of the benefits of the JavaScript is keeping the server traffic flowing. The basic reason of it is validating

user input before sending to the server. Thus in case of invalid user input the website is not loaded to the server. The other benefit is interactivity. Interfaces are reacting according to users' behaviors on the page with JavaScript. Additionally, JavaScript provides reuse of code in development phase and the most important one is that it can be cached by the browser, so that the page appears faster to the users without need to reload [9].

The method of including JavaScript code into websites is embedding the code block on anywhere in the website among HTML code. If the current browser supports JavaScript, the code block must begin with the tag “<script language=“type/JavaScript”> </script>”. Because of the reason that JavaScript code blocks reside in HTML code, these code blocks become automatically open source. This openness can be advantage for the JavaScript developers. Each developer can benefit from current websites. Code blocks can be reviewed in order to develop better websites.

JavaScript and Java languages have similarity in name. Although the both languages are improved based on C and C++ languages, they are not related to each other in reality. They both object-oriented and they have similarities in syntax. However, they have a lot differences. While Java is a programming languages developed by Sun, JavaScript is a scripting languages which is strictly related to the browser. The commands written in JavaScript are executed by the browser. Browser interprets and performs the JavaScript codes. There are some basic concepts to understand JavaScript. They are listed as objects, functions, closures [10].

HTML and XML documents are represented by an API named Document Object Model (DOM). HTML documents are consisting of objects and JavaScript can be able to manipulate all these objects. In HTML, DOM is everything consists of nodes and it consists of Element, CharacterData, and Document. Element keeps the various elements added to HTML document. CharacterData keeps all characters of HTML document according to their type being Text or Comment. HTML document is stored in Document node. Client-side JavaScript programming basically developed on the idea of manipulating the document element [11].

Document Object Model is a W3C standard that has three different types: Core DOM, XML DOM, and HTML DOM. HTML DOM is in our scope of study. HTML Document Object Model of a website is created when a browser loads a website. This HTML DOM can be represented with a tree of Objects as seen in Figure 2.6.

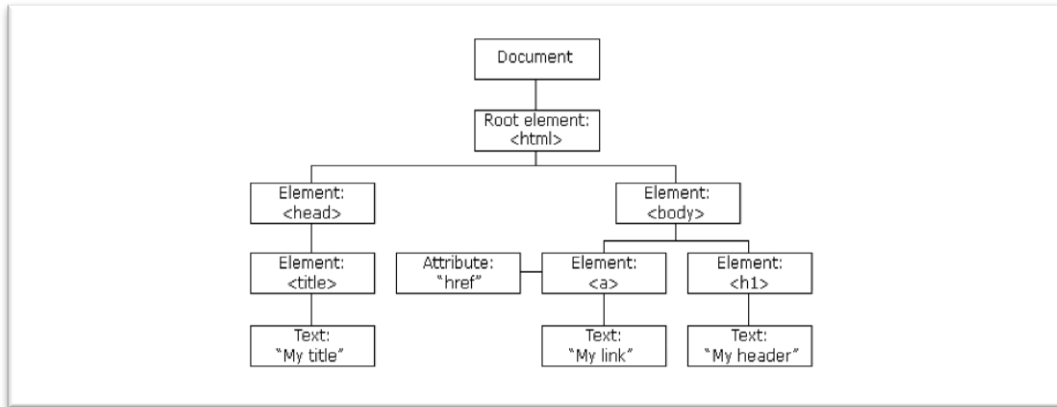


Figure 2. 6. Tree of objects

When Document Object Model of a website is formed, JavaScript has the power for changing a static website to a dynamic one. When dynamic HTML is thought, a few basic features must be highlighted such as adding, removing or altering all the HTML elements and attributes, altering all the CSS elements, to react all the HTML events and creating new events. In below there is an example representation of DOM structure according to given HTML document. The structure is shown in Figure 2.7.

```

<!doctype html>
<html>
<head>
<title>This is an HTML Document</title>
</head>
<body>
<h1>First page</h1>
<p>First paragraph in first page.</p>
<p>Click <a href="http://www.google.com">here</a> to visit google.</p>
</body>
</html>
  
```

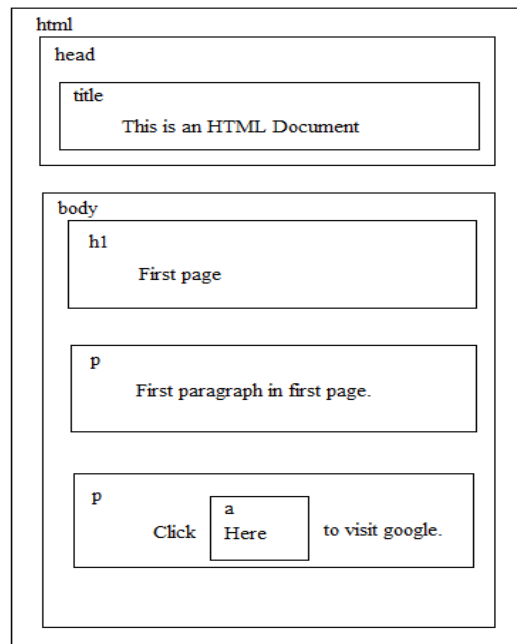


Figure 2. 7. The structure of HTML DOM

In advanced JavaScript programming, difficult tasks are handled with the help of JavaScript libraries. For general view, JavaScript libraries can be thought as the libraries of pre-written JavaScript codes. In today's technology world, there are enormous numbers of JavaScript libraries available. These libraries include functions about basic JavaScript features such as DOM manipulation, animation, Ajax handling etc. In this thesis JavaScript libraries are considered to belong to specific categories. They will be collected under the thirteen categories. These are listed in the following.

- Graphical/Visualization
- Forms
- Animation
- Database
- String and Math Functions
- Fonts
- Debugging and Logging
- Dom Manipulation
- GUI Related
- Web Application Related
- Pure JavaScript/Ajax

- Unit Testing
- Other

There are a few reasons to use JavaScript libraries in web development. Because of the opportunity of code reuse, using libraries can save time and work for the development period. Additionally, libraries supply consistent, reliable, and highly interactive user interface for web development.

2.3. Information Retrieval

Information retrieval is a method of retrieving references to documents in response to information requests. In this system, information can be searched in documents themselves or metadata that includes information about these documents. This search can be performed in a local database or in the Internet, so data like text, images, videos or sounds can be retrieved [12]. In this thesis information retrieval from internet is in the focus. Information retrieval system consists of specific processes that is performed by human or machines to achieve some tasks. These tasks can be indexing documents, formulating the search, searching, getting feedback etc [13].

Information retrieval system supports three basic processes. The first process is document content representation, the second one is user information need representation, and the final is two representation comparisons. These processes are illustrated in Figure 2.8. Squared boxes denote data and rounded boxes denote processes in the Figure 2.8[14].

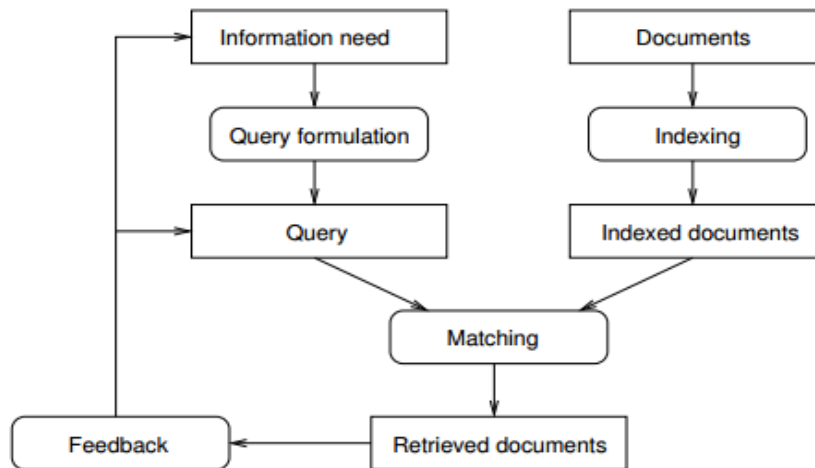


Figure 2. 8. Processes of information retrieval [14]

Indexing process is representing the document. This process can be performed without the interaction of the end user of the information retrieval system. The purpose of the search is information need, so information need process is usually expressed with query formulation process. Query is the result representation. In other words, query formulation process may demonstrate the active dialog among the system and user. Query formulation is not only related with convenient query. It is also related with better understanding of user for information need. Query comparison in document representation is named as matching process. It often produces results in a ranked list of documents. Information need of the users should be searched on this document list. Relevant documents are listed on the top by the ranked retrieval. The frequency distribution of terms in documents can be used by effective ranking algorithms. Ranking algorithms reduce the time spent by user to read documents.

Information retrieval can have several different meanings in real world. As a simple example, reading the serial number that reside on the identity card and then typing it to a system can be a form of information retrieval. It can be defined academically as “Information Retrieval (IR) is finding material of an unstructured nature that satisfies an information need from within large collections”. With this explanation, it can be inferred that information retrieval is task that can be performed by only a few people. However, with the developments in the web today, hundreds of millions of people are performing information retrieval process every day. They face this situation simply while they are searching on internet or searching their email.

Information retrieval is getting more popular form of information access and it is replacing the traditional database style searching [15].

Information retrieval concept also supports users in the process of filtering document collections and also preprocessing on retrieved documents. With the set of documents collected by the information retrieval method, clustering is a step for grouping these documents according to their contents. With the set of topics, classification is a step for determining belonging class(es) of each set of documents. Classification is generally achieved by first classifying a number of documents manually. Then it is expected to be able to classify newly added documents automatically.

In recent years, information retrieval is included in operating systems. Additionally, email programs have the ability of text classification beside its search functionality. They have a spam filter, and they provide mail classification option by manual or automatic in order to replace mails in specific folders. The important point in information retrieval is management of wide range of document types on a computer. Management includes free maintenance of the search system in startup, processing, and disk space usage without the interaction of a user. When retrieval applied on collections like websites belonging to a category, the documents should be kept in a centralized file system. In this case a specified machine perform search over the collection [15].

Information retrieval and classification are core functionalities for this thesis. In this section information retrieval is the focused concept. It has several different definitions but Information retrieval is equivalent to JavaScript library collection in our study. By using Jsoup class belonging to Java language can retrieve the necessary texts from the websites. Jsoup implementation is explained in CHAPTER 4.

2.4. Classification Algorithms

In this thesis, the objective is classification of JavaScript libraries in websites according to these websites' category. One of an idea behind the classification is classifying a given example into given set of categories. Classification has two varieties: supervised classification and unsupervised classification. Supervised classification is implemented with labeled training data while unsupervised classification uses unlabeled data. In this study supervised classification is in the focus.

With the classification, problems such as text categorization, market segmentation, machine vision, fraud detection etc. can be solved. Our problem is closer to the text classification problem because we consider to classifying JavaScript code documents. A flow chart for classification method can be seen in Figure 2.9.

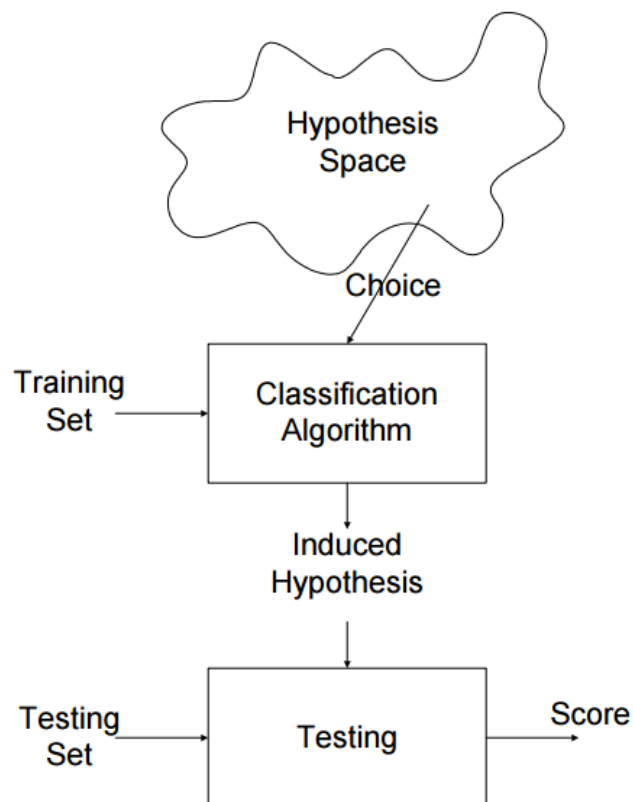


Figure 2. 9. Flowchart of classification process [16]

Some kind of classification problems can only have two categories as positive or negative. This classification named as binary classification. If there are more than two

classes as solution to a problem in a classification, it can be called as multi label classification [17]. In this study, multi label classification is convenient to the specified problem, because there are thirteen classes that are defined before the implementation of JS_Librarian application as explained in next chapters. Classification types are depicted in Figure 2.10.

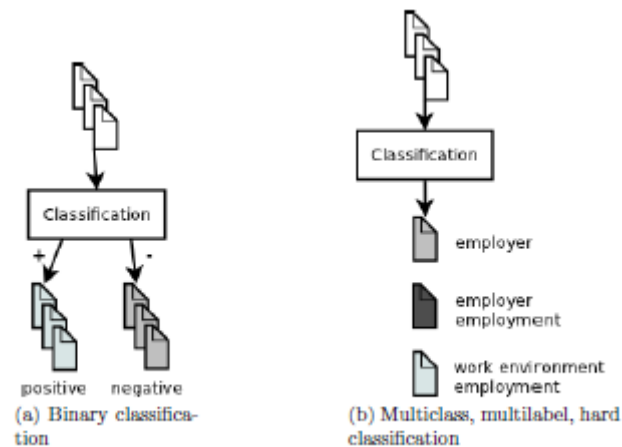


Figure 2. 10. Classification types [17]

Classification processes can be achieved in four steps. These are data collection and preprocessing, building the model, testing the model and finally classification of new documents. Details of these processes are explained in the chapter that prototype is introduced. There are several different algorithms for the implementation of classification. Thus, classifiers are also varying. These algorithms are listed in the following.

- Decision trees
- Perceptron
- The Nearest Neighbor
- Naïve Bayes
- Balanced Window
- Maximum Entropy

CHAPTER 3

METHODOLOGY

This chapter discusses the methodology that is specified to be used in this thesis. There are main two specific aspects to be focused in this study. These basic aspects are collection of HTML and JavaScript codes included in websites and classification of these websites according to used JavaScript libraries in their codes. Among these two aspects the core part for the specified methodology is related to the classification aspect. In order to achieve dedicated classification of this study, machine learning concept has been analyzed firstly.

In last two years, machine learning concept has been in demand in information technology field, because the amounts of data have been increasing with an exponential speed. Most of the time, these data are not meaningful without having any process. However, all these data that are collected and stored in different platforms are actually very valuable when they are exposed to computational processes. This fact finds out the significance of smart data analysis methods in today's technology world. For these reason, machine learning concept is becoming an intriguing and attractive study field nowadays [27].

Classification processes of this study take the advantage of machine learning methods. The basic idea behind the machine learning concept is to predict new information by learning from previously collected data. We are going to apply this opinion while we are finding the class label for our test data, namely in classification. Normally machine learning is used in web search, email spam detection, recommender systems, scoring credit, fraud detection, stock trading etc. Machine learning has several different types and we are focusing on classification in this thesis. In classification concept classifier is the critical aspect. A classifier accepts a vector of discrete or continuous feature vectors. The result produced by a classifier is a discrete value and it is called as class [28]. This class can take two values like yes/no or it can take multi values.

It is determined that machine learning fits well for our problem of JavaScript library classification. However, the problem faced is that there are numerous different

learning algorithms available. This number is counted with thousands according to literatures. Additionally hundreds are introduced each year. Three components can facilitate the selection among these algorithms. These components are representation, evaluation and optimization. Thus, a classifier should be represented in a few formal computer languages. Additionally evaluation function can be used to specify good classifiers from bad ones. Finally the efficiency of learner can be determined by the optimization techniques.

In this study Naïve Bayes classification method has been chosen among thousands of machine learning algorithms. Naïve Bayes classifiers are mostly applied ones among other for classification problems. The main goal of this thesis is to see a result from the process of classifying JavaScript libraries. In the past, there is not any study on this field. There are only a few applications that finds out JavaScript libraries used in websites. However, these applications are not finding any class for websites given as input. It is aimed in this thesis to find class for given websites. As a result an applicable classification method is beneficial for our study. Thus, Naïve Bayes classification method has been chosen to be implemented. Machine learning concept, Naïve Bayes classification method and Naïve Bayes classification algorithm are explained in the following.

3.1. Classification through Machine Learning

With the improvements in the information technologies, amounts of data in the world have reached to huge sizes. Data are continuously collected by the computer systems and stored in different storages. However, these collected data are not meaningful alone without exposed to any processes. Inferences are needed for them to be beneficial information. Day after day, it is getting a harder task to extract information from these data without special methods and tools. At that point machine learning is one of a convenient method for the extraction of information from raw data. After applying any machine learning algorithm on raw data, they are turned into more useful and beneficial information for people to use and for science to make decisions. These algorithms can be applied to several different real world problems such as medical diagnosis, credit approval, target marketing etc.

Machine learning concept is derived from the field of Artificial Intelligence. However it cannot be said that it is Artificial Intelligence itself. It is only a part of this field. Machine learning can be applied for several different applications such as data mining, natural language processing, image recognition, and expert systems. Motivation for the machine learning research is mainly based on the problems from biology, medicine, finance, astronomy, etc. Spam email detection and machine translation problems from the real world; have been solved with machine learning methods [18].

Machine learning can be applied in several different types of applications in real-world and beneficial solutions can be produced as results. These applications can be listed as in the following [19]:

- Speech recognition
- Computer vision
- Bio-surveillance
- Robot control
- Accelerating empirical sciences

This method has two types as unsupervised and supervised learning as mentioned in previously. Class labels of training data are not known in unsupervised learning. It is convenient for data clustering. On the other hand, class labels of training data are known in supervised learning. Supervised learning is convenient for the data classification and because of the purpose of JavaScript library classification of this study, supervised learning is convenient. After the training data set is prepared with the class labels, new data is classified based on to this data set.

Machine learning has two processes in basic and both of them have applied in JS_Librarian application. Model construction is the first step to achieve classification with machine learning. A set of prepared data should be given to the system and a model should be constructed ready to be used. After a model extracted from the data set, data with unknown class is supplied to the system and class is found based on this model.

Machine learning has a few kinds of algorithms that can be applied to the classification problems. Some of them are listed in the following:

- Naïve Bayes
- Logistic Regression
- Decision Trees
- Support Vector Machines

Among these algorithms Naïve Bayes was chosen for this thesis. With Naïve Bayes algorithm training data set is not needed to be too large. Interaction between data is not needed in this study, classification is enough. Explanation about Naïve Bayes classification method and implementation of the algorithm are given in the following section.

3.2. Naïve Bayes Classification

Naive Bayes classification is one of an algorithm belonging to machine learning concept. It can be said that it is also a case for Bayesian Networks which is a probabilistic graphical model to depict a set of variables and their conditional dependencies. Bayesian Network applies a directed acyclic graph for the representation [20]. Naïve Bayes classification algorithm is applied for classification of JavaScript libraries included in website for this thesis.

Naïve Bayes classification method has been investigated since 1950s. This method was developed for text categorization. For the implementation of method, word frequencies are used. It represents the supervised learning method. Naïve Bayes classification aims to determine the class of given data with a series of calculation based on probabilistic principles. In this method a specific number of trained data are given to the system. These data must have class variable. Probability calculations on trained data are processed with the test data to determine class.

Bayes theorem can be formulized as in the following [21]. y is the class variable and x is the feature vector, where x_i represents each features.

$$P(y | x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n | y)}{P(x_1, \dots, x_n)} \quad (3.2.1)$$

Using the naive independence

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y), \quad (3.2.2)$$

for all i ,

$$P(y | x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i | y)}{P(x_1, \dots, x_n)} \quad (3.2.3)$$

Since $P(x_1, \dots, x_n)$ is constant following classification rule can be used:

$$\begin{aligned} P(y | x_1, \dots, x_n) &\propto P(y) \prod_{i=1}^n P(x_i | y) \\ &\Downarrow \\ \hat{y} &= \arg \max_y P(y) \prod_{i=1}^n P(x_i | y), \end{aligned} \quad (3.2.4)$$

Naïve Bayes method has several different types listed as in the following:

- Gaussian Naïve Bayes
- Multinomial Naïve Bayes
- Bernoulli Naïve Bayes
- Out-of-core Naïve Bayes Model

In this study multinomial Naïve Bayes method among all above is implemented. Simple, efficient and effective discriminative parameter learning method is applied by multinomial Naïve Bayes classifiers. This method learns parameters by discriminatively computing frequencies from training data. It can be inferred from literatures that discriminative parameter learning can be related with Discriminative Frequency Estimate (DFE). DFE holds the benefits of both generative and discriminative learning methods [22].

Generative probabilistic distribution is a commonly used learning method for modelling the machine learning and understanding problems such as pattern recognition, artificial intelligence, and perception etc. It supplies rich framework for

monumental structure and prior knowledge on the solution to the problem. In generative probabilistic models, joint probability distribution is used, so features and output variables are presented homogeneously. In classification and regression processes, generative models can be applied, since it estimates joint distribution over all variables. Examples of generative models are listed below [23]:

- Gaussian mixture model and other types of mixture model
- Hidden Markov model
- Probabilistic context-free grammar
- Naive Bayes
- Averaged one-dependence estimators
- Latent Dirichlet allocation
- Restricted Boltzmann machine

While generative modeling approach aims to model combining structure, priors, invariants, latent variables and data for the good joint density specified to domain, discriminative modeling approach deal with optimizing less domain specific model for the classification and regression processes. Distributions of the domain specific variables are not aimed to model in discriminative approaches. This approach deals with optimizing the mapping from the inputs to the expected outputs. Thus, consequent classification is important without modeling the variables. Image and document classification are applicable area for the discriminative modeling. Examples of discriminative approaches listed in below [22]:

- Logistic regression
- Gaussian processes
- Regularization networks
- Support vector machines
- Traditional neural networks

In text classification two classic Naïve Bayes variants that are word vector counts and tf-idf vectors are used. Multinomial Naïve Bayes uses the Naïve Bayes algorithm for multinomially distributed data. The distribution is denoted by vectors

$\theta_y = (\theta_{y1}, \dots, \theta_{yn})$. n is the number of features. For each class y and θ_{yi} is the probability $P(x_i | y)$ of feature i .

θ_y is counted as:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

$N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T , and $N_y = \sum_{i=1}^n N_{yi}$ is the total count of all features for class y .

CHAPTER 4

TECHNOLOGIES USED

The opinion proposed in this thesis is supported with an application named JS_Librarian. JS_Librarian is an application that is developed and implemented only for this thesis. It collects JavaScript codes and classifies websites according to used JavaScript libraries. All the codes of the application are written in Java with Eclipse IDE. Beside Java, several different technologies have been used in JS_Librarian application development phases. In CHAPTER 3 two basic aspects in this study are mentioned, so JS_Librarian application has been designed based on these two aspects. They are specified as information retrieval and classification. Technologies that will be used in the implementation of this application are tried to be selected as compatible to the functions defined.

A Java library named Jsoup is used in information retrieval process of this application. All HTML codes are collected by using the classes and functions of Jsoup library. JavaScript codes are extracted from retrieved HTML codes with the algorithm implemented in Java for this application without applying any tool. Additionally, in information retrieval phase Solr and Lucene technologies are used. Basic website data are kept in Solr architecture and Lucene is used to be able to perform keyword search among HTML and JavaScript codes.

MySQL database management system has been used in order to store the frequency data of keywords that represent JavaScript libraries. Keywords that are specified according to JavaScript libraries are counted and frequency data are stored in this database system to be able to utilize in classification phase. These keywords have been determined according to name of libraries. Naïve Bayes algorithm has been implemented by using the tables and views created in database. Machine learning library of Java is applied for the implementation of Naïve Bayes algorithm. Finally WEKA is also applied to implement the Naïve Bayes algorithm. These technologies help this study for the solution of the problem specified for this thesis.

4.1. Jsoup

Jsoup is a Java library for HTML parsing process. It supplies an API for retrieving and manipulating data. It includes packages in its API. These packages are listed below:

- org.jsoup
- org.jsoup.examples
- org.jsoup.helper
- org.jsoup.nodes
- org.jsoup.parser
- org.jsoup.safety

Each of above packages includes classes and these classes are named as Jsoup, Document and Element. Jsoup class supplies functions for the connection, cleaning, and parsing the HTML document. In this study we have used connect and select functions. Jsoup can be applied for several different examples like getting title of URL, getting title from HTML file, getting total links of URL, getting meta information of URL, getting total images of URL, getting from parameter etc [29].

In this study, information retrieval is the critical function for the rest of the JS_Librarian application. It includes retrieving HTML and JavaScript codes. Jsoup serves for this purpose. Jsoup library works with the HTML included in websites. It performs the functions by using the DOM, CSS, and jquery-like methods. As explained in CHAPTER 2, browsers extract the DOM for each websites and Jsoup works as a browser while it is parsing HTML. A website URL is given as input to the program and Jsoup fetches the home page of this website, parses it to a DOM and selects the data into a list that is derived from Elements class. In the code block written in below, doc object derived from Document class holds the HTML code of the <http://www.irs.gov/> with the usage of the connect() and get() functions from Jsoup library.

```
Document doc = Jsoup.connect("http://www.irs.gov/").get();
```

Beside the HTML code retrieval in this application, Jsoup also extracts the all URL included in a website. Thus, a website can be analyzed through all of its pages. To

achieve this select() method of document class is utilized. The following code is used for this purpose.

```
Elements newsHeadlines = doc.select("a[href]");
```

4.2. Solr

In this thesis website indexing information are stored in Solr architecture. Apache Solr is a project that is developed based on Apache Lucene project. Document is the basic unit in Solr for data representation and fields are the sub elements of Document. According to database expressions, document can be thought of a table row and field can be a table column. Data are represented as document and for the insertion of documents to Solr, a schema should be determined. After the installation of Solr, a file named as schema.xml is created. This schema includes the name of fields, unique/primary key features for fields. Fields in Solr can be in different types such as float, long, double, date, and text. New field types can also be created in Solr. A field is defined with name, type, indexed, stored, and multivalued features [30]. In Solr searches are performed via HTTP GET. A query string is executed from the specified URL.

Solr and Lucene are strictly related to each other. Lucene is an important component in Solr. It is open source enterprise search platform for full-text search. It was written on Java and it runs on Tomcat [24]. It uses Lucene for the search functionality. Actually Solr is very simple to implement. In this study firstly xml schema of the Solr is configured according to related fields. Then documents are inserted to Solr architecture and then Solr basic queries are applied on Solr to retrieve data.

In this application Solr will be applied for storing the lightweight data of the websites. Solr is applied because of the easy access and easy integration with Lucene. We will keep website ID, website URL, website category and operation date in Solr. Therefore we should configure schema.xml files.

Schema.xml files in Solr contain some pre-written data fields. However these data fields cannot server us in order to store basic website data. For this reason additional data fields must be appended to the existing ones. After appending our custom data

fields to all related schema.xml files of Solr, it is ready to record our specific data. After data are inserted, Solr can be queried according to specific queries. We can call our data with queries compatible with Solr architecture.

4.3. Lucene

Lucene is a library written in Java which is used for full-text search basically. Its working principle is based on creating index for raw data. Queries on Lucene can be performed after the index is created.

The advantage of using Lucene is getting fast search responses, because Lucene does not perform a direct search on text, instead it performs search on indexes. The index used by Lucene can be called as inverted-index because a page-centric data structure is converted to keyword-centric data structure with Lucene.

For the searching and indexing concept of Lucene, named Document plays an important role. In other words, an index consists of one or more Document in Lucene. As an example, documents are added to the index during the indexing operation and documents are retrieved during the searching operation.

On the other hand Documents also consist of Fields. Fields are composed of name and value pairs. As an example, a Field in an application can be age of a person and the name of the field will be age, the value of the Filed will be the age of a person in this case. Figure 4.1 shows the structure of index.

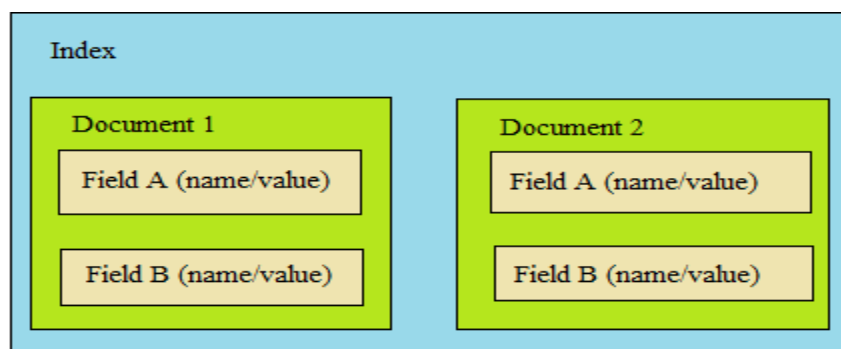


Figure 4. 1. The structure of Lucene Index

The other important issue about Lucene is that it is an API and in order to be able to use it one should know all the class of Lucene. These classes are listed in the following [25]:

IndexWriter class is used to create and maintains indexes. An index can be created by creating an object belonging to IndexWriter class. Creation of new index, opening an existing index to add Document functions are organized by this class. A code block of creating an index is in the following:

```
Directory indexDir = FSDirectory.open(new File("index-directory"));
IndexWriterConfig config = new IndexWriterConfig(Version.LUCENE_4_10_2, new
StandardAnalyzer());
IndexWriter indexWriter = new IndexWriter(indexDir, config);
```

Directory class is used to denote the directory in which the index is stored.

Document class is used to represent a document that is the unit of indexing and searching in Lucene. An example of code block is in the following.

```
Document doc = new Document();
doc.add(new TextField("title", title, Field.Store.YES));
```

Field class is used to represent a part of a document.

Analyzer class is used to extract the tokens of a document. Most commonly used implementation of this class is Standard Analyzer class. With Analyzer class a tokenizer is created firstly then it breaks the stream of characters into raw tokens.

IndexSearcher class is designed to perform search in index.

Term is used to denote the basic components of Lucene search. It can represent a word in a text. Additionally it can represent dates, email addresses etc. A field with its name and value can be represented by a term.

QueryParser class creates a parser to perform a search on index. An example code block is in the following.

```
String querystr = args.length > 0 ? args[0] : "lucene";
Query q = new QueryParser(Version.LUCENE_CURRENT, "title", analyzer).parse(querystr);
```


Query class stores the search criteria created by the Query Parser class.

Term Query is created for matching the documents that meets the query.

Hits class holds the documents that are returned by the query result.

Index of Lucene is splitted to chunks named segments. Each segment can be defined as index actually. Lucene search in all of them in sequence.

When the index writer is opened in the program new segment is created and when a new document is added to the index, it will be added to the next segment. Thus, the files of a segment do not need to be changed.

In Lucene, old segments cannot be changed. When a document is deleted, its segment's info is stored as deleted in a file and the document is kept in segment physically. The operation is similar in updating also. Updated document is marked as deleted and new version of this document is added to the current segment. In some cases Lucene can integrate some segments with an optimize.

- FDT
- FDX
- FNM
- FRQ
- NRM
- PRX
- TII
- TIS
- GEN

4.4. Java Machine Learning Library (Java-ML)

Java machine learning library consists of machine learning algorithms that are implemented in Java language. The basic elements of the library are dataset and instance. These two elements have core interfaces for the implementation. Java-ML algorithms and their features are represented in Table 4.1.

Table 4. 1. Java-ML algorithms and features

Clustering	Classification	Feature selection
K-means-like (7) Self organizing maps Density based clustering (3) Markov chain clustering Cobweb Cluster evaluation measures (15)	SVM (2) Instance based learning (4) Tree based methods (2) Random Forests Bagging	Entropy based methods (4) Stepwise addition/removal (2) SVM_RF Random forest Ensemble feature selection
Data filters	Distance measures	Utilities
Discretization Normalization (2) Missing values (3) Instance manipulation (11)	Similarity measures (6) Distance metrics (11) Correlation measures (2)	Cross-validation/evaluation Data loading (ARFF and CSV) Weka bridges (2)

In coding world, machine learning library is written in several different languages such as, Python, .NET, C++ etc. Each machine learning library contains sub libraries and Java has also. Its sub libraries are listed in the following:

- Spark
- Mahout
- Weka
- Mallet
- JSAT

In this study Naïve Bayes algorithm is also implemented with Weka library of Java.

4.5. WEKA

WEKA stands for Waikato Environment for Knowledge Analysis and uses the GNU General Public License (GPL). It is a machine learning platform which is written in Java. It includes the implementation of classification, clustering, and association rule mining algorithms. WEKA has five basic aspects named as data processing, classification, clustering, attribute selection, and data visualization. It is compatible with

ARFF file format basically. Additionally, it supports several different formats such as CSV and ASCII. WEKA processes the data by using a large number of methods.

On the other hand WEKA includes large number of classification methods. Classification methods can be categorized as in the following:

- Bayesian methods: Naïve Bayes, Bayesian nets, etc.
- Lazy methods: Nearest neighbor and variants
- Rule-based methods: Decision tables, OneR, RIPPER
- Three learners: C4.5, Naïve Bayes trees, M5
- Function-based learners: Linear regression, SVMs, Gaussian processes
- Miscellaneous methods

WEKA supports unsupervised learning by clustering schemes such as EM-based, k-means etc. Additionally, attributes are necessary for the performance of classification. Finally, specialized tools are used for visualization such as tree viewer, Bayes network viewer, and dendrogram viewer.

All the features mentioned above can be implemented by several different graphical user interfaces of WEKA. These interfaces are named as Explorer, Experimenter, and Knowledge Flow. Explorer interface based on weather data can be shown in Figure 4.3. Explorer is the popular interface of WEKA that supports all the features listed above dynamically.

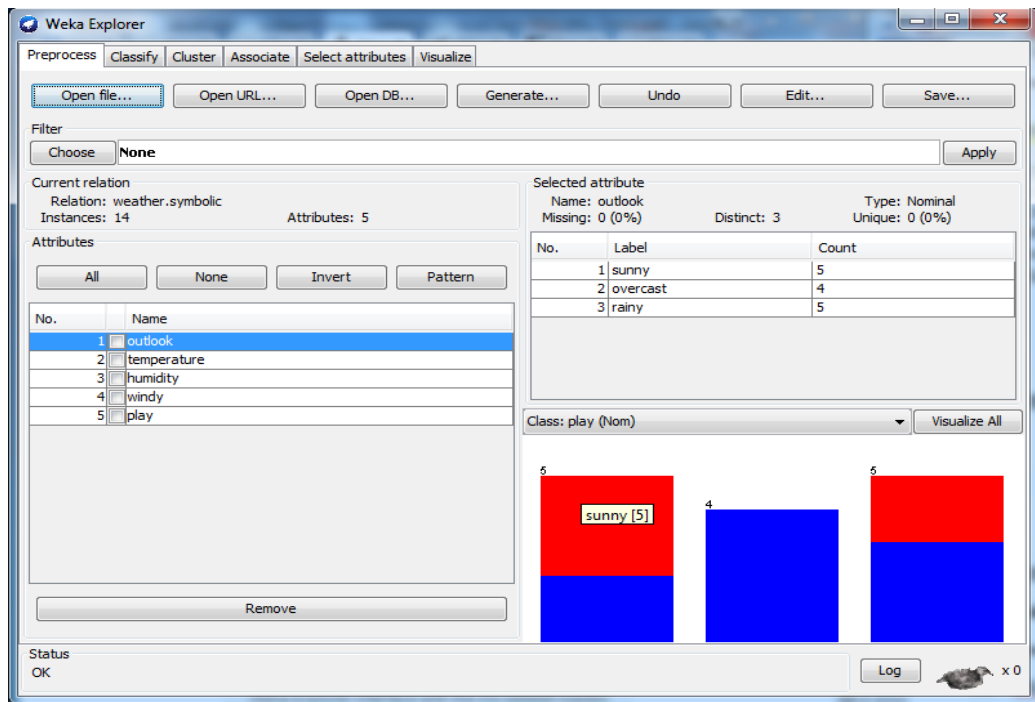


Figure 4. 2. Graphical user interface of WEKA

Experimenter interface is used for the implementation of machine learning experiments on classification and regression methods. Experiments can be implemented as parallel on different computers. These computers are in a network. Thus, the methods of performance analysis can be separated over multiple machines.

On the other hand, the knowledge flow interface is similar with the Explorer for data exploration, processing and visualization processes. Only difference is that knowledge flow interface performs these processes in a workflow oriented system. A workflow including the way of data load, preprocessing, evaluating and visualization can be determined with this interface.

Beside the graphical user interface WEKA has also Bayes network editor, an SQL viewer, ARFF data file viewer editor, command line interface and Java API. WEKA is applied for the machine learning and data mining fields and it starts to be used also widely in several different academic fields nowadays. The reason for WEKA to be used widely in different fields is being free and open-source software. Additionally, it is portable that its last version is written in Java. Older versions were only used in Unix operating systems. However, today it can be used on different operating systems through the Java Virtual Machine. On the other hand, graphical user interface of WEKA plays an important role for its wide usage. Machine learning

practices can be implemented with Explorer interface by users. Thus, WEKA became accessible for the users. On the other hand, new algorithms can be implemented and compared to the other algorithms. It provides a collection of data for this experiment [31].

Java API of the WEKA will be used in this study. For the implementation of machine learning algorithms, WEKA can be used in the Java code. `weka.core.FastVector` and `weka.core.Attribute` classes are the core classes for expressing the problem with features and create a simulation of .arff file. `weka.core.Instances` class is applied for creating a training set and the training test is filled with an object belonging to class `weka.core.Instance`. The final process for WEKA API is choosing a classifier from the `weka.classifiers.Classifier` class.

CHAPTER 5

PROTOTYPE DEVELOPED

The opinion of this thesis is supported with a prototype named JS_Librarian. Methodologies and technologies used for this prototype have been explained in CHAPTER 3 and CHAPTER 4, so the prototype has been developed in Eclipse platform with Java language. It is a web application and it runs on Tomcat server. All methodologies and technologies that have been explained in previous chapters are implemented in this application as explained in this chapter.

Information retrieval and classification are the basic two functions that are implemented in this study. Information retrieval consists of retrieving HTML codes of the websites and extracting JavaScript codes from these codes. HTML and JavaScript codes are stored in the file system in this study. They can be retrieved and downloaded from the application whenever they are needed. In addition, Solr and Lucene implementations have been also achieved in parallel with the information retrieval processes.

For the classification processes MySQL database management system has been used and training data set are kept in this database system. For the implementation of supervised learning, class labels must be defined previously. Thus, we have determined our class labels firstly and we specified thirteen class labels in this study. These class labels can be seen in Appendix A. Learning data is the critical factor for the supervised classification methods. For this reason we have created our learning data and specify their classes to be used in the next processes. Finally test data are given as input to the system and the implemented Naïve Bayes algorithm uses the training data set for the test data processing. Metrics are important for the software quality measurements, so we have applied WEKA for the quality assessment of our implementation.

5.1. Overview of the JS_Librarian Application

This thesis has been decided to be written based on the problem of JavaScript library classification included in websites that are belonging to several different categories. The origin of this problem is based on the difficulties faced when developing web sites and web applications in today's technology world. With the rapid development in the web technologies tools and libraries have also developed and their numbers have increased. One of them is JavaScript libraries. There are enormous numbers of JavaScript libraries available and each of them serves for specific purposes. Therefore, there appear so many choices for the web developers and among these complexity decisions making is getting harder for the web development day by day. The aim of this thesis is to contribute to web development world by classifying JavaScript libraries according to website categories. It is expected that this aim is achieved with the help of the JS_Librarian application.

JS_Librarian application has been designed as having interactions with several different technologies. The first function of the application is extracting HTML code of the given URL with Jsoup library. Then the next function is extracting JavaScript code from the extracted HTML. Extracted HTML and JavaScript codes are saved to the text documents. Basic website data such as id, URL, category, crawling date are inserted to Solr architecture and retrieved from there. After gaining the HTML and JavaScript codes the next step is indexing them with Lucene to perform keyword search. Then, libraries used in JavaScript codes are mined with the algorithm written in Java. Mined results are inserted to database designed with MySQL. After all of these steps are finished, Naïve Bayes algorithm written in Java is applied to determine the class of the given URL.

When more than a specified number of website URLs are given as input to the system, reports that represents the solution to the specified problem can be received from JS_Librarian application.

5.2. JS_Librarian Application Specification

Before implementing JS_Librarian application, there are two basic phases that must be applied firstly. One of them is specifying the website categories. In order to be able to represent meaningful reports to the web developers, inputs should be collected according to specific website categories. Thus the outcomes that are taken from the application can represent such kind of website categories include specific JavaScript categories. This kind of outcome will make our study more valuable.

The second one is specifying categories of JavaScript library. There are several kinds of JavaScript libraries in today's technology as we mentioned before. Because of the reason that we want to classify JavaScript libraries that are included in website, we firstly specify the classes and keywords that represent JavaScript libraries. After all functional requirement can be implemented successfully.

JS_Librarian has to perform several specific tasks. These tasks will be performed in a specific order. First of all, the website categories, JavaScript categories and keywords belonging to JavaScript libraries should be specified. Starting point of this application will be the website URL. Algorithms that are responsible of achieving specific tasks will be triggered after a URL is given to the application as an input.

The first task is inserting the entered website lightweight data to the Solr structure. In order to achieve this task, Solr should be configured as explained above.

The second task is parsing and storing HTML codes included in the entered website URL. The algorithm will go through the website URL; it will take and keep the HTML codes in a buffer. At the same time, the algorithm will create a folder that has a same name with the website with HTML suffix. Then it will create a document named as websitename_HTML.html and write the HTML codes from buffer to this document.

The following task is parsing JavaScript codes included in entered website URL. The algorithm is designed to collect codes located between the <script> and </script> tags. It will collect all the JavaScript codes with this method and keep the code in a buffer. While extracting the codes with this method, it will also check the JavaScript libraries used in the code. If it catches any JavaScript library, it will go through this library and get the code. At the same time algorithm will create a folder that has the same name with the website with JavaScript suffix. Then it will create a document named as websitename_JavaScript.js. At the end, the algorithm creates two js file. One

of them is the code block located between `<script>` `</script>` tags and the other is all JavaScript codes with the JavaScript library codes.

After the HTML and JavaScript codes are parsed and stored in specific documents, Lucene algorithm will index these HTML and JavaScript documents and create indexes in the HTML and JavaScript folders of the entered website. With these indexes, any keyword can be searched on HTML and JavaScript documents.

After these phases, the other algorithm will go through the JavaScript folder of the entered website URL and begin to read the js file. Its purpose is to find specific keywords included in JavaScript code and to calculate the count of appearance of these keywords. Whenever it finds and calculates the frequency it inserts the fresh data to the related table located in MySQL database system.

The following algorithm will apply Naïve Bayes classification method to classify the entered website according to JavaScript codes included in it. This algorithm will use the older data stored in MySQL as learning data to predict the newly entered website class.

One of the algorithms will perform the task of downloading HTML and JavaScript codes. It will be triggered with a button near the website data retrieved from Solr. When the button is clicked regarding website HTML codes or JavaScript codes will be downloaded.

Another algorithm will be used optionally in the case of entering website URL collectively. In this case the algorithm will go through a website and take the website URLs located in it. This algorithm is compatible with the website www.alexacom.com. From this website our algorithm retrieves website URLs according to website categories.

5.3. JS_Librarian Application Architecture

In the previous chapters we introduced the basic concepts that are applied in JS_Librarian application developed and the motivation for this thesis. This chapter will go through the specifications, architecture and design phases of the application.

Software architecture provides a high level overview for a system. It should be completed before the design and implementation phase. The main target of software architecture is basically to represent the subsystems and general interactions between these systems which construct the overall systems. Thus it must represent all subsystems and the interactions between them successfully. Behind, software architecture should include properties of subsystems and their relations. Thus details can be explained in design phase based on a successful architecture.

For the JS_Librarian application, there appears several numbers of subsystems which have interaction with each other. Before we explain all these subsystems, it should be better to show them in a diagram with their interaction. Software architecture diagram for JS_Librarian application can be seen in Figure 5.1.

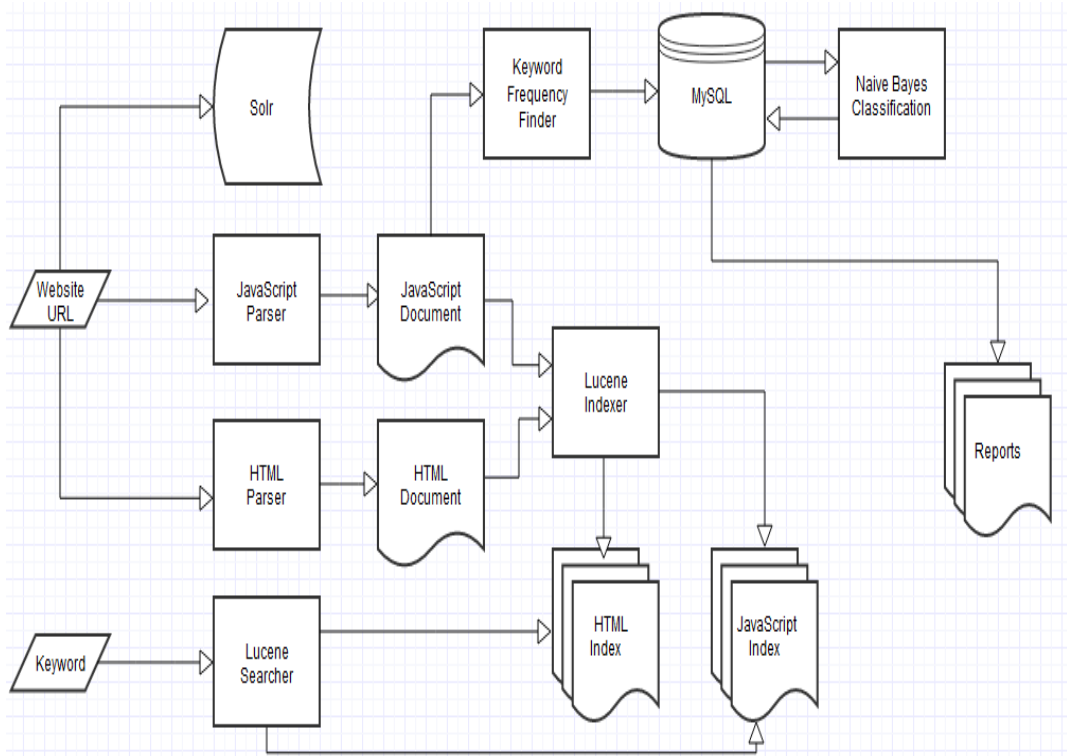


Figure 5. 1. Software architecture diagram of JS_Librarian application

JS_Librarian application has been designed in order to solve the problem of collecting and classification of JavaScript libraries included in websites. While solving our problem we have applied algorithms that have been developed specifically for this project. These algorithms can be explained according to their tasks performed in the application. For example, the task of first algorithm is to parse HTML codes included in websites, the second algorithm parses the JavaScript codes, the third algorithm finds the frequencies of specific keywords included in JavaScript codes, the fourth algorithm applies Naïve Bayes classification methods and validate the result according to WEKA class, the fifth algorithm inserts the basic website data to Solr structure such as URL, category, operation date and the sixth algorithm performs the index operations for HTML and JavaScript codes by using Lucene structure.

Algorithms are applied on the Eclipse platform with the Java language and the solution to our problem will be reached firstly by supplying a website URL to the application. After a URL is given as an input, all the algorithms mentioned above are executed in a specific order and all the result will be stored in MySQL database. When an enough number of inputs are supplied to the application, specific reports that are related to our problem can be taken. These reports can bring us to the solution of our problem.

5.3.1. Solr Architecture in JS_Librarian Application

Solr processes are performed initially in the program flow. Determined attributes of websites are initially inserted to the Solr system which is configured according to requirements. Solr processes can be followed after a single URL or collective URL entrance according to architectural design. Design of the Solr insertion processes is illustrated in Figure 5.2.

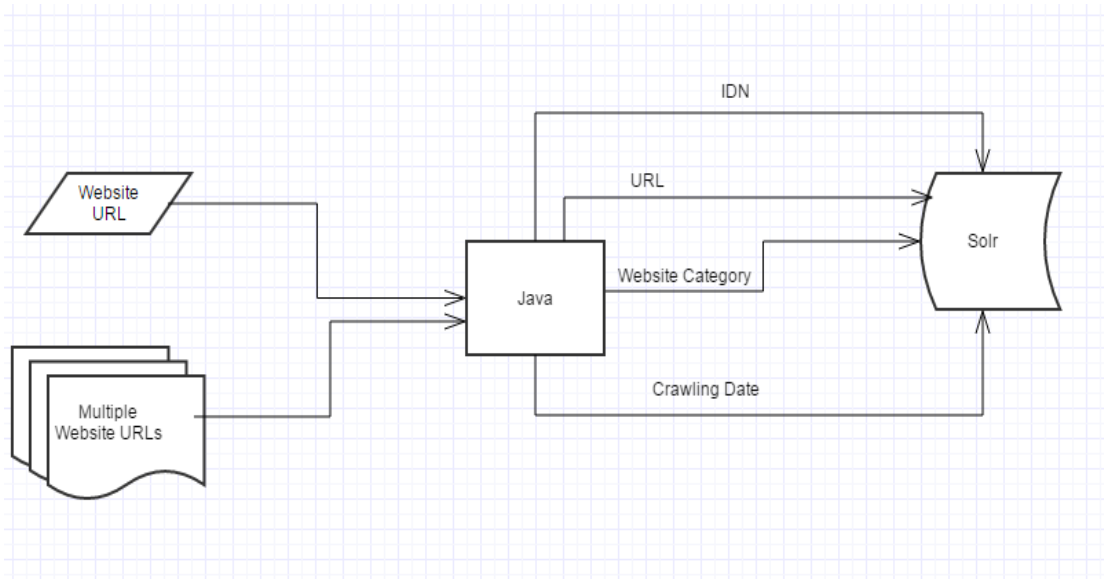


Figure 5. 2. Architecture of Solr insertion processes

5.3.2. Lucene Architecture in JS_Librarian Application

Lucene processes are implemented after HTML and JavaScript codes are retrieved from websites. Whenever the program extracts these codes and saves them in documents, Lucene indexes that on the fly. For each websites there are only one index and searching keywords are performed on this index. Lucene index and search processes architecture is depicted in Figure 5.3.

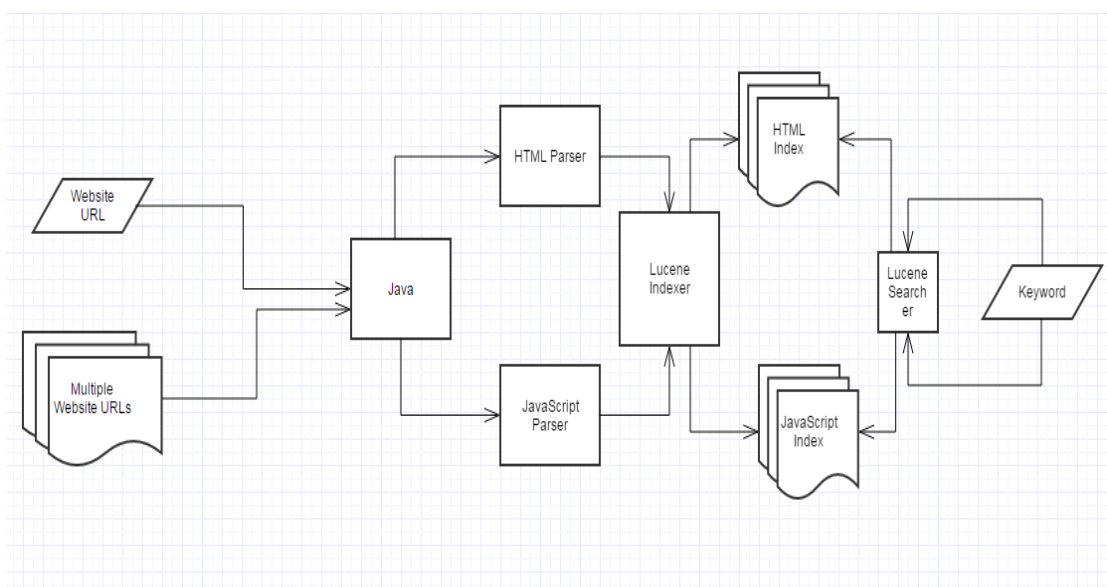


Figure 5. 3. Lucene indexing and searching architecture

5.3.3. Database Structure in JS_Librarian Application

After insertion of basic attributes of websites to Solr, MySQL is used for the operations of data created from remaining processes such as applying Naïve Bayes classification algorithm. MySQL provides flexibility with regards to Solr in data operations. Additionally experimental reports are obtained with the help of views created in MySQL database system. Tables and views for the JS_librarian application can be seen in Figure 5.4.

First of all website URL and its content category are stored in TBL_CRAWLED_DATA table. This table also contains two more fields for first and second degree of JavaScript categories. However, in the first round these two fields are not determined because they are specified with the Naïve Bayes algorithm. After this table is filled another table named TBL_KEYWORD_FREQUENCY are filled with the keyword frequencies of given website JavaScript codes. When these two tables are filled, algorithm written for Naïve Bayes method implementation uses the views created according to requirements of this method.

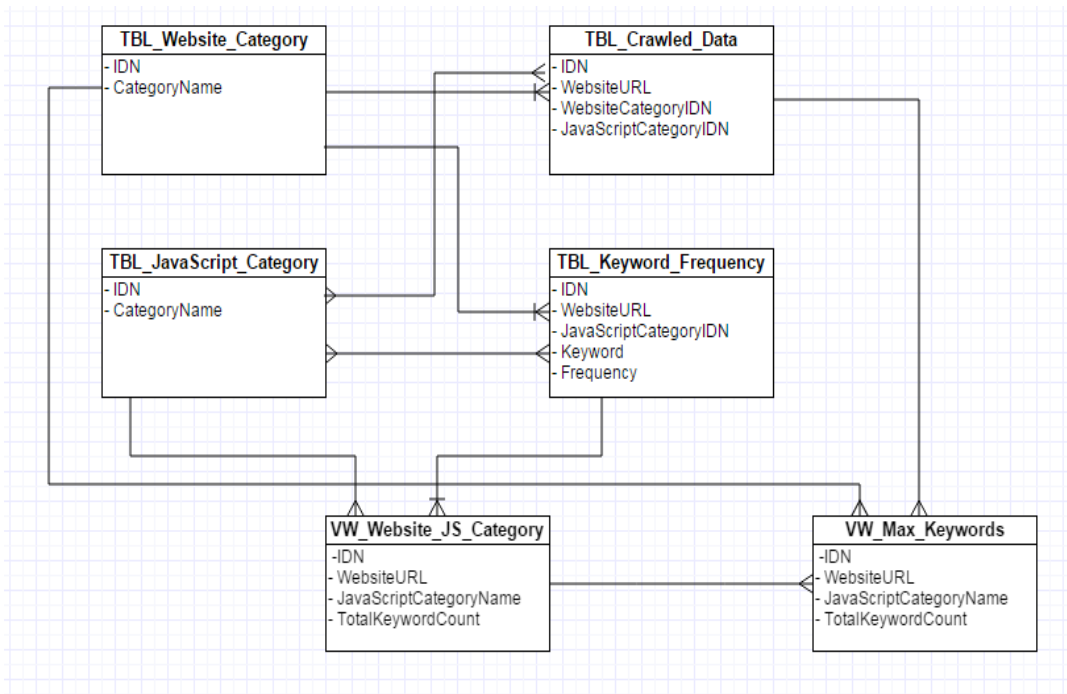


Figure 5. 4. Tables and views

5.3.4. Information Retrieval in JS_Librarian Application

In this study websites are analyzed according to their categories. These categories are arts, business, computer, games, health, home, kids and teens, news, recreation, reference, regional, science, shopping, society, sports, world. Alexa website contributed for the implementation. Alexa website provides traffic of web data to the users. Its basic objective is to collect data during to browsing with the help of its toolbar and this toolbar sends collected data to the Alexa website. Data are analyzed by Alexa and then data traffic and rankings are represented to the user. Today Alexa is working with nearly 30 million websites and it can store websites according to categories mentioned above [26]. Thus, Alexa is a good choice for this thesis to work with enormous number of websites, which are belonging to different categories.

The starting point of JS_Librarian project implementation is to crawl the Alexa website according to selected category. The program first goes to the <http://www.alexa.com/topsites/category> and collects the specified number of websites from this URL. After reaching the URLs of websites that are based on specific categories, the program continues from the second step. Figure 5.5 shows the crawling processes of the application.

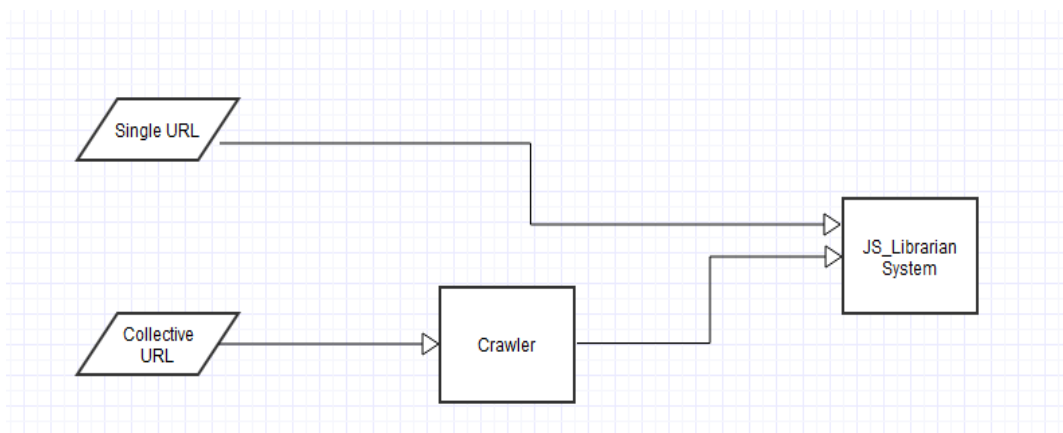


Figure 5. 5. Crawling process

After collecting the website URLs, each URL is visited by the program and then HTML and JavaScript codes are parsed. A directory with URL name is created and HTML and JavaScript codes are saved in this directory.

JavaScript source codes are collected along with their libraries. Each library used in JavaScript is also navigated and codes are extracted from these libraries. Finally all JavaScript codes are collected in one document as ready to be analyzed.

5.4. Website Categories for JS_Librarian Application

JS_Librarian application has been implemented in Eclipse JEE Luna platform. User interfaces of the application have been developed by using Java Server Pages. The application runs on Tomcat server. In the following, implementation of each subsystem has been explained.

Before explanation of the subsystems, it would be better to mention categories of websites and categories of JavaScript libraries. In order to be able to begin the implementation these categories should be specified first of all.

Website categories that will be used in the application have been specified by examining the Alexa system because Alexa is working with nearly 30 million different websites and it has separated the websites according to their categories. Thus, there are Alexa based seventeen website categories and in the application these sixteen categories are used. Website categories that are applied in the development are listed below.

- Arts
- Business
- Computers
- Games
- Health
- Home
- Kids and Teens
- News
- Recreation
- Reference

- Regional
- Science
- Shopping
- Society
- Sports
- World

5.5. JavaScript Library Categories for JS_Librarian Application

After specifying the categories of websites, the next step is specifying the categories of JavaScript libraries. Determined categories are listed below.

- Animation
- Database
- Debugging and Logging
- DOM Manipulation
- Fonts
- Forms
- Graphical/Visualization
- GUI Related
- Pure JavaScript/AJAX
- String and Math Functions
- Unit Testing
- Web Application Related
- Other

All categories listed above include several different libraries. Another important point for the implementation is defining all libraries belonging to each category.

5.6. JS_Librarian Application Implementation

5.6.1. Input Data for the Application

The application accepts inputs in two ways as mentioned in previous chapters. One of the ways is entering a single website URL to the given textbox. Related interface can be seen in Figure 5.6.

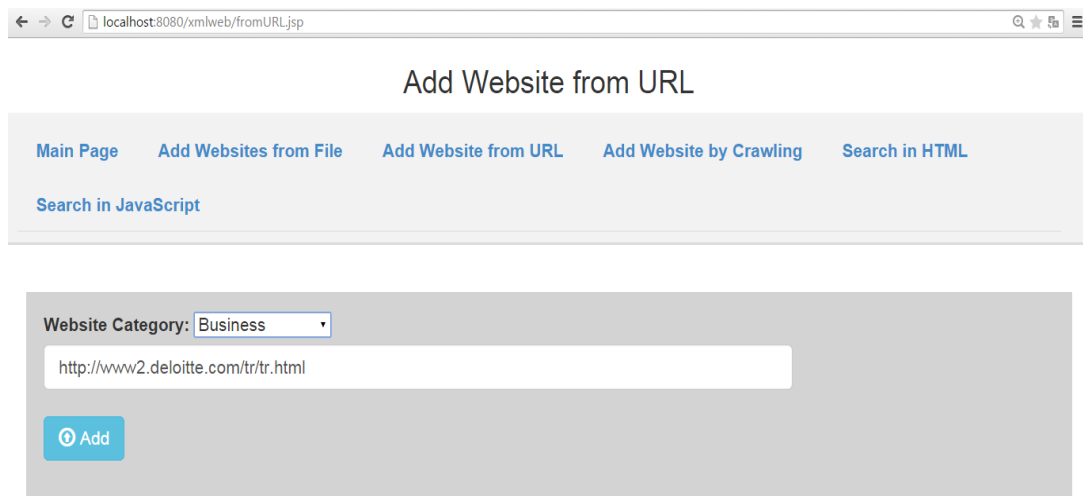


Figure 5. 6. Single URL as input

The other way is collecting websites by visiting the Alexa system. In this way, written Java code crawls the `http://www.alexa.com/topsites/category` address according to selected website category. The program makes a request to the page and gets the page source. Then it extracts the URLs included in this source. Interface related with this function can be seen in Figure 5.7. As seen in Figure 5.7 there appears textbox in the interface which accepts a page number. This page number corresponds to the page numbers of Alexa web page which lists the websites according to selected category. Alexa has nearly ten websites in one of its page. Our algorithm finds all website URLs in a page at once. If more websites is needed as input, other page numbers can be written on this textbox such as 2, 3, 4 etc. Then the program makes requests to these pages and get website URLs.

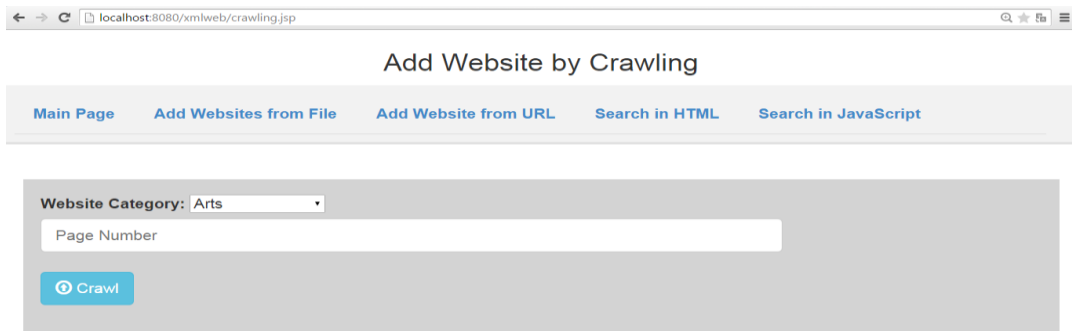


Figure 5. 7. Websites as input by crawling Alexa

5.6.2. Storing Basic Input Data in Solr

After a website URL is given to the system, the next process is storing basic data of this website to the Solr system. These basic data are ID, category, URL, and date. The main page of the application lists the websites that are stored in Solr. Related interface can be seen in Figure 5.9. To be able to store these basic data the schema of the Solr has been updated. These four fields have been added to the schema as seen in Figure 5.8.

```

<field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" />
...
<field name="webCategory" type="string" indexed="true" stored="true" omitNorms="true"/>
<field name="webURL" type="string" indexed="true" stored="true" omitNorms="true"/>
<field name="crawlingDate" type="date" indexed="true" stored="true" omitNorms="true"/>

<field name="javaScript" type="string" indexed="true" stored="true" omitNorms="true"/>
<field name="html" type="string" indexed="true" stored="true" omitNorms="true"/>

...
<field name="sku" type="text_en_splitting_tight" indexed="true" stored="true" omitNorms="true"/>
<field name="name" type="text_general" indexed="true" stored="true"/>
<field name="manu" type="text_general" indexed="true" stored="true" omitNorms="true"/>
<field name="cat" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="features" type="text_general" indexed="true" stored="true" multiValued="true"/>

```

Figure 5. 8. Solr schema xml

Java code that performs the process of insertion to Solr can be seen below.

```
String solrServerUrl = "http://localhost:8983/solr";
SolrServer server = new CommonsHttpSolrServer(solrServerUrl);

TimeZone tz = TimeZone.getTimeZone("UTC");
DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
df.setTimeZone(tz);

DateFormat id_dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
java.util.Date id_date = new java.util.Date();

SolrInputDocument solrDoc = new SolrInputDocument();
solrDoc.addField("id", URL + " - " + id_dateFormat.format(id_date));
solrDoc.addField("category", category_s);
solrDoc.addField("webURL", URL);
solrDoc.addField("crawlingDate", df.format(new java.util.Date()));
solrDoc.addField("html",html_fileName);
solrDoc.addField("javaScript",java_fileName);

try
{
server.add(solrDoc);
server.commit(); }
catch (SolrServerException e1)
{
// TODO Auto-generated catch block
e1.printStackTrace();
}
```

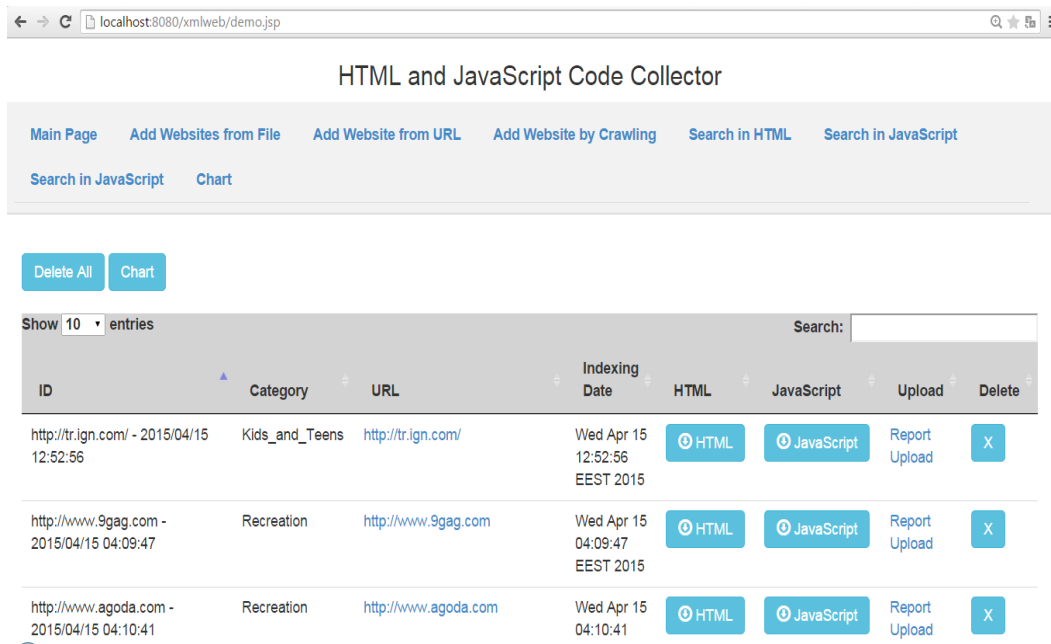


Figure 5. 9. The main page of the application

5.6.3. Parsing HTML and JavaScript Codes for the Input

After the insertion process to the Solr, HTML and JavaScript codes are extracted from the URL. For this process, `getHTMLDoc`, `getScript`, and `deep_js` functions have been developed.

`getHTMLDoc` function applies the `URLConnection` class in Java. `openConnection()` function of `URL` class opens the URL that is given as input and assigns the result to the object belonging to `URLConnection` class. Then this object applies the `getInputStream()` function to get the HTML code in of a website. At first, HTML codes are held in an object that is belonging to `BufferedReader` class. Then it is written to a document. For this process a directory is created with the `mkdir()` function belonging to `File` class. The name of the directory becomes the name of the website which is extracted from given URL by `substring()` function. After the directory has been created, HTML codes that are held in `BufferedReader` object are written to a file named with website with the ".html" suffix.

`getScript` function is slightly different from the `getHTMLDoc`. This function is a recursive function. It calls itself until it finds all the JavaScript codes between `<script>` and `</script>` tags. A directory named with the website name is created and then whenever a JavaScript code block is caught, it is written to a file named with website.

After `getScript` completes its process and stores all JavaScript codes in a file, `deep_js` function reads this file and finds the used JavaScript libraries and navigates through them to get codes. Finally, the function creates a file named as `deep_js` and stores all JavaScript codes in this file to be used in the next steps.

All these functions described above utilize `StringBuilder` Java class to hold gathered HTML or JavaScript codes that are included in websites. The reason for selecting `StringBuilder` class instead of `String` and `StringBuffer` classes for this process is mainly related to efficiency issue because these classes have some kinds of differences. These differences may have advantages or disadvantages according to purpose of use. For example storage area in `String` class is constant string pool, while it is heap for `StringBuffer` and `StringBuilder` classes. Additionally, modifiable feature is different for these classes as value of an object that is created from `String` class is immutable. It cannot be changed after its first initiation. It can only be overwritten and the old value still exists in string constant pool with a lost reference, so the memory can get larger as the new value is assigned to the object of the `String` class. However, `StringBuffer` and `StringBuilder` classes are mutable. The values of objects created from these classes can be modified without remaining any unreferenced values. `String` and `StringBuffer` classes are thread safe, while `StringBuilder` is not. Processes cannot use `String` and `StringBuffer` objects simultaneously. Finally, `StringBuffer` class has poor performance according to `String` and `StringBuilder` classes. `JS_Librarian` application uses for and while loops to collect HTML and JavaScript codes from websites and an object value can be changed in these loops. If objects from `String` class are used for these processes, numerous garbage values cover the memory. Consequently, `StringBuilder` class is applied in `JS_Librarian` application in order to supply better performance during the experiments.

5.6.4. Indexing HTML and JavaScript Codes with Lucene

After parsing HTML and JavaScript codes, files that are holding these codes are indexed with the Lucene. For this process SimpleFileIndexer class from Lucene library is applied. From this class index() function performs the indexing process. It takes three arguments for the call. These arguments are index file that indexes are stored, file that is going to be indexed and the suffix of this file. When the correct arguments are given to the function, indexes are created. Usage of index function in Java code can be seen below.

```
File indexDir = new File(klasor +"HTML/" + html_fileName +"/index/");
File dataDir = new File(klasor +"HTML/" + html_fileName );
String suffix = "html";

SimpleFileIndexer indexer = new SimpleFileIndexer();
try {
    int numIndex = indexer.index(indexDir, dataDir, suffix);
}
catch (Exception e)
{
    e.printStackTrace();
}
```

Indexes are created for both HTML and JavaScript codes. The code block resides in above are also applied for the JavaScript. After the HTML and JavaScript codes are indexed, any keyword can be searched both in HTML and JavaScript codes. For the search process Directory, IndexSearcher, QueryParser, Query, TopDocs, ScoreDoc, and Document classes belonging to Lucene library are used. The directory that index is stored is given as argument to the IndexSearcher class's constructor. Search() function of IndexSearcher class is used to search the given keyword. Parse() function of QueryParser class is used to create a query object. Parse() function receives the search keyword and assigns its result to Query object and Query object is used in Search() function. Then the websites which contain searched keyword are listed in the application. Figure 5.10 depicts the user interfaces of search processes among HTML documents.

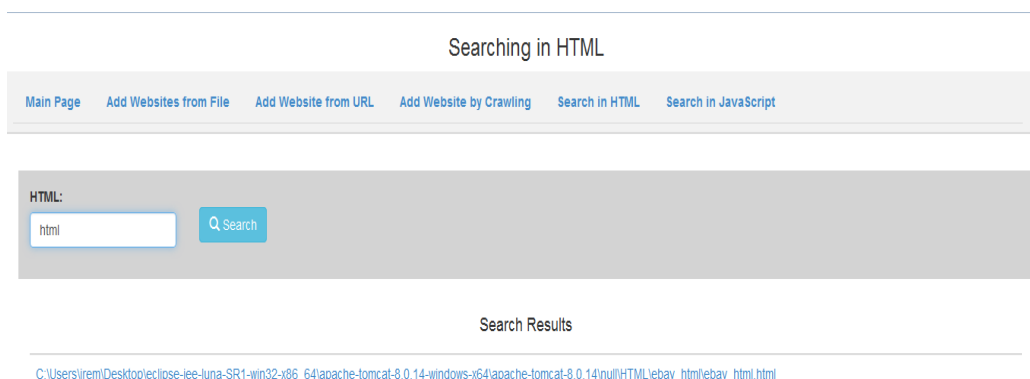


Figure 5. 10. Search a keyword in HTML codes

Figure 5.11 depicts the user interfaces of search processes among JavaScript documents.

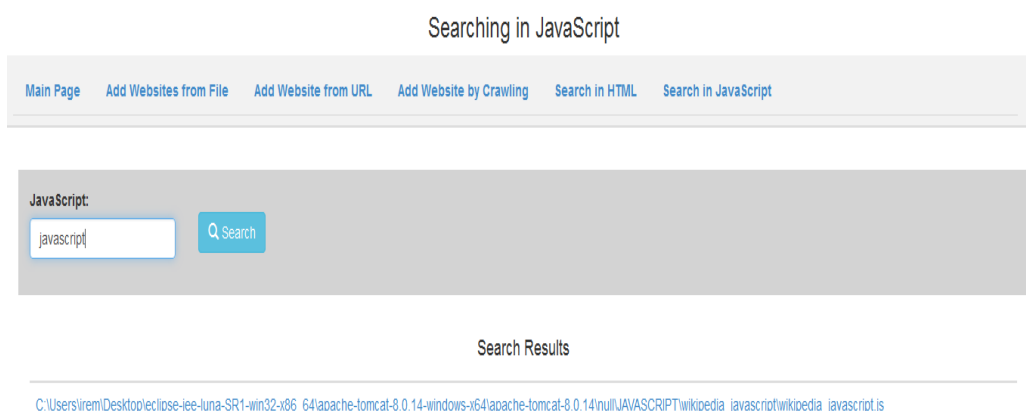


Figure 5. 11. Search a keyword in JavaScript codes

5.6.5. Finding Frequency of Specified Keywords

After recording basic website data in Solr, parsing HTML and JavaScript codes reside in website and indexing these codes with Lucene, the next step is finding the frequency of keywords that are belonging to JavaScript libraries. For these purpose first of all keywords belonging to each category have been decided. After that JavaScript codes of the focused website is read and each keyword is searched and count. At the end

of the JavaScript code each keywords are inserted to the MySQL database with their counts and their JavaScript categories as seen in Figure 5.12.

website_url	keyword	frequency	category_idh
http://www.deviantart.com/	jQuery	33387	8
http://www.forbes.com	jQuery	17277	8
https://www.tumblr.com/	Prototype	3468	8
https://www.tumblr.com/	Prototype	3468	10
http://www.medscape.com/	jQuery	3119	8
http://www.kayak.com	Prototype	2580	8
http://www.kayak.com	Prototype	2580	10
http://www.detroitnews.com	Prototype	2291	10
http://www.detroitnews.com	Prototype	2291	8
http://www.head-fi.org	Prototype	1861	8
http://www.head-fi.org	Prototype	1861	10
http://www.complex.com	html5	1805	12
http://www.codecademy.com/	Prototype	1782	8
http://www.codecademy.com/	Prototype	1782	10
http://www.sports.yahoo.com	Bootstrap	1651	9
http://www.npr.org/	Prototype	1615	8
http://www.npr.org/	Prototype	1615	10
http://www.mark.sandspencer.com	jQuery	1603	8
http://www.cnbc.com	jQuery	1533	8
http://www.cnn.com	jQuery	1523	8
http://www.foxnews.com/	jQuery	1484	8
http://www.hotels.com	Prototype	1288	10

136 rows fetched in 0.0312s (0.0878s)

Edit Apply Changes Discard Changes First Last Search

Figure 5. 12. Table that holds keyword frequency

5.6.6. Implementation of Naïve Bayes Algorithm

Naïve Bayes classification implementation is initiated with determining the training examples types. Before all, data that will be used as training example should be specified. After that training set should be gathered.

Naïve Bayes classification algorithm has been selected for the implementation of JS_Librarian application. The reason for selection of Naïve Bayes algorithm is that it comes into prominence with its applicability and performance. Additionally the independence of data processed in the application plays a role of this selection. The algorithm is based on probabilistic calculations and these calculations have been implemented in Java languages in order to determine JavaScript class of a given website.

The critical function of JS_Librarian application is the classification. All of the tasks that have been explained in previous sections provide data for the implementation of Naïve Bayes classification algorithm. Training data are constructed from these collected data. Nearly 150 websites have been inserted to the system for the training data construction.

Training process has been achieved based on the formula in the following:

$$P(a|b) = \frac{P(b|a) P(a)}{P(b)}$$

(5.6.6.1)

$$P(a|B) = P(b1|a) * P(b2|a) * ... * P(bn|a) * P(a) \quad (5.6.6.2)$$

(a) denotes the given class and (b) denotes the predictor in this formula. In Naïve Bayes algorithm it is assumed that the effect of a predictor like (b) on a given class like (a) is independent and the other predictors' values are not affected from (a).

$P(a|b)$ is the probability of class given predictor.

$P(a)$ is the probability of class.

$P(b|a)$ is the likelihood which is the probability of predictor given class.

$P(b)$ is the probability of predictor.

All the codes written for the Naïve Bayes classification algorithm are certainly based on these formulas and the algorithm is validated with the WEKA Java API by using Naïve Bayes classifier of WEKA.

5.6.7. Implementation of WEKA

The Naïve Bayes Classification algorithm written for this thesis is validated by using WEKA. During the validation feature vectors are built, classifier is trained, tested and finally used. All these functions are implemented in Java without

Explorer/Experimenter GUI. A classifier is created in WEKA and it can be used in programs to write a training/testing loop.

ARFF file that stands for Attribute-Relation file Format is commonly used for the representation of instances that are sharing specific set of attributes. This file has two parts one of the part is Header and the other part is Data. They hold header and data information. The name of the relation, a list of the attributes and their types are included in the Header of the ARFF file.

For the validation of the Naïve Bayes Classification algorithm first task is creating .arff file. For this purpose `weka.core.FastVector` and `weka.core.Attribute` classes of WEKA are applied. In this study there are one numeric and two nominal features and a nominal class. Numeric feature represents the total number of frequencies of specified keywords that are belonging to a JavaScript category. Two nominal features represent website URL and JavaScript categories of the keyword frequencies.

Objects from Attribute class are derived, with their names first of all. If the attributes are numeric there is no need to declare their values. However if they are nominal then their possible values must be declared. For these reason an object from `FastVector` class is created and all the possible values are inserted to this object.

After the .arff file format is mapped with the `FastVector` and `Attribute` classes, the next step is training the classifier. Instances are created for training and classifier is specified in training process. `weka.core.Instances`, `weka.core.Instance`, `weka.classifiers.Classifier`, `weka.classifiers.bayes.NaiveBayes` classes are applied in this section. Firstly an object from `Instances` class is derived with a name, with attributes created and given size. In this study this object has been created with name `weka_data`. Its attributes have been created previously as mentioned in above and the size has been specified with the count of row in the `vw_weka_data` view. Then an object from `Instance` class is derived to fill the training set. Java code of these processes is in the following.

```
// Create an empty training set
Instances trainingData = new Instances("weka_data", attributes, size);
// Set class index
trainingData.setClassIndex(3);

// Create the instance
String sql_instance = "SELECT * FROM vw_weka_data";
```

```

        PreparedStatement stmt_instance = db.conn.prepareStatement(sql_instance,
Statement.RETURN_GENERATED_KEYS);
        ResultSet rs_instance = stmt_instance.executeQuery(sql_instance);

        while(rs_instance.next())
        {

            Instance instance = new Instance(4);
            instance.setValue((Attribute) attributes.elementAt(0),
rs_instance.getString("website_url"));
            instance.setValue((Attribute) attributes.elementAt(1),
rs_instance.getString("JavaScript"));
            instance.setValue((Attribute) attributes.elementAt(2), rs_instance.getInt("freq"));
            instance.setValue((Attribute) attributes.elementAt(3), rs_instance.getString("Class"));

            // add the instance
            trainingData.add(instance);

        }

```

After the training data set is filled, classifier is selected from Classifier and Naïve Bayes classes for the model creating.

```

Classifier classifier = (Classifier)new NaiveBayes();
classifier.buildClassifier(trainingData);

```

Then test data is created for testing the classifier for this purpose Instances and Instance classes are applied again and the test data set is filled with the input data which is a website URL and its attributes like website_url, frequencies of keywords for each JavaScript category and JavaScript categories of each frequency value.

```

// Create an empty test data set
Instances test = new Instances("weka_data_test", attributes, 1);

// Set class index
test.setClassIndex(3);

String sql_url =
"SELECT website_url, j.name as JavaScript,sum(frequency) as freq
FROM tbl_website_data
inner join tbl_javascript_category j

```

```

on
j.idn = category_idn where website_url = "+URL+"
group by category_idn, website_url";
PreparedStatement stmt_url = db.conn.prepareStatement(sql_url,
Statement.RETURN_GENERATED_KEYS);

ResultSet rs_url = stmt_url.executeQuery(sql_url);
while(rs_url.next())
{
Instance intance = new Instance(4);
intance.setValue((Attribute) attributes.elementAt(0), rs_url.getString("website_url"));
intance.setValue((Attribute) attributes.elementAt(1), rs_url.getString("JavaScript"));
intance.setValue((Attribute) attributes.elementAt(2),
rs_url.getInt("freq"));

//add the instance
test.add(intance);

}

```

The classifier can be tested with the test data created above.

```

double label = classifier.classifyInstance(test.instance(0));
test.instance(0).setClassValue(label);
System.out.println(test.instance(0));

```

CHAPTER 6

EXPERIMENTAL RESULTS

In this thesis the problem was collection and classification of JavaScript libraries included in websites. For the solution to that problem JS_Librarian application has been developed. Implementation of this application has been explained in previous chapter. This chapter expresses the results gained from JS_Librarian application. Experimental descriptions are explained first of all and then experiment with JS_Librarian application and experiment with WEKA are mentioned. Finally the results of the experiments are depicted in Appendix B.

6.1. Experiment Description

Experiment of this study is based on basically collection and classification processes. HTML and JavaScript codes of websites have been collected in the scope of information retrieval concept. It has been inferred that Jsoup library gets the HTML codes successfully and from collected HTML code JavaScript codes and used libraries can be extracted and the codes gathered can be stored in a specified file.

As explained in the implementation section, main attributes of a website such as IDN, URL, category, and crawling date are held in Solr system. As a result it is seen that information retrieval from Solr system is easy with its query language and interaction with Lucene is available. On the other hand, with Lucene indexes keyword search has been performed successfully. Only one index has been created for all of the documents and searching has been performed on this index.

While going through the solution of our problem, the critical point was detecting the JavaScript libraries used in websites. For this reason frequencies of specified words are calculated and then stored in MySQL database system because its flexibility was essential beside to Solr for the classification process.

In the classification phase Naïve Bayes classification algorithm have been applied in two different ways. One of them is writing the algorithm by hand according to its

formulation and the other way is applying the algorithm by using WEKA. In order to get more beneficial results, classification has been applied three times, because first two classes are same for all kinds of websites. Thus, three classes for each input are created.

At the end when enough number of websites have been processed by the JS_Librarian system, a general reports about the JavaScript classes according to website categories have been retrieved. Experimental results are explained in the following sections.

6.2. Experiment with JS_Librarian Application

JS_Librarian application collects the websites as input in two ways. One of these ways is entering a single URL, so any website can be inserted to the system in this way. The system stores websites with their categories. Therefore, website category should be chosen for the each URL in this way in order to get more useful reports at the end. On the other hand the other way is collective input entrance. In this way URLs are entered to the system by crawling the Alexa because Alexa categorizes the websites according to their content, so from each category numerous websites can be obtained.

In this experiment, websites have been collected for sixteen categories by using Alexa. From each category nearly ten URLs have been entered to the system. All the processes that are listed in order as storing basic website attributes to Solr, parsing the HTML and JavaScript, finding the frequencies of specified keywords, determining the JavaScript class of given website with formula based written Naïve Bayes algorithm, and determining the JavaScript class of given website with WEKA are applied to each URL obtained from Alexa. These processes are implemented in a loop until the specified numbers of websites have been inserted to the system.

In the case of sufficient numbers of websites have been processed in the system, views created MySQL database are applied for getting the analyses. One of our objectives while preparing this thesis is to be able to extract the information which indicates that websites belonging to specific categories mostly apply the JavaScript libraries that are belonging to specific categories.

Our experimental results illustrate that JavaScript libraries that are belonging DOM Manipulation category are most used libraries in enormous number of websites. This category includes the most popular libraries. These are jquery and prototype. On

the other hand, libraries belonging to Web Application Related category are the second most used libraries. These categories are varying only on third and fourth categories as seen in Table 6.1.

Table 6. 1. Class of each website category

<u>Website Category Name</u>	<u>Class 1</u>	<u>Class 2</u>	<u>Class 3</u>	<u>Class4</u>
Art	DOM	Web A. R.	<u>Graphical/Visualization</u>	<u>Modernizr</u>
Business	DOM	Web A. R.	<u>Modernizr</u>	<u>Graphical/V</u>
<u>Computer</u>	DOM	Web A. R.	<u>Modernizr</u>	<u>Graphical/V</u>
Game	DOM	Web A. R.	<u>Graphical/Visualization</u>	<u>Modernizr</u>
<u>Health</u>	DOM	Web A. R.	<u>Modernizr</u>	<u>Graphical/V</u>
Home	DOM	Web A. R.	<u>Pure JavaScript/Ajax</u>	<u>Graphical/V</u>
<u>Kids and Teens</u>	DOM	Web A. R.	<u>Modernizr</u>	<u>GUI Related</u>
News	DOM	Web A. R.	<u>Modernizr</u>	<u>Graphical/V</u>
<u>Recreation</u>	DOM	Web A. R.	<u>Modernizr</u>	<u>Graphical/V</u>
Reference	DOM	Web A. R.	<u>Graphical/Visualization</u>	<u>GUI Related</u>
<u>Regional</u>	DOM	Web A. R.	<u>Modernizr</u>	<u>Graphical/V</u>
<u>Science</u>	DOM	Web A. R.	<u>Modernizr</u>	<u>GUI Related</u>
<u>Shopping</u>	DOM	Web A. R.	<u>GUI Related</u>	<u>Modernizr</u>
<u>Society</u>	DOM	Web A. R.	<u>Modernizr</u>	<u>Graphical/V</u>
<u>Sport</u>	DOM	Web A. R.	<u>Graphical/Visualization</u>	<u>Modernizr</u>
World	DOM	Web A. R.	<u>Pure JavaScript/Ajax</u>	<u>Graphical/V</u>

6.3. Experiment with WEKA

In this thesis we have implemented Naïve Bayes classification method and we have chosen WEKA API. For the quality measurements of the implementation we have used 10 folds cross validation. The objective of cross validation is to estimate the accuracy of a predictive model. Cross validation implementation includes partitioning a sample data into subsets in a loop. One of these subsets that is called as training set is used for the analysis and the other that is called as testing set is used for the validation of the analysis. Several rounds of cross validation are performed based on different partitions and results are averaged. Related code block can be seen in below.

```
for (int i = 0; i<10; i++)
{
    System.out.println("Iteration " + i );
    Instances trainingInstances = new Instances(instances);
    Instances testingInstances = new Instances(instances, begin, (end -
        begin));

    for (int j = 0; j < (end-begin); j++)
    {
        trainingInstances.delete(begin);
    }
    NaiveBayes nb = new NaiveBayes();
    nb.buildClassifier(trainingInstances);

    Evaluation ev = new Evaluation(testingInstances);
    ev.evaluateModel(nb, testingInstances);

    System.out.println("P: " + ev.precision(1));
    System.out.println("R: " + ev.recall(1));
    System.out.println("F: " + ev.fMeasure(1));
    System.out.println("E: " + ev.errorRate());

    precision += ev.precision(1);
    recall += ev.recall(1);
    fmeasure += ev.fMeasure(1);
    error += ev.errorRate();

    begin = end + 1;
    end += size_cv;

    if(i==9)
    {
        end = instances.numInstances();
    }
}
```


Results gained from cross validation have been compared with the results gained from WEKA API. Training data set size has been reached to 3330 rows with nearly 310 websites. Experimental results are gained from WEKA API based on these 310 website with their keywords frequencies data set and we have gotten a result with nearly 99 % correctness. An example result gained from WEKA API is seen in Figure 6.1.

```

http://www.estatesales.net/, 'STRING AND MATH FUNCTIONS',0,?
http://www.estatesales.net/, FONTS,0,?
http://www.estatesales.net/, 'DEBUGGING AND LOGGING',0,?
http://www.estatesales.net/, 'DOM MANIPULATION',440,?
http://www.estatesales.net/, 'GUI RELATED',0,?
http://www.estatesales.net/, 'WEB APPLICATION RELATED',220,?
http://www.estatesales.net/, 'PURE JAVASCRIPT/AJAX',110,?
http://www.estatesales.net/, OTHER,420,?
http://www.estatesales.net/, 'UNIT TESTING',0,?

Correctly Classified Instances      3330      99.2844 %
Incorrectly Classified Instances    24        0.7156 %
Kappa statistic                    0.9673
Mean absolute error                 0.0814
Root mean squared error             0.1699
Relative absolute error              234.5934 %
Root relative squared error         129.8289 %
Total Number of Instances           3354

62.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|3.0|0.0|0.0|0.0|0.0|
0.0|13.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|
0.0|0.0|13.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|
0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|
0.0|0.0|0.0|0.0|13.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|
0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|
0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|
0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|2951.0|0.0|0.0|0.0|0.0|
0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|1.0|3.0|0.0|9.0|0.0|0.0|
0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|4.0|0.0|61.0|0.0|0.0|0.0|
0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|2.0|0.0|0.0|76.0|0.0|0.0|
0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|5.0|0.0|0.0|0.0|138.0|0.0|
0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|0.0|

```

Figure 6. 1. An example of WEKA result

CHAPTER 7

CONCLUSION

In this thesis collection and classification of JavaScript libraries included in websites was the focused issue. JavaScript libraries available today are found out and distinctive features are determined for each library. Libraries are core factors for this thesis after the information retrieval process is completed. Alexa is the helpful site for us to collect websites codes and Naïve Bayes classification algorithm is facilitate our classification problems. Our intent is to be able to see if these huge amounts of libraries can be classified to facilitate the usage for web development processes.

In this study, JavaScript libraries have been studied for their classification. However, when we applied Naïve Bayes algorithm on our data for the first time, we realized that libraries belonging to DOM Manipulation category are commonly used among the all kinds of websites. As known JQuery library which is a popular library that facilitate web developer's studies while writing JavaScript codes, is belonging to DOM Manipulation category. Prototype library is also one of the most used libraries in websites like JQuery. Thus we have not got any distinctive result about the relationship between website categories and JavaScript library categories in the first time. Then we applied Naïve Bayes algorithm second time. At this time we realized that all kinds of websites are using libraries belonging to Web Application Related category. This result was also not distinctive. When we applied the algorithm third times we realized that the used libraries are varying according to website categories. Website categories and mostly used libraries' graphical representation can be seen in Appendix B.

We applied Naïve Bayes classification algorithm for the solution to our problem. However in the future works, other classification can be implemented with other machine language algorithms and the experimental results can be compared. Additionally, test data set size can be increased to be able to extract other meaningful information from row data stored in the system.

REFERENCES

- [1] P. J. Denning, "The ARPANET after twenty years," vol. 77, pp. 530–535, 1989.
- [2] D. Walden, "The arpanet IMP program: Retrospective and resurrection," IEEE Ann. Hist. Comput., vol. 36, no. 2, pp. 28–39, 2014.
- [3] P. Relations and L. Vegas, "Digital Media Center," Main, no. 713.
- [4] W. W. Cohen and W. Cohen, "Learning and discovering structure in web pages," IEEE Data Eng. Bul, 2003.
- [5] <http://www.sitepoint.com/web-foundations/css/> (accessed date : 2 January 2015)
- [6] Dūūna K., Analysis of Node.js platform web application security. TALLINN UNIVERSITY OF TECHONOLGY, 2012.
- [7] Flanagan, David; Ferguson, Paula (2006). JavaScript: The Definitive Guide (5th ed.). O'Reilly & Associates. ISBN 0-596-10199-6.
- [8] Shang ,Wentao; Thompson, Jeff; Cherkaoui , Meki; Burke , Jeff; Zhang , Lixia. NDN. *JS: A JavaScript Client Library for Named Data Networking*. Department of Computer Science, UCLA.
- [9] Viejo, M., *JavaScript Tips for ASP.NET*. Cambria.
- [10] Masek R., *Object-oriented and Functional Programming in JavaScript*. Prague, 2012.
- [11] Aresh A., *Real-Time Voice Chat Realized in JavaScript*. Luleå University of Technology
- [12] Polyvyanyy, A., *Evaluation Design of Information Retrieval System with eTVSM Specific Extensions*. University of Potsdam, Germany.

- [13] Robertson, S., *The methodology of Information Retrieval Experiment*.
- [14] Hiemstra, D., *Information Retrieval Models*. University of Twente.
- [15] C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.
- [16] <http://www.ee.columbia.edu/~vittorio/Lecture-2.pdf> (accessed date : 2 May 2015)
- [17] Sørensen, A., *Evaluating the use of Learning Algorithms in Categorization of Text*, Norwegian University of Science and Technology, 2012.
- [18] Wagstaff K., *Machine Learning that Matters*. California Institute of Technology, Pasadena, 2012.
- [19] Mitchell T., *The Discipline of Machine Learning*. Carnegie Mellon University Pittsburgh, 2006.
- [20] Markov Z., *Probabilistic Reasoning with Naïve Bayes and Bayesian Networks*. 2007.
- [21] http://scikit-learn.org/stable/modules/naive_bayes.html (accessed date : 10 March 2015)
- [22] Discriminative Multinomial Naïve Bayes for Network Intrusion Detection, 2000.
- [23] Jebara, T., *Discriminative, Generative and Imitative Learning*. McGill University, 1996
- [24] VEENHOF N., *Improving Acquia Search and the Apache Solr Search Integration Drupal Module*. BarcelonaTech, 2011 – 2012
- [25] Addagada S., *Indexing and Searching Document Collections using Lucene*. University of New Orleans, 2007.
- [26] <http://www.alexa.com/about> Alexa. Retrieved January 18, 2015.

- [27] Smola A., Vishwanathan S., *Introduction to Machine Learning*. Cambridge.
- [28] Domingos P., *A few Useful Things to Know about Machine Learning*. University of Washington.
- [29] <http://www.javatpoint.com/jsoup-examples> (accessed date : 27 December 2014).
- [30] <http://www.solrtutorial.com/basic-solr-concepts.html> (accessed date : 10 November 2014).
- [31] Bouckaert R., Frank E., Hall M., Holmes G., Pfahringer B., Reutemann, Witten I., *WEKA – Experiences with a Java Open-Source Project*. University of Waikato, New Zealand.

APPENDIX A

CATEGORIES OF JAVASCRIPT LIBRARIES

Animation

Libraries belonging to Animation category allow developing web applications with wide range of animations and effects. Most popular libraries of this category are listed below.

- [script.aculo.us](#)
- [Pixastic](#)
- [moo.fx](#)
- [scripty2](#)
- [jsAnim](#)
- [C3DL](#)
- [GX](#)
- [Scriptio](#)
- [JSTweener](#)

Database

Libraries belonging to Database category allow performing database operations for the web applications. Most popular libraries of this category are listed below.

- [Taffy DB](#)
- [ActiveRecord.js](#)

Debugging and Logging

Libraries belonging to Debugging and Logging category helps debugging the JavaScript codes to find syntax and logic errors for the web applications. Most popular libraries of this category are listed below.

- Blackbird
- NitobiBug
- Firebug Lite

DOM Manipulation

Libraries belonging to DOM Manipulation category supplies function to change DOM for the web applications. Most popular libraries of this category are listed below.

- Dojo
- jQuery
- Midori
- MooTools
- Prototype
- YUI

Fonts

Libraries belonging to Fonts category allow using several different fonts in web applications. Most popular libraries of this category are listed below.

- typeface.js
- Cufón

Forms

Libraries belonging to Forms category supply several different forms in web applications. Most popular libraries of this category are listed below.

- wForms
- Validanguage
- LiveValidation
- Yav
- qForms
- formreform

Graphical/Visualization

Libraries belonging to Graphical/Visualization category allow several different graphics in web applications. Most popular libraries of this category are listed below.

- Intense
- FusionCharts
- D3
- JavaScriptInfoVisToolkit
- Kinetic
- Processing
- Raphaël
- SWFObject
- Three
- EaselJS
- Fancybox
- retina

GUI Related

Libraries belonging to GUI Related category allow several different user interfaces in web applications. Most popular libraries of this category are listed below.

- App
- AngularUI
- DHTMLX
- Sencha_ext
- jQueryUI
- SAPOpenUI5
- Qooxdoo
- GooglePolymer
- SmartClient
- TelerikKendoUI
- Webix
- WinJS
- Bootstrap
- ZURBFoundation

Pure JavaScript/AJAX

Libraries belonging to Pure JavaScript/AJAX category are listed below.

- Lazy
- GoogleClosureLibrary
- Joose
- jsPHP
- MicrosoftAjaxlibrary
- MochiKit
- PDF_js
- Rico

- SocketIO
- Spryframework
- Underscorejs

String and Math Functions

Libraries belonging to String and Math Functions category allow several string and math functions for web applications. Most popular libraries of this category are listed below.

- Date
- Sylvester

Unit Testing

Libraries belonging to Unit Testing category allow several functions for testing web applications. Most popular libraries of this category are listed below.

- Gremlins
- Jasmine
- Unitjs
- QUnit
- Mocha

Web Application Related

Libraries belonging to Web Application Related category are listed below.

- AngularJS
- Backbone
- Cappuccino
- Chaplin

- Echo
- Ember
- Enyo
- GoogleWebToolkit
- JavaScriptMVC
- Knockout
- MooTools
- Prototype JavaScript Framework
- RialtoToolkit
- SproutCore
- WakandaFramework

Other

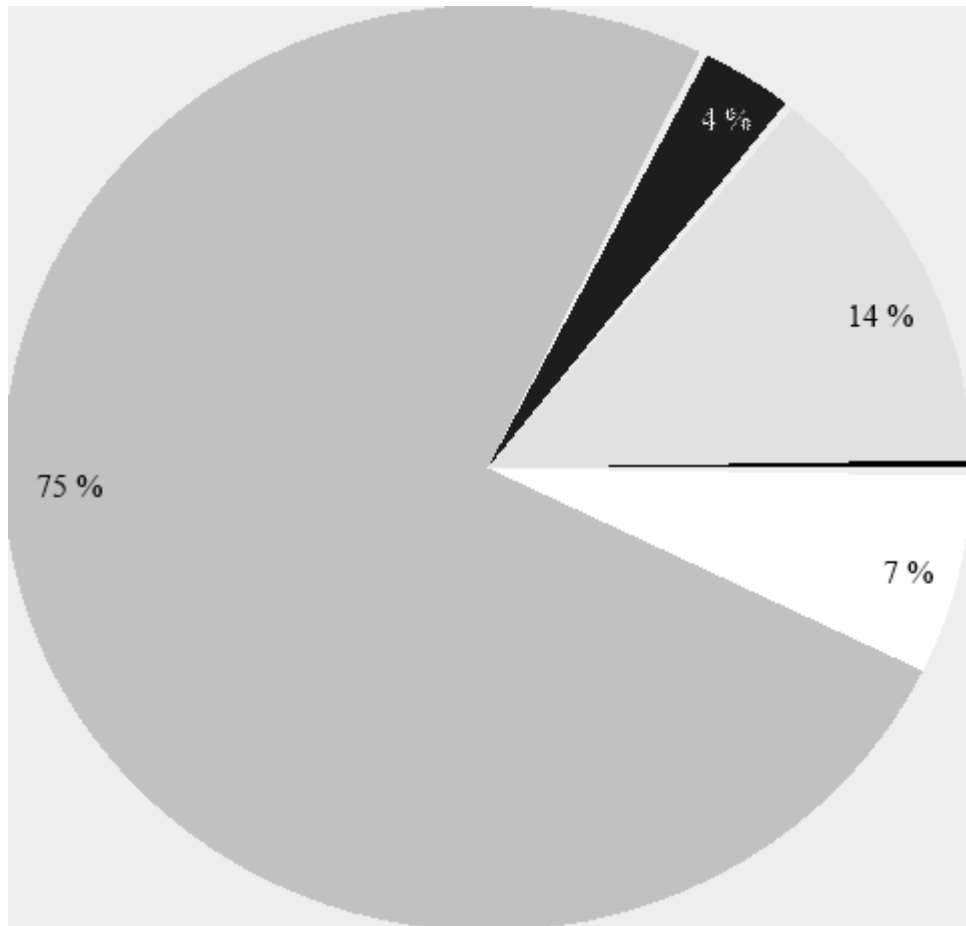
Libraries belonging to Other category are listed below.

- html5
- modernizr

APPENDIX B

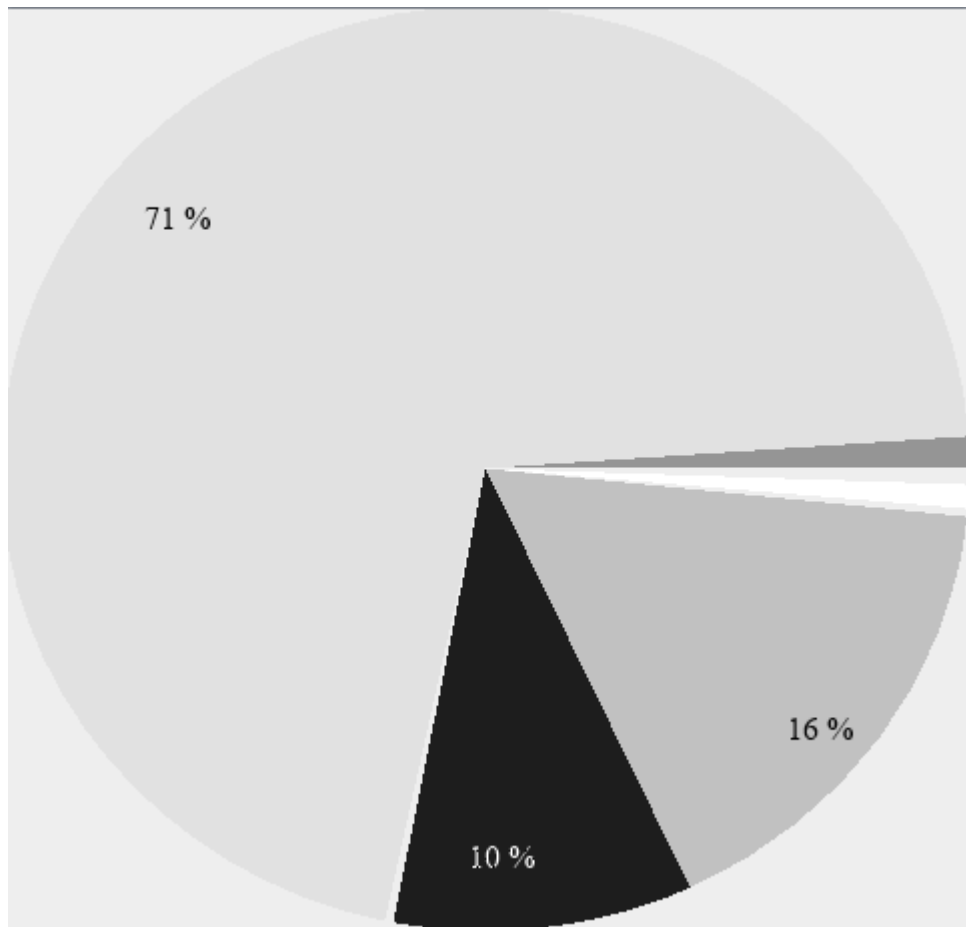
GRAPHICAL REPRESENTATION OF RESULTS TAKEN FROM JS_LIBRARIAN APPLICATION

ARTS



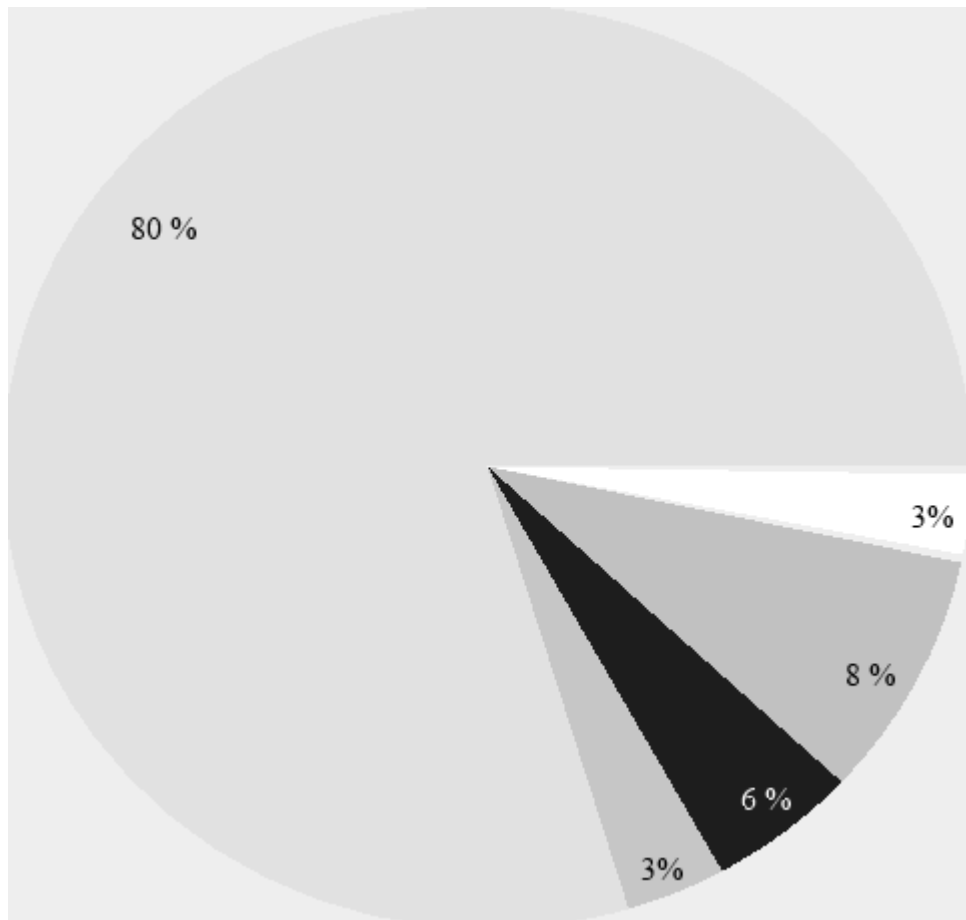
- Graphical/Visualization 75 %
- Modernizr 14 %
- Pure JavaScript/Ajax 7 %
- Web Application Related 4 %

BUSINESS



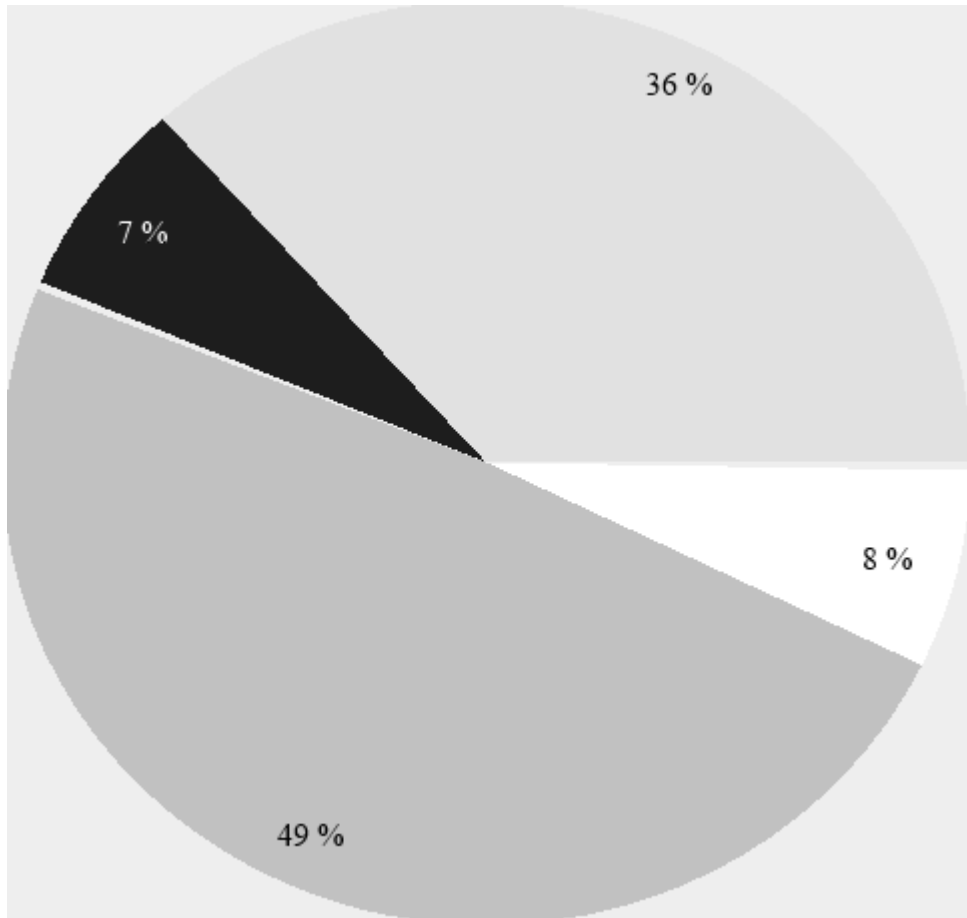
- Modernizr 71 %
- Graphical/Visualization 16 %
- Web Application Related 10 %

COMPUTER



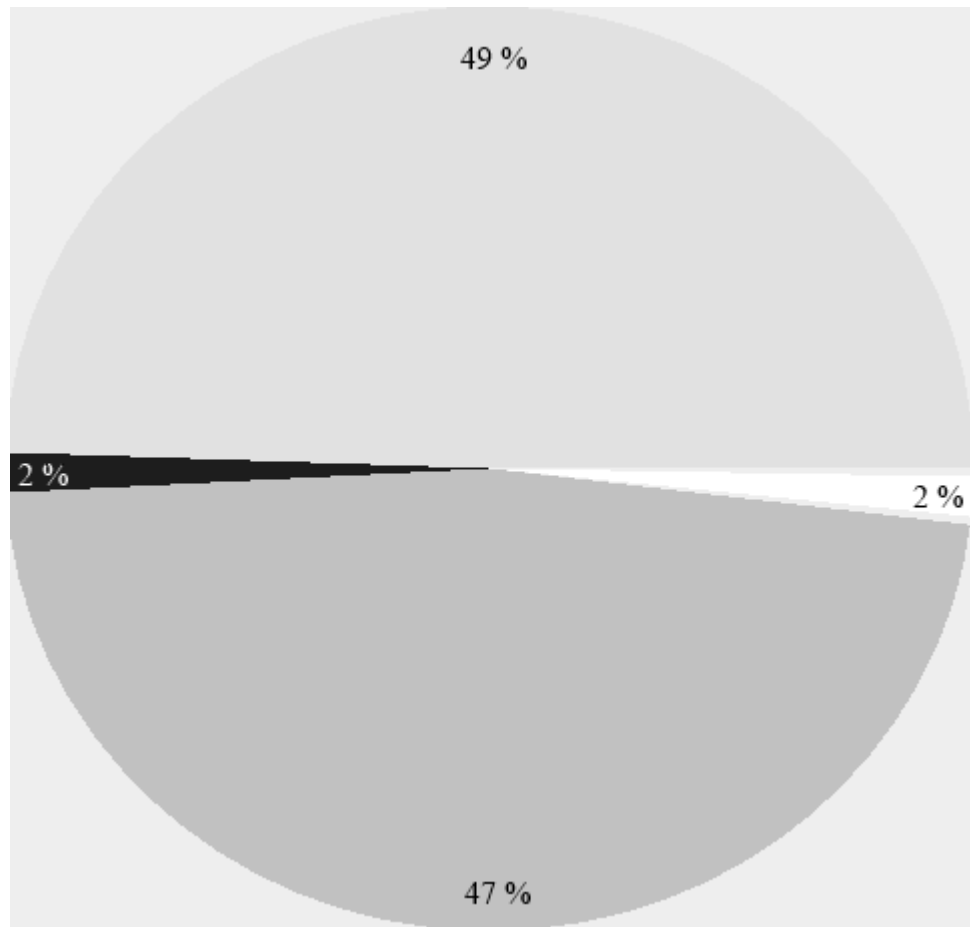
- Modernizr 80 %
- Graphical/Visualization 8 %
- Web Application Related 6 %
- Debugging and Logging 3 %
- Pure JavaScript/Ajax 3 %

GAME



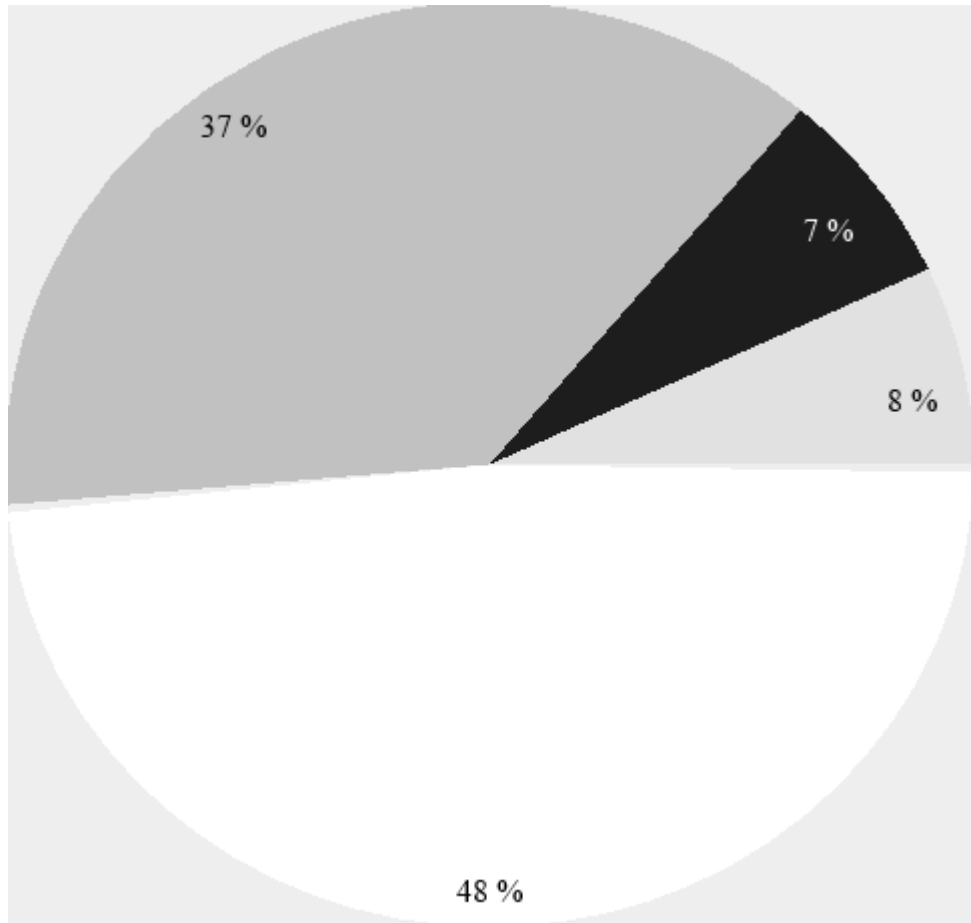
- Modernizr 36 %
- Graphical/Visualization 49 %
- Pure JavaScript/Ajax 8 %
- Web Application Related 7 %





HEALTH



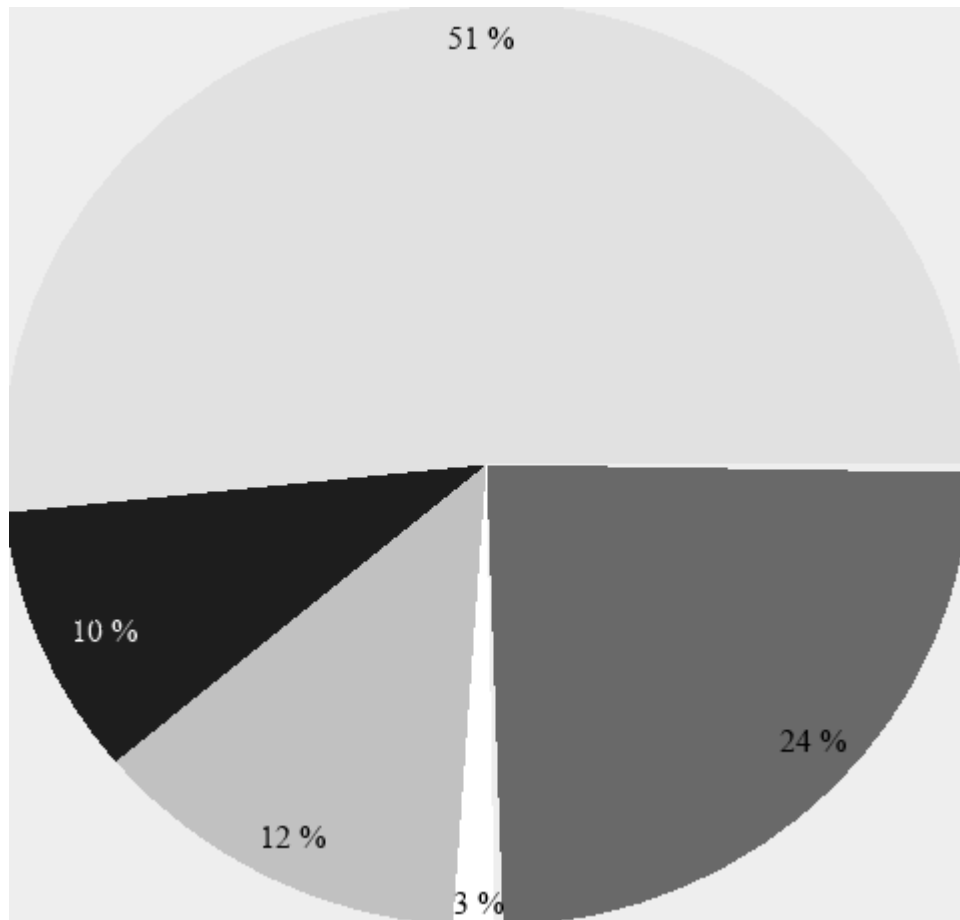
- Modernizr 49 %
- Graphical/Visualization 47 %
- Web Application Related 2 %
- Pure JavaScript/Ajax 2 %

HOME



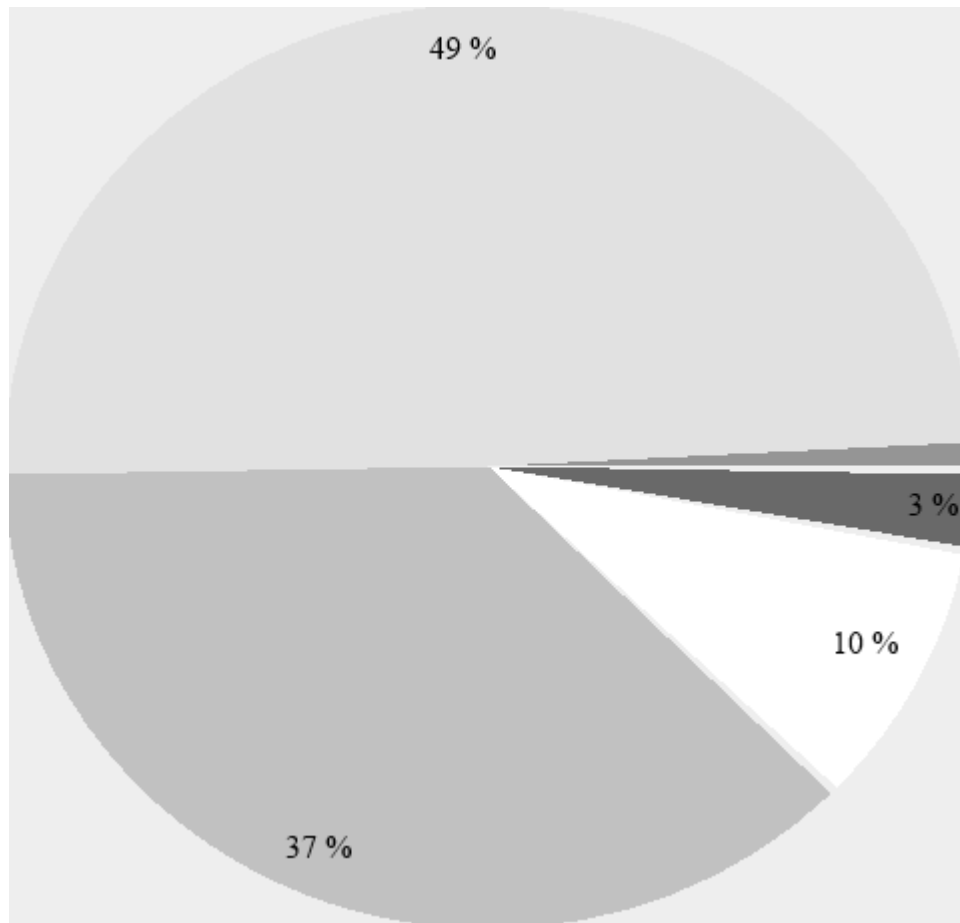
-  Graphical/Visualization 37 %
-  Pure JavaScript/Ajax 48 %
-  Modernizr 8 %
-  Web Application Related 7 %

KIDS AND TEENS



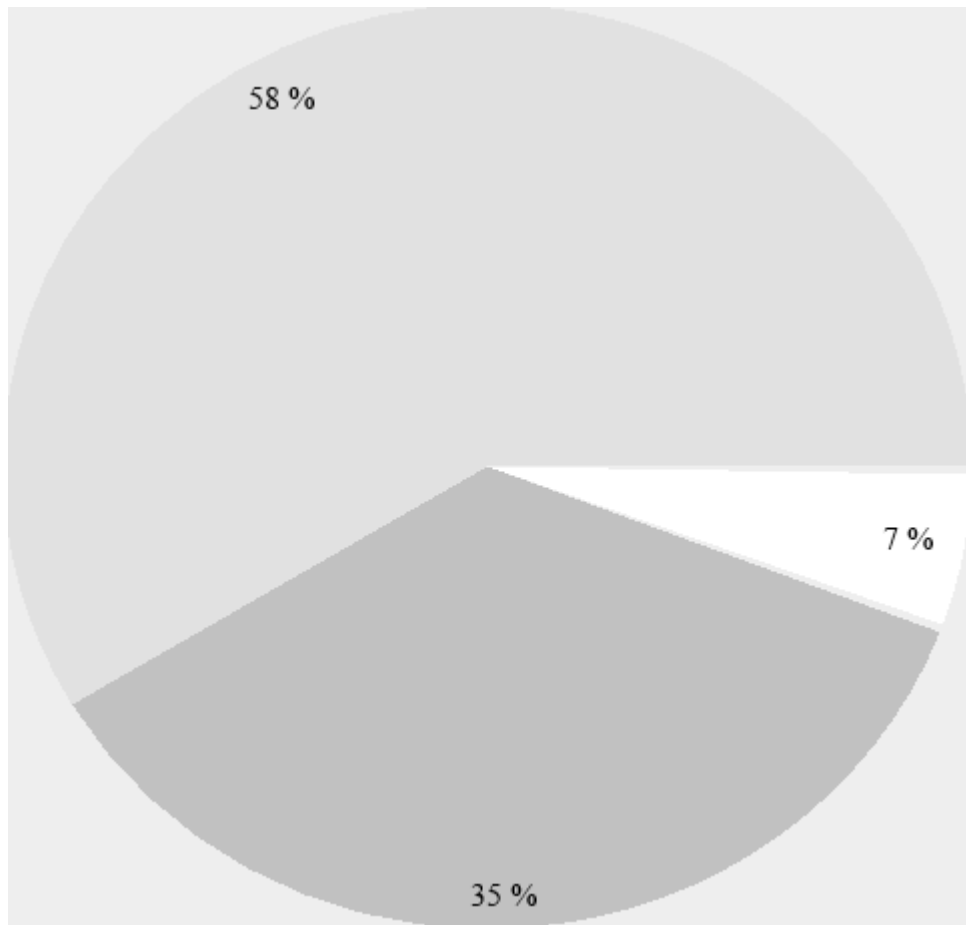
- Modernizr 51 %
- GUI Related 24 %
- Graphical/Visualization 12 %
- Web Application Related 10 %
- Pure JavaScript/ajax 3 %

NEWS



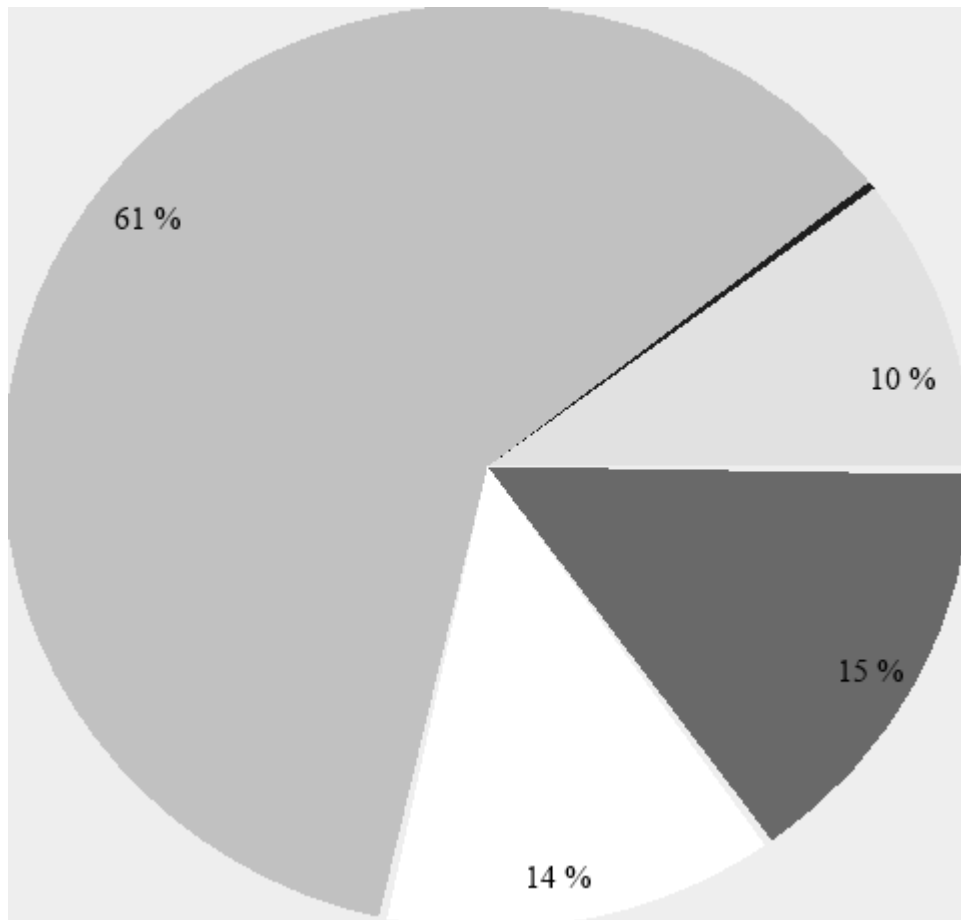
- Modernizr 49 %
- Graphical/Visualization 37 %
- Pure JavaScript/Ajax 10 %
- GUI Related 3 %

RECREATION



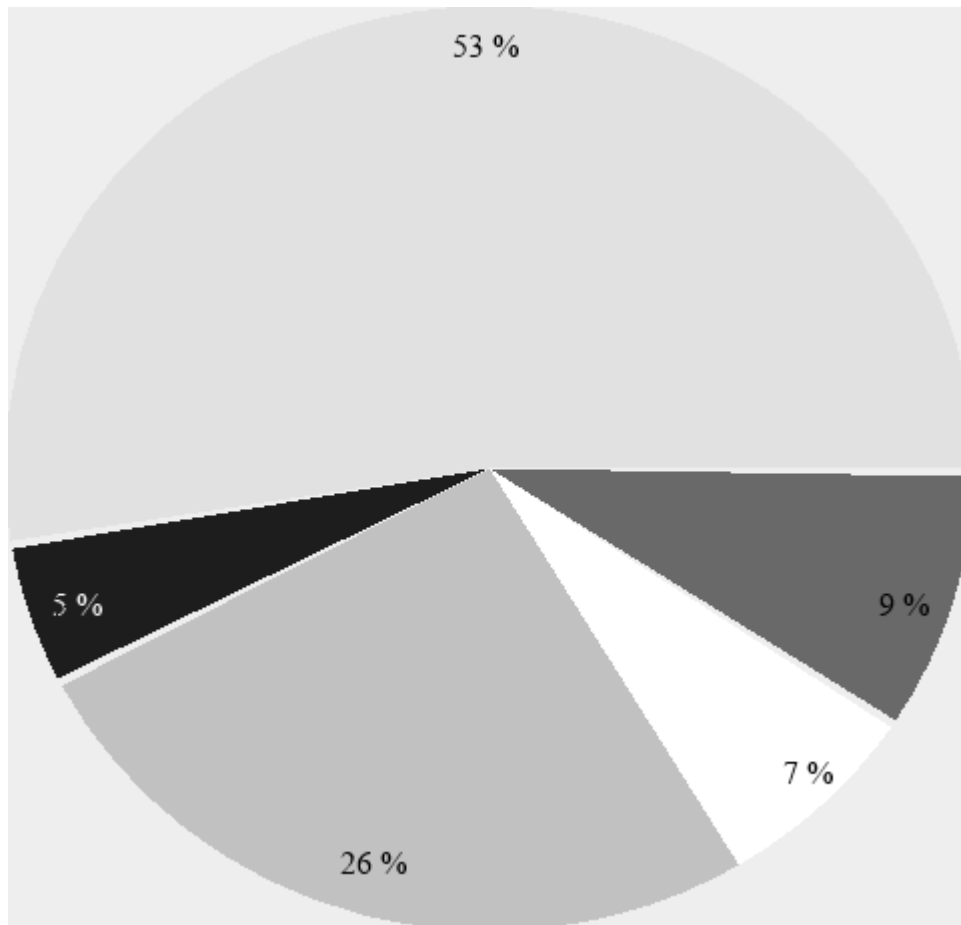
- Modernizr 58 %
- Graphical/Visualization 35 %
- Pure JavaScript/Ajax 7 %

REFERENCE



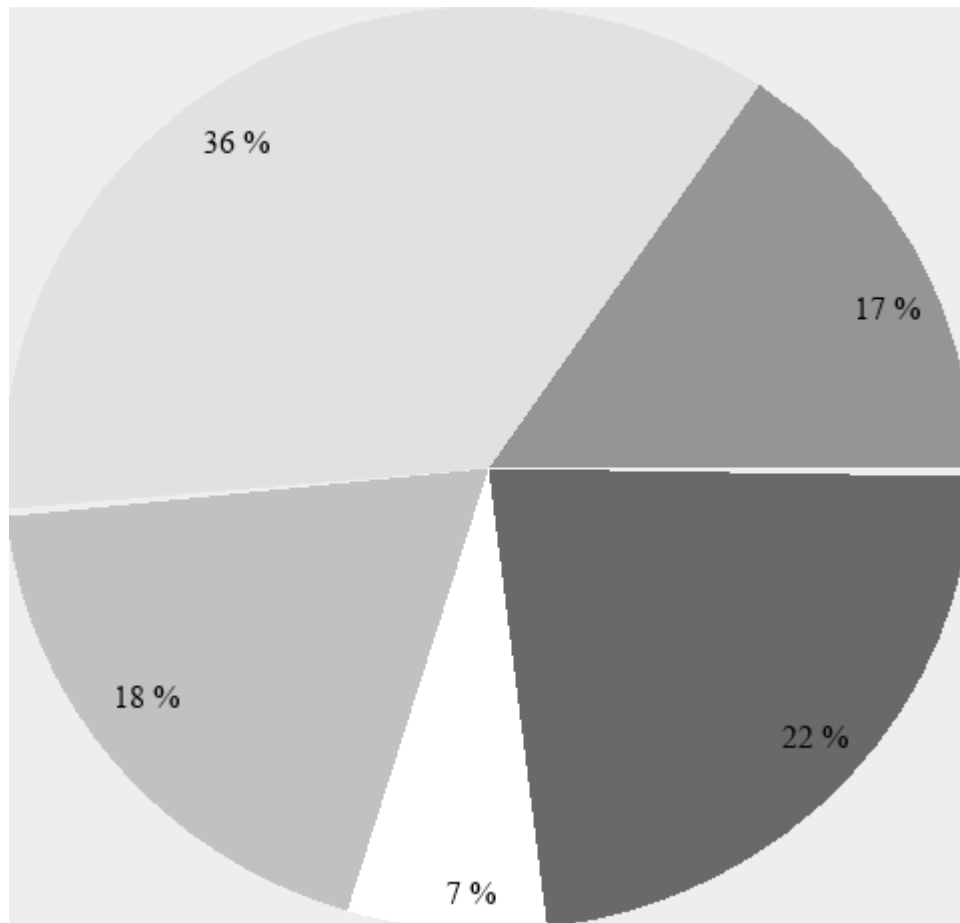
- Graphical/Visualization 61 %
- GUI Related 15 %
- Pure JavaScript/Ajax 14 %
- Modernizr

REGIONAL



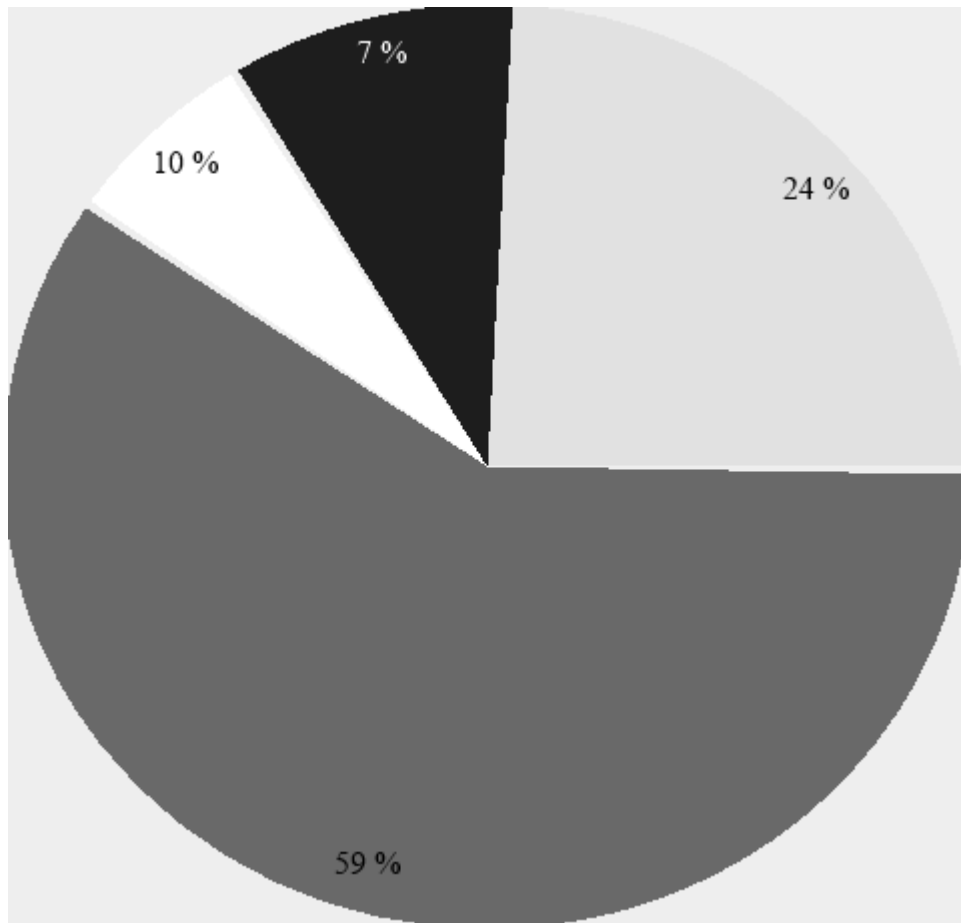
- Modernizr 53 %
- Graphical/Visualization 26 %
- GUI Related 9 %
- Pure JavaScript/Ajax 7 %
- Web Application Related 5 %

SCIENCE



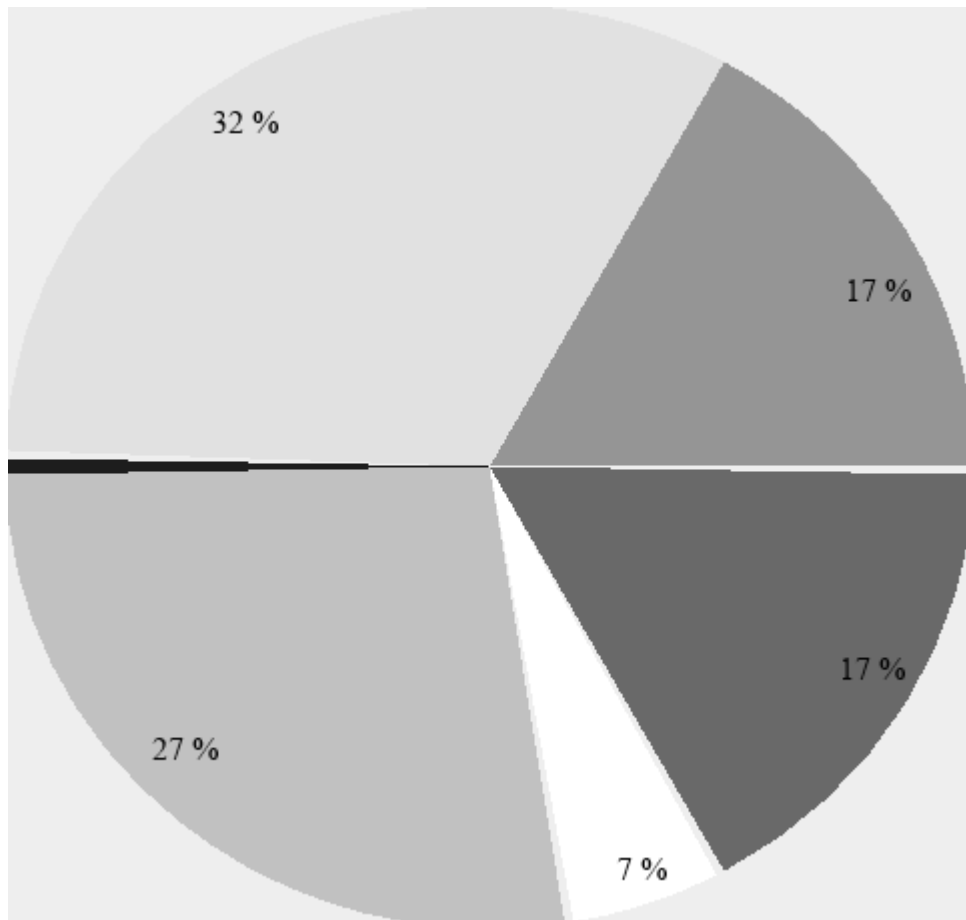
- Modernizr 36 %
- GUI Related 22 %
- Graphical/Visualization 18 %
- DOM Manipulation 17 %
- Pure JavaScript/Ajax 7 %

SHOPPING



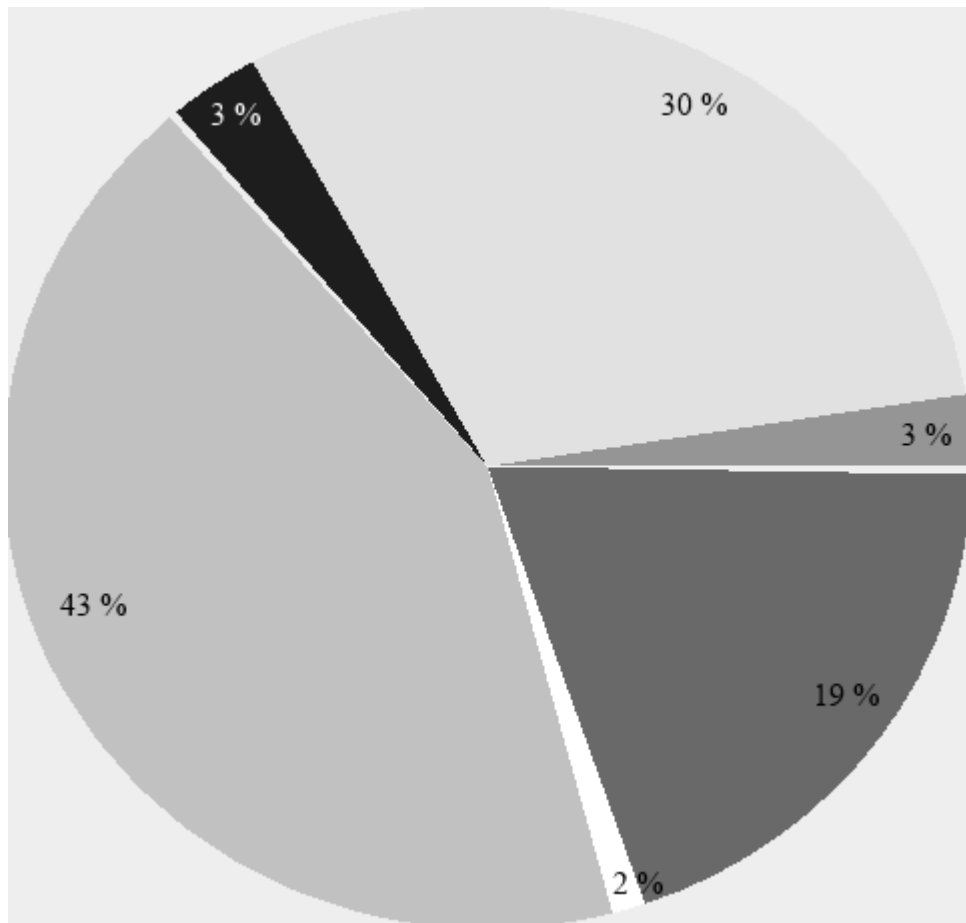
- Modernizr 24 %
- GUI Related 59 %
- Pure JavaScript/Ajax 10 %
- Web Application Related 7 %

SOCIETY



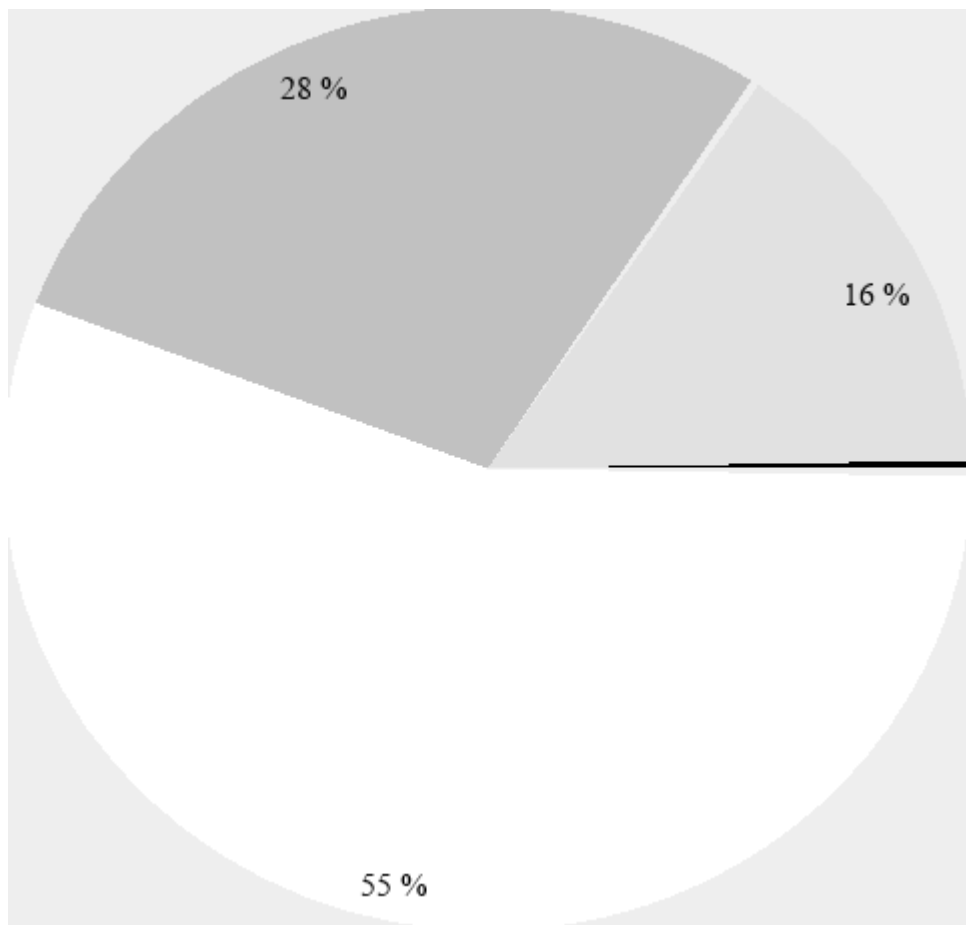
- Modernizr 32 %
- Graphical/Visualization 27 %
- DOM Manipulation 17 %
- GUI Related 17 %
- Pure JavaScript/ajax 7 %

SPORT



- Modernizr 30 %
- Graphical/Visualization 43 %
- GUI Related 19 %
- DOM Manipulation 3 %
- Web Application Related 3 %
- Pure JavaScript/Ajax 2 %

WORLD



- Pure JavaScript/Ajax 55 %
- Graphical/Visualization 28 %
- Modernizr 16 %