TABLE OF CONTENTS

1. Introduction

1.1. Abstract

Suggestions4You aims to understand the semantic meaning behind a person's web-browsing activities. In web-browsers used today, there is a need to inherently learn and understand the interests of a user so as to provide a richer browsing experience. The project aims to fulfill this by using the following components –

- Browser Add-Ons to act as a middleware between a user and his/her web-browser.
- APIs for semantic analysis of web content
- Ranking Algorithms to organize the learned information of user tastes.

The project shall lead to the development of algorithms and finally, a software which shall learn the general interest of the browsers and based on this information gathered, shall suggest related links to provide a meaningful user experience.

1.2. Problem Statement

Web-browsing is an integral part of every person's daily activities these days. A property of web-browsing is that the content viewed via a web-browser is local to the user's (person browsing) tastes and preferences. Such an interest may be a result of an activity the user is generally involved in. For example, a computer engineering student may look up information on programming languages on a regular basis whereas a house-wife would generally like to look for the latest in the fashion industry. Getting useful results for such areas of interest, however, is independent of such a definite interest. The internet being the vast resource it is makes it difficult to obtain a good range of results for a given topic. No mechanism currently exists which can capture this general area of interest of the user. If there were a mechanism to do so, the interests inferred could add to the effectiveness of the results displayed for queries. One approach to design such a mechanism would be to suggest links in the topics of interests inferred even when the user is not actively engaged in a searching activity while browsing the internet. The aim of our project is to thus make use of this property of localization of browsing content based on of the user's interests and provide suggestions to make browsing a meaningful experience on the World Wide Web.

1.3. Motivation for the Project

The motivation for such a project stems from our experience in the development of some prediction systems in our Minor Project titled *Transport4You*. The exposure to some Machine Learning techniques made us realize it's applicability in various other domains. Web-browsing being an everyday activity we perform, extrapolating concepts applied there suggested at improving some of the existing functionality.

1.4. Aim of the Project

The project shall lead to the development of algorithms and finally, a software which shall learn the general interest of the browsers and based on this information gathered, shall suggest related links to provide a meaningful user experience.

1.5. Organization of the Report

This report is organized as follows:

Section 2 lists the key aspects identified in the requirements' analysis of the project.

Section 3 contains the Software Requirements Specification of the project. This section is useful for designers who wish to extend on the existing design.

Section 4 contains the Software Design Document of the project. This section is useful for teams implementing the existing design.

Section 5 describes the interface design.

Section 6 contains our Project Plan. This lists out our schedule in the execution of the Project.

Section 7 contains our team's Management Plan.

Section 8 describes the Development Cycle followed in the implementation of our Project.

Section 9 lists out the implementation aspects of the project.

Section 10 discusses our test plan for the system developed.

Section 11 discusses some limitations and future scope of the work done.

Section 12 lists out the material referenced in our project.

2. Requirements Analysis

2.1. Initial study of functional requirements

Our initial study of the functional requirements included the identification of the following functionalities-

i. **Basic requirements which need to be fulfilled by the system**
   a. The system must be installable on an individual machine.
   b. It should be able to learn the set of interests a user possesses. Such interests are gathered from the queries entered by the user in a browser's search bars.
   c. Once learnt, the system must display links and information related to interests of the user. This should be displayed on the browser as a continuous stream.
   d. The suggested links and articles displayed should be fairly random and should keep changing. This is required in order to increase the chances of a user liking and using a particular suggestion displayed.
   e. There should exist a non-intrusive feedback system, wherein a user may reflect on the information learnt by the system.

ii. **External Events**
   a. The user enters a query in a search-bar when looking for some information on the internet. This may be followed by the results of the query being visited.
   b. The user optionally provides a feedback on the interests inferred by the system.

iii. **Temporal Events:**
   a. A user may have a particular field of interest for a given amount of time. The interests inferred should reflect these changing tastes of the user.
   b. The feedback provided is done keeping no particular time schedule in mind.

2.2. Feasibility Study

Based on the various key functional requirements identified in the previous sections, the following aspects of the execution of the projects were analyzed in the course of our Feasibility Study.

2.3. Implementation Details

As the group lacked a rich exposure in the various facets of the implementation tasks that would have followed, much of the Requirements Analysis phase was spent in getting a clear idea of how the different requirements analyzed would be implemented. Based on the analysis, the group had a good grasp of the complexity of such sub-systems that would be required to be implemented.

2.3.1. Web Browser Add-ons

Mozilla Firefox and Google Chrome, two popular web-browsers at the time of development of this project, provide very good support for developing add-ons and extensions to it. The intended system could function as an add-on, which would facilitate parsing of queries entered by the user and using the machine's resources to perform computations. Many popular Machine Learning libraries such as Weka [1] and APIs of other ontology tools [2] can be accessed through such an Add-on.

2.3.2. Database Requirement

The add-on would have to store the topics learnt either as a flat file or in a database. The feasibility of a database connection to an add-on has been studied and found to be possible.

2.3.3. Ontology Tools

Several tools to determine the semantics of an input text are available. The APIs of such tools maybe used in association with Add-on tools to parse and tag scanned web-pages, based on its content.

2.4. A Mathematical Model to Learn User Interests

The system is capable of learning based on queries input by the user. Efficient Machine Learning algorithms may be implemented based on mathematical (mostly probabilistic) models available.

In our case, the system has with it queries, which are incomplete English sentences. Based on these queries and the subsequent web-pages viewed on entering such a query, the system should infer the general topics to which the query pertains to. Having collected such topics discretely through every search, the system should then be able to infer the general interests of the user. The keywords specifying such an interest should be looked up on the internet and relevant material needs to be displayed to the user.

Based on an initial study of the modeling techniques available, we have narrowed down on the a technique called *Probabilistic Latent Semantic Analysis* to deal with our problem of predicting interests based on queries provided. However, we have found multiple tools [2] on the internet (along with their APIs) which are able to tag a given text-input. Such tools may directly be used in our project.

2.5. Ranking of Inferred Topics of Interest

The topics inferred need to be ranked in order to prioritize the interests of the user. We have identified the various parameters that shall contribute to such a ranking formula -

1. Frequency (word count)

2. Time of page opened

3. Through-links of search results

4. User feedback

2.6. Technologies to be used

i. **Database :** SQLite

ii. **Web Based Technology to Develop Add-Ons:** JSP/Python

iii. **Data Formats:** JSON/XML

iv. **Ontology Tools:** TagFinder, TextWise [2]

3. Software Requirements Specification

3.1. Introduction

3.1.1. Purpose

Web-browsing is an integral part of every person's daily activities these days. A property of web-browsing is that the content viewed via a web-browser is local to the user's (person browsing) tastes and preferences. Such an interest may be a result of an activity the user is generally involved in. For example, a computer engineering student may look up information on programming languages on a regular basis whereas a house-wife would generally like to look for the latest in the fashion industry. Getting useful results for such areas of interest, however, is independent of such a definite interest. The internet being the vast resource it is makes it difficult to obtain a good range of results for a given topic. No mechanism currently exists which can capture this general area of interest of the user. If there were a mechanism to do so, the interests inferred could add to the effectiveness of the results displayed for queries. One approach to design such a mechanism would be to suggest links in the topics of interests inferred even when the user is not actively engaged in a searching activity while browsing the internet. The aim of our project is to thus make use of this property of localization of browsing content based on the user's interests and provide suggestions to make browsing a meaningful experience on the World Wide Web.

The motivation for such a project stems from our experience in the development of some prediction systems in our Minor Project titled *Transport4You*. The exposure to some Machine Learning techniques made us realize it's applicability in various other domains. Web-browsing being an everyday activity we perform, extrapolating concepts applied there suggested at improving some of the existing functionality.

3.1.2. Scope

Suggestions4You aims to understand the semantic meaning behind a person's web-browsing activities. In web-browsers used today, there is a need to inherently learn and understand the interests of a user so as to provide a richer browsing experience. The project aims to fulfill this by using the following components –

- Browser Add-Ons to act as a middleware between a user and his/her web-browser.
- APIs for semantic analysis of web content
- Ranking Algorithms to organize the learned information of user tastes.

The project shall lead to the development of algorithms and finally, a software which shall learn the general interest of the browsers and based on this information gathered, shall suggest related links to provide a meaningful user experience.

3.1.3. Definitions, Acronyms and Abbreviations

- *'the system'* unless specified otherwise, refers to the Suggestions4You system, in general. Any sub-system of Suggestions4You is specified as a *system* in a given context.
- *User* is used to describe a person who browses the internet on his/her personal computer.
- *Interest* refers to the various interests of a User, which act as the basis for suggestion of links by the system.
- *Browser* in the current context refers to a Mozilla Firefox web-browser. It specifically refers to this product as the system being developed is currently supported on this platform. Cross-platform independency is currently not aimed for. However, a *browser* in a particular context may also refer to User. (Analogous to *computer :one who computes*)
- The text consistently uses *she* and *her* as the pronoun for a *User*. This is to generalize the text and does not imply at any constraint on the gender of the person browsing.

3.1.4. References

The IEEE 830-1998 standard [4] of SRS documents was used as an outline for this document and was consulted continuously throughout each section and subsection.

3.1.5. Overview

This document is organized as follows:

Sub-section 2 covers an overall description of the various functionalities provided by the system. This section shall be useful for potential users/customers in order to understand the working of the system.

Sub-section 3 covers the requirement specifications of the system. This section shall come in handy to potential developers of the system. This section documents the specifications based on Use Cases imagined and on the basis of the various stimuli to the system (considering it to be event driven). The specifications include the functional requirements, non-functional requirements and other miscellaneous requirements of the system.

Sub-Section 4 discusses requirements concerning the environment of the execution of the project. It also lists out other non-functional requirements, implicit to the given problem statement.

3.2. Overall Description

A User begins a browsing session by starting the web-browser. In order to obtain articles on a topic she has in mind, the User searches the web through a search engine. On repeatedly looking for articles on a topic, the system, which is installed on the User's machine, is able to learn this interest the User currently possesses. On having inferred such an interest, the system looks for more articles on the web corresponding to such interests and keeps displaying obtained results on the browser. The results are displayed presumably in a non-obtrusive way so as to enable the User to carry on her normal functions on the web-browser and use these 'extra' results only if it interests her.

In the case of a User possessing multiple interests and trying to look for topics related to each such interest (which is what happens practically), the system identifies each such interest and based on the relevance and 'strength' of the interest of the User, suggests links on the topics. As an example, if a User possesses an interest in both, computer algorithms and chess, the links displayed shall pertain to both these topics. However, if the interest in chess is only minor and is restrained to checking scores on match days, the suggestions provided shall focus more on computer algorithms than on results on chess.

The system shall also have a way to obtain a feedback from the User, wherein it calibrates its calculations and inferences and determines if it agrees to the actual interests of the User. Based on such a feedback, the system may correct any wrong inferences made.

3.2.1. Product Perspective

The current system being developed is web-browser dependent. This is so because the development APIs for web-browsers is not supported for all available browsers. Also, there exists no common cross-platform API solution for the browsers. Mozilla Firefox, having the best developer community for Add-on development, is the natural choice for the deployment of such a system.

The core-functionalities of the proposed system do exist as individual components of logic in various software systems throughout different application domains. *StumbleUpon* [6], for example, is a discovery engine (a form of web search engine) that finds and recommends web content to its users. Its features allow users to discover and rate Web pages, photos, and videos that are personalized to their tastes and interests using peer-sourcing and social-networking principles. Our system differs from a discovery engine such as StumbleUpon in the following aspects –

- No explicit tagging of pages is required to let the system know of the User's interest.
- The repository of searches is contained to the machine on which the browsing activity is performed. Hence, the results and inferences are quick and contained to the tastes of the Users.

The major components interacting with the system can be understood from Figure 3.1. Software Ideas Modeler [7] was used for the design of this figure.

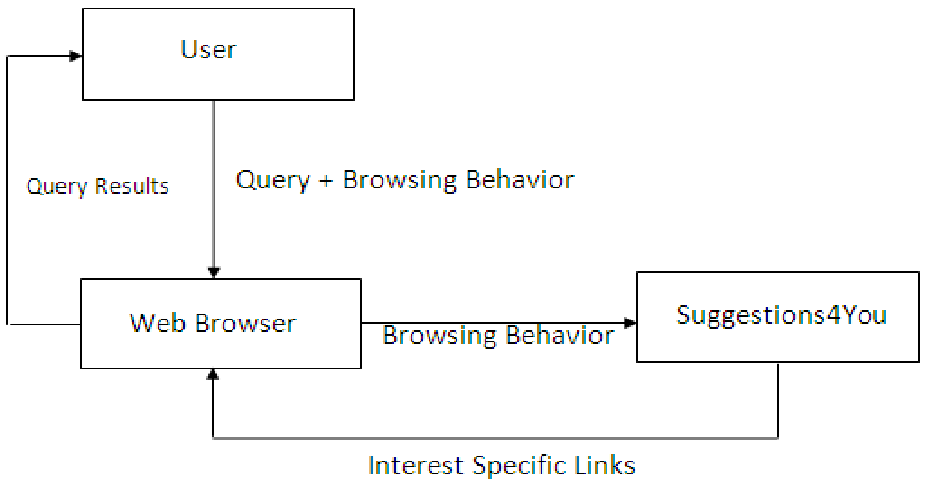


Figure 3.1- Context diagram for the Suggestions4You System

Referring to Figure 3.1, we see that the external users of the system are

- Users
- Web Browser

3.2.1.1. User Interfaces

The interface between a User and the system relies on certain key assumptions described in the Assumptions and Dependencies sub-section. It is assumed that every user has a web-browser and has enabled Javascript on it.

3.2.1.2. Hardware Interfaces

No hardware interfaces exist.

3.2.1.3. Software Interfaces

The system relies on the following software for its functionality –

i. *Name*: Mozilla Firefox
ii. *Mnemonic*: Firefox
iii. *Version*: Minimum 3.6.1

This software is used to support the functionality of the entire system. An appropriate scripting technology, which shall be the core technology to run the algorithms, shall be run through this platform.

3.2.1.4. Communications Interfaces

No communication interfaces exist.

3.2.1.5. Memory Constraints

We assume that the minimal physical memory is utilized to operate our system. No specific constraints, however, exists.

3.2.1.6. Site Adaptation Requirements

No explicit site information is mentionable.

3.2.2. User Characteristics

No special qualification concerning a User's technical expertise, experience or educational level is mentionable.

3.2.3. Constraints

i. The performance of the browser should not be hindered because of the existence of the system.
ii. Minimum bandwidth utilization should be made by the system.
iii. The links suggested should be 'fresh' in the sense that a new set of links to a given topic needs to be generated as often as possible.

3.2.4. Assumptions and Dependencies

i. The web-browser used is Mozilla Firefox.
ii. It's assumed that the majority of usage is by one user only. This localizes the interest of the user.
iii. There needs to exist a tangible, recognizable set of interests the user possesses in order to produce best results.
iv. The User enables scripting facilities of the browser at all times.

3.3. Specific Requirements

3.3.1. External Interfaces

3.3.1.1. Display Suggested Links

Name of Item: ***Display Suggested Links***

Description of Purpose:

This interface will suggest interest specific links to the user in a non intrusive behavior.

Source of Input or Destination of Output:

Based upon the interest learnt by the Suggestion engine, the engine downloads Suggestion links from the internet.

The suggestion links are then displayed onto the pull down menu.

Timing:

The links are displayed to the user whenever the user clicks the *Suggestions* option provided on top of the web page.

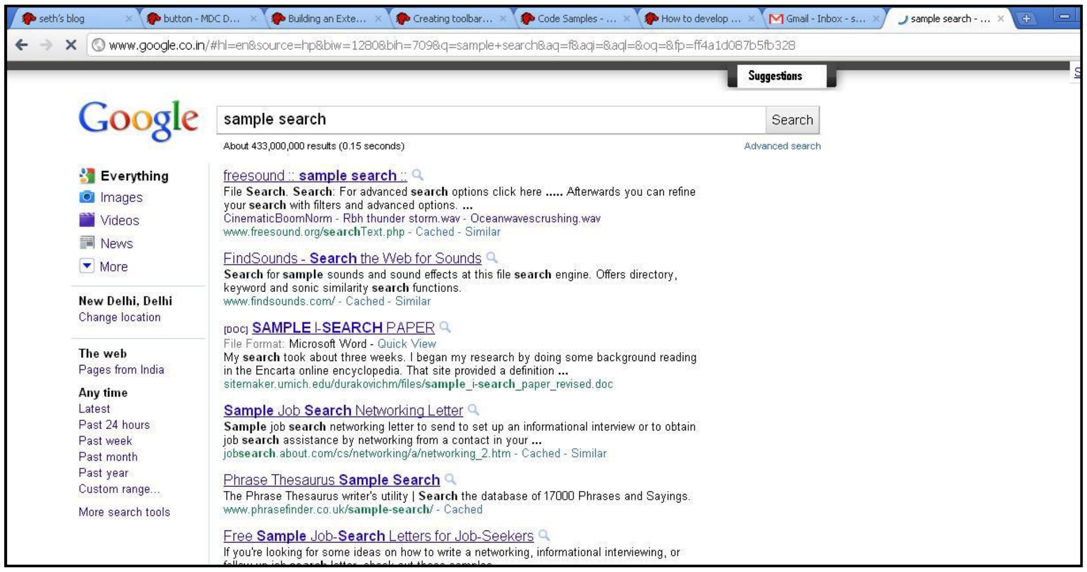
The links provided on the Suggestion's pull down menu should refreshed once the user visits the suggested links or after some fixed interval of time.

Relationships to Other Inputs/Outputs – No relationship

Data Formats – The information about the user interests is available from the database store, which will be in varchar format.

Command Formats – No explicit commands. GUI available for input.

Screen Formats/Organization



www.google.co.in/#hl=en&source=hp&biw=1280&bih=709&q=sample+search&aq=f&aqi=&aql=&oq=&fp=ff4a1d087b5fb328

sample search

About 433,000,000 results (0.15 seconds)

freesound :: **sample search** ::
File **Search**. **Search**: For advanced **search** options click here ..... Afterwards you can refine your **search** with filters and advanced options. ...
CinematicBoomNorm - Rbh thunder storm.wav - Oceanwavescrushing.wav
www.freesound.org/**search**Text.php - Cached - Similar

FindSounds - **Search** the Web for Sounds
Search for **sample** sounds and sound effects at this file **search** engine. Offers directory, keyword and sonic similarity **search** functions.
www.findsounds.com/ - Cached - Similar

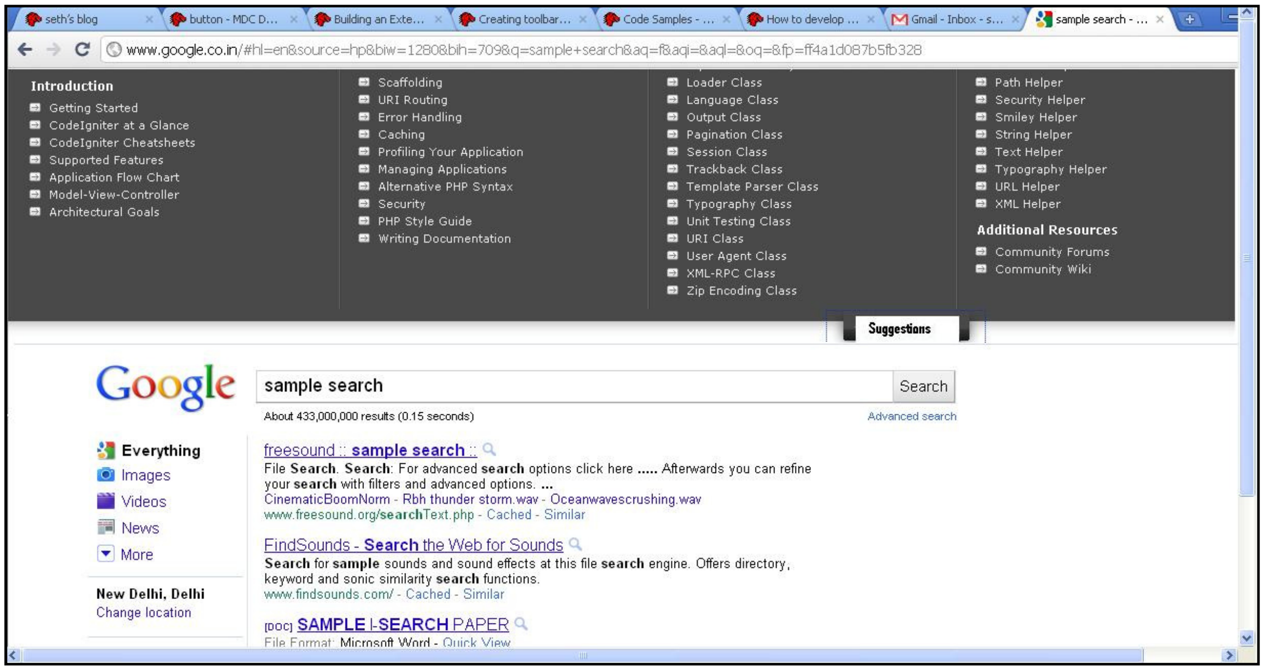
[DOC] **SAMPLE** I-**SEARCH** PAPER
File Format: Microsoft Word - Quick View
My **search** took about three weeks. I began my research by doing some background reading in the Encarta online encyclopedia. That site provided a definition ...
sitemaker.umich.edu/durakovichm/files/**sample**_i-**search**_paper_revised.doc

Sample Job **Search** Networking Letter
Sample job **search** networking letter to send to set up an informational interview or to obtain job **search** assistance by networking from a contact in your ...
job**search**.about.com/cs/networking/a/networking_2.htm - Cached - Similar

Phrase Thesaurus **Sample Search**
The Phrase Thesaurus writer's utility | **Search** the database of 17000 Phrases and Sayings.
www.phrasefinder.co.uk/**sample-search**/ - Cached

Free **Sample** Job-**Search** Letters for Job-Seekers
If you're looking for some ideas on how to write a networking, informational interviewing, or

Figure 3.2– Pull-Down Suggestions Menu



About 433,000,000 results (0.15 seconds)

freesound :: sample search ::
File Search. Search: For advanced search options click here ..... Afterwards you can refine your search with filters and advanced options. ...
CinematicBoomNorm - Rbh thunder storm.wav - Oceanwavescrushing.wav
www.freesound.org/searchText.php - Cached - Similar

FindSounds - Search the Web for Sounds
Search for sample sounds and sound effects at this file search engine. Offers directory, keyword and sonic similarity search functions.
www.findsounds.com/ - Cached - Similar

[DOC] SAMPLE I-SEARCH PAPER
File Format: Microsoft Word - Quick View

Figure 3.3–Proposed Look and Feel for the Suggestions

End Messages – No end messages, just the menu of the suggested links to be displayed.

3.3.2. Functions

The functions of the system identified here are on the basis of the various stages of the system mainly, capturing the behavior of the user from its browsing activities, displaying the interest specific links and finally getting the feedback from the user.

Some of the key functions of the system identified are –

3.3.2.1. Behavior Capture

The user browses the internet

Validity checks on the inputs

Since there shall be no direct input from the user, the engine shall capture the behavior from the user activities like,

- search queries being entered.
- sequence of links followed.
- time spent on a particular page.

These events shall form the input to this function.

Exact sequence of operations

- User opens his web browser to surf the web.
- User may enter a search query or URL.

- User may navigate from one web page to other web page by clicking on the links on the web page.
- User may enter a new query or URL.
- User may open a new tab and starts a new session of web browsing.
- User spends time reading the content of the web page.
- The user closes the web browser.

Responses to abnormal situations

- Browser may crash mid-way: The database of inferred interests should remain consistent and may omit the most recent session's inference.
- One of the property files of Add-on is deleted – There is no response which can get the system back to normalcy. All information shall be lost in that case. An appropriate message to uninstall the existing system should be provided.

Effect of parameters

Interest-driven browsing shall assist in its identification by the system. This may update the already learned interests of the system or shall add a new interest to the system.

Input /Output sequences

The input is User generated browsing.

There is no visible output. The calculations performed are reflected in the following function.

3.3.2.2. Display Suggested Links

Validity checks on the inputs

No inputs are expected from the User.

Exact sequence of operations

- Links on the inferred interests are shown in the drop-down menu (as described in Figure 3.2,3.3)
- The content in this menu is dynamically updated based on the interests learned.
- Clicking on any of the displayed links will redirect to the intended page of interest.

Responses to abnormal situations

- Internet connection is disabled – The last shown links should be displayed.
- The scripting features of the web browser are switched off – A message notifying the User to switch on such features should be displayed.

Effect of parameters

The click on a link opens a new tab/window which displays the suggested link.

Input /Output sequences

Input involves a User clicking on a suggested link.

The output is the suggested link being displayed on the browser.

3.3.2.3. Feedback

This function needs to ascertain the topics of interest inferred by the system. The nature of input and output are left to the designers to choose. However, it must be noted that the system developed should be as unobtrusive as possible to the User's normal browsing activities.

Effect of parameters

The input should be able to calibrate the interests learned so far and correct the status of any learned interests if found incorrect. The status of an interest refers to the relative importance that interest has among all inferred interests. Please refer to *Section 3.2, Overall Description*, for a description on multiple interest inference.

3.4. Non-Functional Requirements

3.4.1. Unobtrusive Presence

The presence of the proposed system on the web browser should not hinder the normal activities of a User. Minimal feedback from the system should be initiated. Also, the performance of the web browser should not degrade with the new add-on installed.

3.4.2. Ease of Use

The display of suggestions should be easily accessible.

3.4.3. Quality of Links

The links on a given topic of interest need to be refreshed over a period of time. The quality of the links should be high in relation to the relevance of the content and the corresponding interest.

3.4.4. Operating Environment Requirements

3.4.4.1. Hardware

No special hardware is needed to support the system. A personal computer with a modest configuration should do.

3.4.4.2. Software

A Linux/Windows/Mac operating environment should support the system. The system may also need to be able to access a database system to store and perform intermediate computations.

4. Software Design Document

4.1. Introduction

4.1.1. Purpose

Web-browsing is an integral part of every person's daily activities these days. A property of web-browsing is that the content viewed via a web-browser is local to the user's (person browsing) tastes and preferences. Such an interest may be a result of an activity the user is generally involved in. For example, a computer engineering student may look up information on programming languages on a regular basis whereas a house-wife would generally like to look for the latest in the fashion industry. Getting useful results for such areas of interest, however, is independent of such a definite interest. The internet being the vast resource it is makes it difficult to obtain a good range of results for a given topic. No mechanism currently exists which can capture this general area of interest of the user. If there were a mechanism to do so, the interests inferred could add to the effectiveness of the results displayed for queries. One approach to design such a mechanism would be to suggest links in the topics of interests inferred even when the user is not actively engaged in a searching activity while browsing the internet. The aim of our project is to thus make use of this property of localization of browsing content based on the user's interests and provide suggestions to make browsing a meaningful experience on the World Wide Web.

The motivation for such a project stems from our experience in the development of some prediction systems in our Minor Project titled *Transport4You*. The exposure to some Machine Learning techniques made us realize it's applicability in various other domains. Web-browsing being an everyday activity we perform, extrapolating concepts applied there suggested at improving some of the existing functionality.

4.1.2. Scope

Suggestions4You aims to understand the semantic meaning behind a person's web-browsing activities. In web-browsers used today, there is a need to inherently learn and understand the interests of a user so as to provide a richer browsing experience. The project aims to fulfill this by using the following components –

- Browser Add-Ons to act as a middleware between a user and his/her web-browser.
- APIs for semantic analysis of web content
- Ranking Algorithms to organize the learned information of user tastes.

The project shall lead to the development of algorithms and finally, a software which shall learn the general interest of the browsers and based on this information gathered, shall suggest related links to provide a meaningful user experience.

4.1.3. Definitions, Acronyms and Abbreviations

- *'the system'* unless specified otherwise, refers to the Suggestions4You system, in general. Any sub-system of Suggestions4You is specified as a *system* in a given context.
- *User* is used to describe a person who browses the internet on his/her personal computer.
- *Interest* refers to the various interests of a User, which act as the basis for suggestion of links by the system.
- *Browser* in the current context refers to a Mozilla Firefox web-browser. It specifically refers to this product as the system being developed is currently supported on this platform. Cross-platform independency is currently not aimed for. However, a *browser* in a particular context may also refer to User. (Analogous to *computer :one who computes*)
- The text consistently uses *she* and *her* as the pronoun for a *User*. This is to generalize the text and does not imply at any constraint on the gender of the person browsing.

4.1.4. References

The IEEE 830-1998 standard [5] of SRS documents was used as an outline for this document and was consulted continuously throughout each section and subsection.

4.1.5. Overview

This document is organized as follows:

Sub-section 2 covers an overall description of the various functionalities provided by the system. This section shall be useful for potential users/customers in order to understand the working of the system.

Sub-section 3 covers the requirement specifications of the system. This section shall come in handy to potential developers of the system. This section documents the specifications based on Use Cases imagined and on the basis of the various stimuli to the system (considering it to be event driven). The specifications include the functional requirements, non-functional requirements and other miscellaneous requirements of the system.

Sub-Section 4 discusses requirements concerning the environment of the execution of the project. It also lists out other non-functional requirements, implicit to the given problem statement.

4.2. Architecture Overview

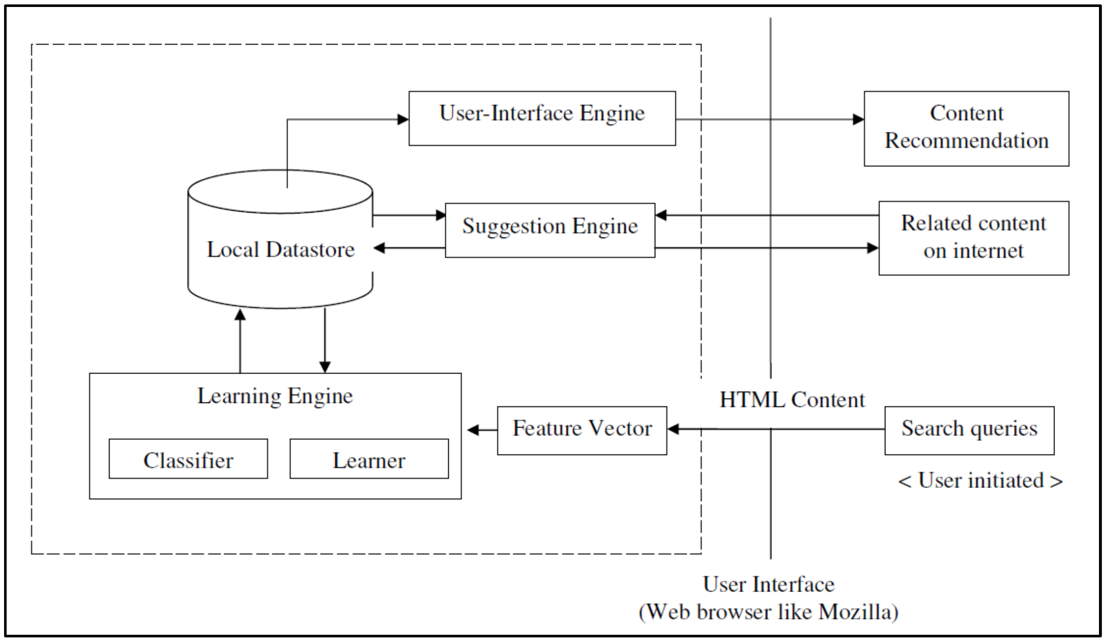


Figure 4.1 - System Architecture

Suggestions4You will be an add-on which acts as a middleware between the user (effectively, the user's web browser) and the web.

As shown in Fig. 1, *Suggestions4You* shall comprise the following components:

- **Semantic Analyser**
  The Semantic Analyser will analyse the user's browsing content. It will then generate tags and create a feature vector for user interests. For the purpose of the project, pre-existing APIs will be used for the semantic analysis of web content and generation of tags.
- **Learning Engine**
  The Learning Engine will use the feature vector created by the Semantic Analyser and certain other parameters and then use ranking algorithms on them to rank user interests. These results will then be stored on the local data-store of the browser.
- **Local Datastore**
  Suggestions to a user will be based on their local browsing. The Local Datastore of the browser will contain all the ranking data based on user history.
- **Suggestion Engine**

The Suggestion Engine will create suggestions based on ranking data and related content from the web in order to be shown to the user.

- **User-Interface Engine**
  The User-Interface Engine will display the final suggestions to the user in an appropriate format. This will be responsible for the actual data that gets displayed to the user.

4.3. Design Overview

4.3.1. Use-Cases

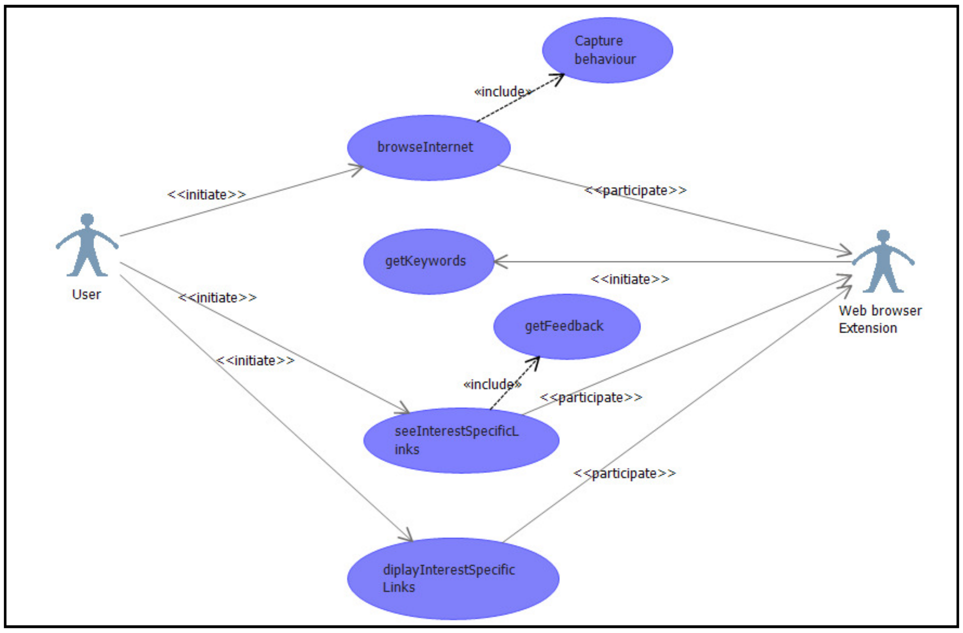


Figure 4.2 - Use-Case depicting high level design of system

The primary actors identified in our system are –

- User
- Web-Browser Extension

A User is one who would be surfing the web for information. She may perform the following actions when doing so –

- Enter a query in a search/address bar
- Visit results of the query provided
- Visit links suggested by our system

The Web-browser Extension is essentially the entire *Suggestions4You* system. Every query entered by the User goes through this extension. On entering a query, the system parses the results of the query and classifies them into semantic-rich signatures. The signatures describe the

topics these results belong to. The system collects every analyzed signature and then ranks them according to relevance to the User. These ranked signatures are then made use of to search the web independently in order to provide additional links in the recognized topics of interest. The suggestions gathered are then conveniently displayed on the web-browser.
As a mechanism to ensure the relevance of the suggestions provided by the system, a User visiting the suggested links is used as feedback to improve the quality of future suggestions.

The flow of events among the various use-cases described in Figure 4.2 has been described below.

- **browseInternet:**

Use case name	***browseInternet***
Participating Actors	Initiated by *User* Communicates with *web browser extension* *Includes captureBehaviour.*
Flow of events	* User enters query or URL in the address bar. * The browser performs the respective action and generated the results. * The extension calls captureBehavious to capture the users' behaviour. * The user enters a new query in the search bar or a address bar, the extension saves the data for the current session.
Entry condition	The user enters a Query or URL in the web browser.
Exit condition	The user manually enters a new query in the address bar.

- **captureBehaviour:**

Use case name	***captureBehaviour***
Flow of events	* The extension captures the tags or the keywords from the queries entered or content visited.
Entry condition	The user enters a Query or URL in the web browser.
Exit condition	The user manually enters a new query in the address bar.

- **seeInterestSpecificLinks:**

Use case name	***seeInterestSpecificLinks***
Participating Actors	Initiated by the user. Communicates with the web browser extension.
Flow of events	* The user clicks an interest specific links from the ticker. * For each user action getFeedback is called to capture the user feedback. * The link is replaced by new valid link.
Entry condition	The user enters a link on the ticker.
Exit condition	The user closes the browser.

- **displayInterestSpecificLinks:**

Use case name	***displayInterestSpecificLinks***
Participating Actors	Initiated by the user. Communicates with the web browser extension.
Flow of events	* The browser starts his browsing session. * The web browser extension calculates the interest rating from the data saved from the previous browsing sessions. * On the basis of this ranking, the extension displays the user specific links to the user.
Entry condition	The user enters a Query or URL in the web browser.
Exit condition	The user manually enters a new query in the address bar.

- **getKeyword:**

Use case name	***getKeyword***
Participating Actors	Initiated by the web browser extension.
Flow of events	* The user enters a query in the web browser and visits a page on the internet. * The content of the visited web page is tagged to get the tags describing the text.
Entry condition	The user enters the query or visits the web pages.
Exit condition	The user closed the browser.

4.4. Data Flow Diagram

The Level-1 DFD of our system is depicted in Fig. 3. It shows how data interacts with the User, who is the central actor, initiating the processes in the system.

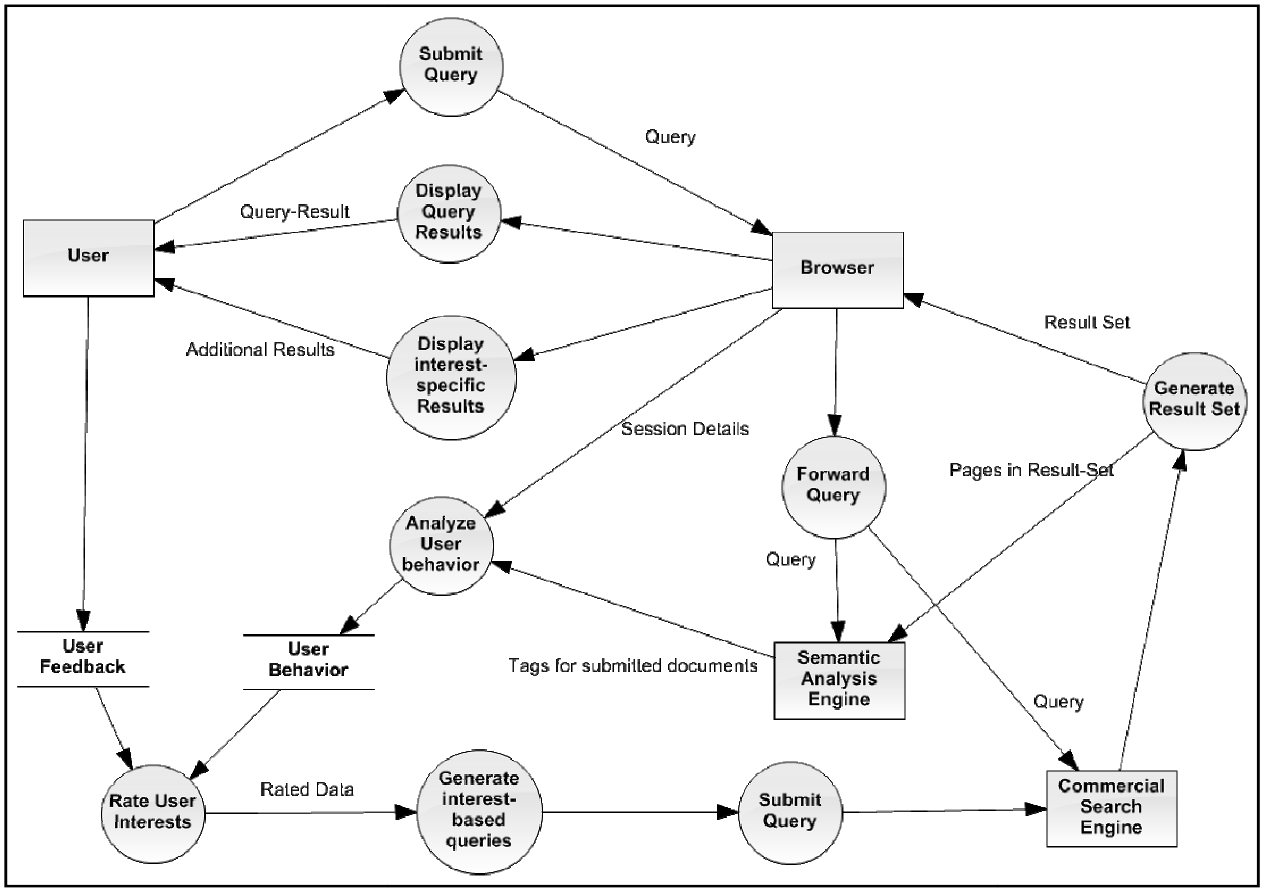


Figure 4.3 - Level 1 Data Flow Diagram

The processes initiated by a User are

- *Submit Query*- pertains to the User browsing the web. Once a User submits a query, the browser forwards it to a commercial search engine. Along with the query entered, the results of the query are then forwarded to a Semantic Analysis Engine which classifies the content of the result-set into rich semantic signatures. These signatures are then stored in a local data-store. This collective information is then used to rank the areas of interests of the User. Based on the interests analyzed, queries are generated to search the web for additional links related to it. These links are then displayed to the User on the web-browser.

- *User Feedback* –pertains to the User validating the results inferred by the system. As a mechanism to ensure the relevance of the suggestions provided by the system, a User visiting the suggested links is used as feedback to improve the quality of future suggestions.

4.5. Decomposition Description

The System Architecture described in Section 4.2 can be further decomposed into subsystems as discussed in the following sub-sections –

4.5.1. Learning Engine

The Learning Engine is responsible for analyzing the semantic properties of the page content opened by the User. Semantic analysis here means understanding the topics of the page content. Every webpage that's opened would pass through this engine, resulting in the semantic interpretation of the topics the page pertains to. In order to achieve this, a learning engine typically shall comprise the following components –

- Classifier – A classifier is that component which makes a feature vector of the words on the page and applies algorithms like Probabilistic Latent Semantic Analysis [3] in order to obtain a set of word-tags which describe the topics the pages describe.
- Learner – Once the classifier creates the tags to pages, the Learner is responsible for making use of ranking algorithms to grade the topics by their importance and relevance to the User using the system.

Classification engines are commercially available for public usage. Web services like textwise.com [2] provide for semantic tags given a body of text. The following sections discuss the learner component of the Learning Engine.

4.5.2. Motivation

Every page that is browsed by a User can be characterized by the following features –

4.5.2.1. Semantic Words

These words describe the topics of the content of the given page. For example, the words describing a page on the *Shortest Path Algorithm* can include the set $\{Computer\ Science, Graph\ Theory, Edsger\ Dijsktra\}$. These words shall be obtained from a third party commercial Semantic Engine (Refer to System Architecture), which would return these set of words along with the weights describing the relative relevance of each of the returned words with the page. Hence, the returned set would be of the following format –

$\{Computer\ Science$: 0.89

$Graph\ Theory$: 0.79

$Edsger\ Djikstra$: 0.7 }

The weights described here are normalized on a scale of 0 to 1.

4.5.2.2. Time of Page Opened

This metric refers to the amount of time a particular page being viewed is kept open. The metric has been chosen on the presumption that the longer a User goes through an opened page, the better are the chances that the content of the page has interested her. Also, a short interval of page opening may be interpreted as a User going through the content of the page and possibly discarding the page. Moreover, the metric also needs to support noisy input possibilities. A page opened for ten-odd minutes could be interpreted in two ways - the User being very much interested in the content of the page and as a result, reads it for the full ten minutes or the User reads the page for a while, leaves it open and gets involved in some other unrelated activity.

Considering such noisy inputs, the metric's impact to the overall inference of the User's interests needs to be modeled considering the following scenarios –

The metric's impact should be maximum for a time interval of 1 to 6 minutes, wherein, the increase of each minute (starting from the first minute) should weigh non-linearly to the overall inference. For e.g. The interest of a person who has opened a page for 4 minutes should be weighed much more than if she has opened the page for 3 minutes. The increase in every minute of page viewing confirms an interest in the topics pertaining to that page.

In order to account for the noisy input when pages are opened for too long, the metric's impact needs to taper off after a period of 15 minutes (say) as we would not want to wrongly heavily weigh a page kept unintentionally open for a long time.

4.5.3. Graph based Page-Visitation Metric

Let $Words(i,j)$ represent the semantic words which represent the topics associated with the page that was opened. Here, $Words(i,j)$ would represent the j^{th} word described as a topic of the i^{th}page opened, assuming a vector of tagged words $T_i: \{w_1, w_2, w_3 \dots w_j\}$ describe the topics for every given page.

Let a directed graph $G(V,E)$ represent a graph which models the page visitation behavior of the User. For every page i visited by the User, a node $v_i \in V$ is created in G. The node is characterized by the tag vector T_i. A directed edge $v_a \rightarrow v_b$between any two nodes v_a and $v_b$is created if page b is opened on clicking a link on page a. We shall refer to such an action as *spawning*, where a new page is *spawned* by an already opened *parent page*.

As an illustration, let graph G_1denote the User's page browsing in an arbitrary session-

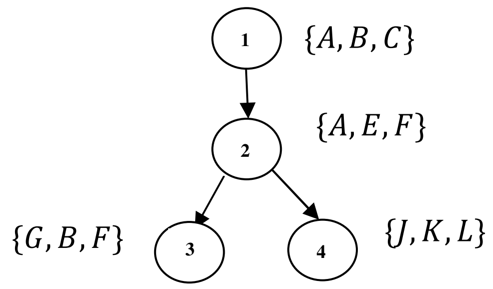


Figure 4.4– Page Browsing Graph – Session 1

Here, the vectors $\{A, B, C\}$ etc. denote the tag-vectors T_i for nodes 1,2,3,4 respectively. Also, node 2 is spawned by 1 and nodes 3, 4 are spawned by 2. It may so happen that the spawned pages are similar in topics to the parent page. In the above graph, $\{A, B, C\}$ may represent $\{Computer\ Science, Graph\ Theory, Edsger\ Dijsktra\}$ whereas $\{A, E, F\}$ may represent $\{Computer\ Science, Economic\ Game\ Theory, Von\ Neuman\}$. We may thus conclude that the general interest of the User lies in the broad domain of Computer Science and branches of into specifics such as Graph Theory and Game Theory.

Similarly, let G_2denote the User's page browsing in another session-

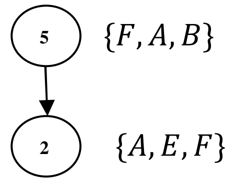


Figure 4.5 – Page Browsing Graph – Session 2

We notice that there are two nodes, namely 2 and 3 which are common to both browsing sessions. From the algorithm, the cumulative graph at the end of both the browsing sessions would be –

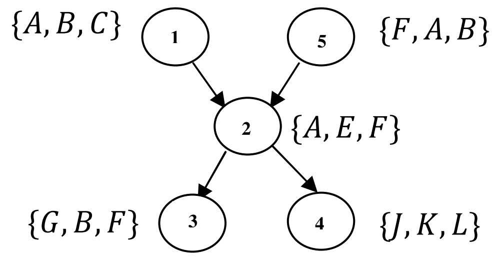


Figure 4.6 – Cumulative Page Browsing

The metrics discussed in the Motivation Section can be applied on the above graph as follows –

Hidden Interest Metric

The cumulative page browsing graph of the browsed sessions reveals seemingly unexplored relations between browsed pages. In Session 2, the User spawned Page 2 through Page 5 and stopped there. However, in a previous session, the User had browsed on from Page 2 to Page 3. We may thus extrapolate that in an arbitrary session, the User may have ended up at Page 3 having started from Page 5. Hence, the 5→2→3 connection is the newly discovered browsing pattern of the User.

Increased Interest Metric

In session 1, there exists a common keyword A between Pages 1 and 2. Since Page 2 is a child spawned immediately by Page 1, we may apply the metric of increased interest by adding the weight of word A relative to Page 1 (as calculated by the Semantic Engine) to an exaggerated weight of word A relative to Page 2. The exaggeration may be presented by multiplying the weight by a constant factor such as the depth of the page relative to the parent page. In this case, the depth of Page 2 relative to Page 1 is 2.

Hence, the overall weight of word A would be –

$$Weight_{A,\ Overall} = Weight_{A,\ Page\,1} + (2 \times Weight_{A,\ Page\,2})$$

However, in spite there being a word common between Page 1 and Page 3 (namely B), the multiplying factor of the depth (in this case 3) would not be incorporated as Page 3 is not immediately spawned by Page 1. In the calculation of weights at Page 3, the metric shall be made use of to calculate the weight of word F in the following manner –

$$Weight_{F,\ Overall} = Weight_{F,\ Page\,5} + \left(2 \times Weight_{F,\ Page\,2}\right) + \ (3 \times Weight_{F,\ Page\,3})$$

Time of Browsing Metric

The implementation framework provides for the calculation of the time spent by a User on an opened browser window. This time, on normalizing to a scale of 1, may then be used to accentuate the weights of the calculated words. The general rule of the metric would be to assign a higher multiplying factor given a high amount of time spent on the page.

4.5.4. Algorithm

This section describes an approach to measure the importance and relevance of tagged words of opened web pages and to make use of the abovementioned metrics in calculating this importance measure.

```
$\boldsymbol{ALGORITHM : Calculate\ Relevance\ Rank\ of\ Tagged\ Topics\ of\ Browsed\ Webpages}$
1. $cumulativeWeights\ [\,]: \{0\}$
2. $DFS(currentNode, commonKeywords)\{$
3.   $for\ each\ child\ of\ currentNode\{$
4.       $newCommonKeywordSet = commonKeywords(commonKeywordSet, child.keyword);$
5.       $newCommonKeywordSet.updateDepth(\,);$
6.       $for\ each\ keyword\ in\ newCommonKeywordSet\{$
7.           $x = normalize(child.timeSpent(\,));$
8.           $y = normalize\big(child.timeAccessed(\,)\big);$
9.           $cumulative[keyword] \mathrel{+}= x \times y \times keyword.depth \times child.keyword[keyword].weight;$
10.      $\}\ END\ FOR$
11.  $DFS(child, newCommonKeywordSet);$
12. $\}\ END\ FOR$
13. $\}\ END\ FUNCTION$
```

In the algorithm, $cumulativeWeights$ is a vector which stores the final calculated weight of every keyword in the list of topic-tags. It is initially set to 0 and is updated at each depth-level of the page browsing graph. The graph is traversed depth-first, wherein each child is visited and the weights at that child are calculated for its corresponding tag-vector. This continues till the leaf-nodes of the graph are visited. The final results are reflected onto a data-store.

4.6. Data Design

4.6.1. Design Motivation

The following guidelines were kept in view while designing the database for the system-

- The database had to have as few tables as possible, as a large number of tables would hinder the smooth working of the web browser.
- The focus of data storage would be the web browsing history of the user.

Considering these guidelines, one key table was decided upon to contain all key session details in one table. This information is then made use of by querying to analyze the data based on the time the page was opened, whether it was spawned from a page having common tags and other metrics discussed in the previous section.

Once the browsing data has been analyzed and an appropriate ranking achieved, the words are stored in a table which would then be accessed by the presentation logic in order to suggest related links.

Implementation specific information such as the list of suggested links already visited by the User are stored in another table.

4.6.2. Logical Design Elements

The various tables thus identified are-

i. Session_details
ii. Page_keywords
iii. Final_weights
iv. Dimensions
v. History
vi. Search_results

4.6.3. Entity-Relationship Diagram

The Entity-Relationship Diagram for the system can be described as follows -

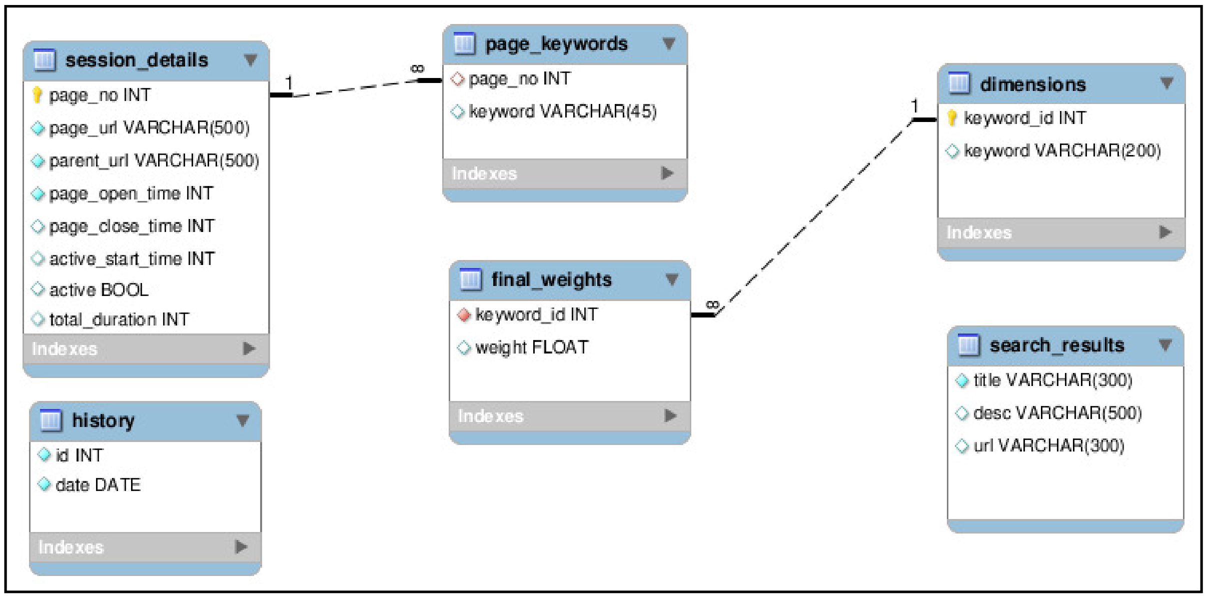


Figure 4.7– Entity Relationship Diagram for the System

Legend for the diagram-

Primary Key	🔑	Not-null attribute	🔹
Foreign Key	♦	Null-allowed attribute	◇
Relationship between tables	_ _ _ _ _		

4.6.4. Data Dictionary

i. Session_details

The master table containing basic session information about the user.

Col no	**Name**	**Description**	**Null?**	**Notes**
1	Page_no	Unique number assigned to every page opened	No	Primary Key, Auto Increment
2	Page_url	URL of the opened page	No	
3	Parent_url	URL of the parent page spawning the opened page	No	
4	Page_open_time	Time the page was opened	No	
5	Page_close_time	Time the page was closed	No	
6	Active_start_time	The timestamp at which the tab opening the current page is returned to	No	
7	Active	Current status of the tab that is opened. It is 1 if the tab is active or 0 if it switched to another tab	No	
8	Total_duration	Reason for banning the user (if applicable)	Yes	

ii. Page_Keywords

This table maintains the tag-vector information for each of the web-pages opened.

Col no	**Name**	**Description**	**Null?**	**Notes**
1	Page_no	Unique number assigned to every page opened	No	Foreign Key
2	Keyword	Tag-vector for the page	No	

iii. Dimensions

This table maintains the list of all possible topic-tags a page can be assigned. This table is used to interpret the words corresponding to the ID sent by the third party classification engine.

Col no	**Name**	**Description**	**Null?**	**Notes**
1	Keyword_id	Unique ID assigned to every word in the list of possible topics.	No	Primary Key
2	Keyword	Corresponding topic	No	

iv. Final Weights

The table is primarily accessed by the classification engine to enter the measure importance of a given topic of web browsing. This is used to prioritize and rank the most relevant topic of interest corresponding to the User.

Col no	**Name**	**Description**	**Null?**	**Notes**
1	Keyword_id	Unique ID assigned to every word in the list of possible topics	No	Foreign Key
2	Weight	Corresponding weight as calculated by the Learning and Ranking Engine. Refer to Section 4.5.1 for details.	No	

v. History

The table is accessed by the presentation engine in order to keep a track of all the suggested links that have been visited by the User. It is accessed to show only the latest of links to the User.

Col no	**Name**	**Description**	**Null?**	**Notes**
1	Id	The ID of the suggested link, as shown in the presentation area.	No	Primary Key
2	Date	The date it was visited on	No	

vi. Search Results

This table is accessed by the Suggestion Engine in order to store intermediate search results obtained by querying the ranked keywords. The highest ranked words are picked from the Final Weights table and passed onto the search APIs.

Col no	Name	Description	Null?	Notes
1	Title	Title of the search result	No	
2	Desc	Summary of the result	No	
3	Url	URL of the result	No	

4.7. Interaction Viewpoints

The User interacts with the system in a manner which does not affect her browsing. A session created by a User is constantly captured and analyzed by the system. The sequence of events maybe further explained from the diagram depicted below. The design of the Sequence Diagrams was done referring to [7]

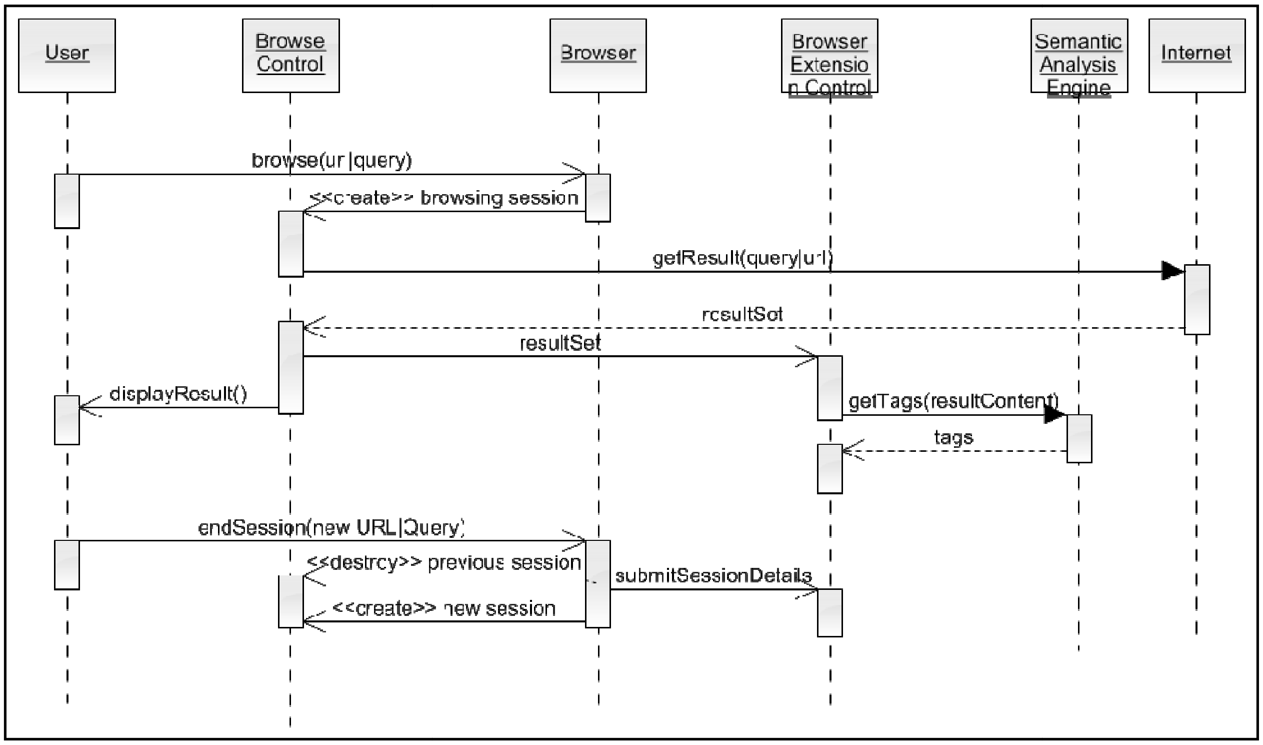


Figure 4.8– Sequence Diagram for the System

On entering a URL, th control is then passed onto the Learning Engine, which forwards the page to a commercial document tagging software. The software returns the topics relevant to the page being viewed, which is then handled by an intermediate script belonging to the system, which would then classify the most relevant topic of interest of the User. This set of most important tags would then be used to query the internet for related links to the topic.

5. Human Interface Design

5.1. Overview of User Interface

The User interface of our system is presented as an Add-On to the Mozilla Firefox web browser. Firefox Add-On development supports component development through XUL [10], an XML format language used to describe palette components like sidebars, buttons and other GUI elements. The UI should be unobtrusive in nature, giving full control to the User to show/hide the suggested links.

Considering the above non-functional requirements, the following design approach was considered.

5.2. Design

5.2.1. Constraints

The following constraints hold for the presentation of the suggested links-

- The presented information should have the option of being visible or not in the main browsing area and such visibility should be controlled by the User.
- The links displayed ought to be "fresh", in the sense, should change over time and must reflect any change in the browsing interests of the User.
- The displayed information should be presented such that any form of feedback to it in the subsequent development of the software should be easily captured and implementable.

5.2.2. Features

The entire presentation logic has been provided through the available components in the Mozilla Firefox Extension development kit. It has been provided as a sidebar the presence of which can be toggled through the activation/de-activation of a button available on the status-bar of the web browser. Some of the salient features that have been provided are as follows –

5.2.2.1. Dynamic Tree Structured Sidebar Display

This facility provides for the dynamic structuring of results displayed in a sidebar on the web browser. The results are shown as a list of titles. On clicking any title, a tab opens consisting of the URL to the link along with the description of the topic. Refer to Screenshot 1 and 2 in the following section.

5.2.2.2. Visibility Control of Sidebar

The sidebar opens on the click of a button, placed next to the address bar of the browser. It toggles shut on clicking it again. Refer to Screenshot 3 in the following section.

5.2.2.3. Options against Each Suggested Link

Each item suggested in the list has the following options to be operated on –

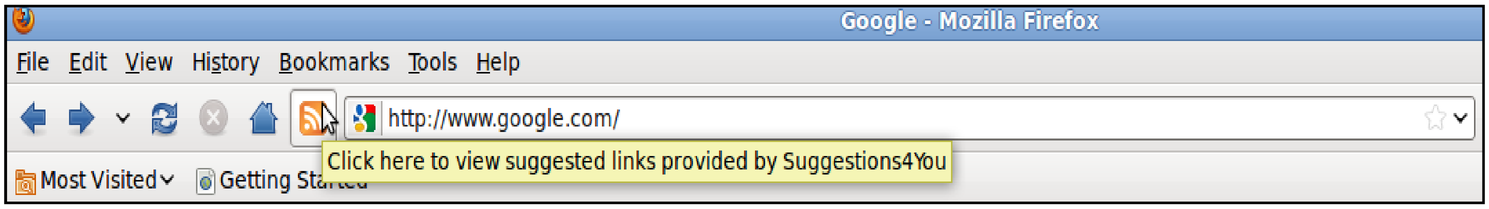
Open, Open in New Tab, Open all in Tabs, Mark as Read, Copy Title, Copy Link and *Options.*

This is displayed as a Menu-list on right-clicking any of the items in the list. The *Options* option provides some configurable parameters for the Users. Refer to Screenshot 4 in the following section.

5.3. Screenshots

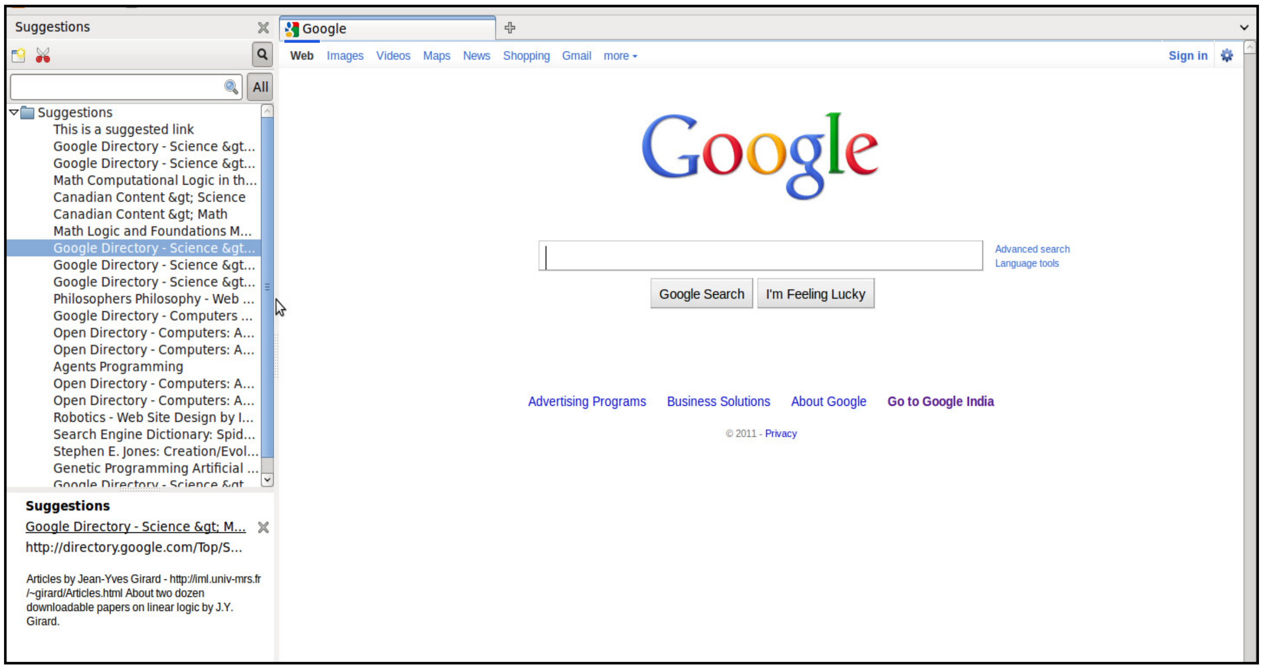
The following are the screenshots of the Suggestions4You system.

5.3.1. Visibility Control Button



Click here to view suggested links provided by Suggestions4You

5.3.2. List of suggested links



Suggestions
- This is a suggested link
- Google Directory - Science &gt...
- Google Directory - Science &gt...
- Math Computational Logic in th...
- Canadian Content &gt; Science
- Canadian Content &gt; Math
- Math Logic and Foundations M...
- Google Directory - Science &gt...
- Google Directory - Science &gt...
- Google Directory - Science &gt...
- Philosophers Philosophy - Web ...
- Google Directory - Computers ...
- Open Directory - Computers: A...
- Open Directory - Computers: A...
- Agents Programming
- Open Directory - Computers: A...
- Open Directory - Computers: A...
- Robotics - Web Site Design by I...
- Search Engine Dictionary: Spid...
- Stephen E. Jones: Creation/Evol...
- Genetic Programming Artificial ...
- Google Directory - Science &gt

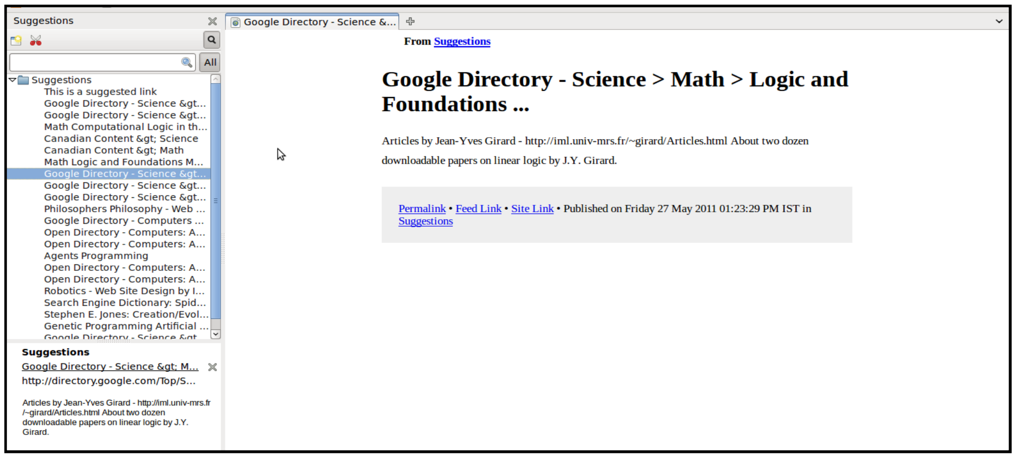
Suggestions

Google Directory - Science &gt; M...

http://directory.google.com/Top/S...

Articles by Jean-Yves Girard - http://iml.univ-mrs.fr /~girard/Articles.html About two dozen downloadable papers on linear logic by J.Y. Girard.

5.3.3. Dedicated tab for each link



Suggestions
This is a suggested link
Google Directory - Science &gt...
Google Directory - Science &gt...
Math Computational Logic in th...
Canadian Content &gt; Science
Canadian Content &gt; Math
Math Logic and Foundations M...
Google Directory - Science &gt...
Google Directory - Science &gt...
Google Directory - Science &gt...
Philosophers Philosophy - Web ...
Google Directory - Computers ...
Open Directory - Computers: A...
Open Directory - Computers: A...
Agents Programming
Open Directory - Computers: A...
Open Directory - Computers: A...
Robotics - Web Site Design by I...
Search Engine Dictionary: Spid...
Stephen E. Jones: Creation/Evol...
Genetic Programming Artificial ...
Google Directory - Science &gt...

Suggestions
Google Directory - Science &gt; M...
http://directory.google.com/Top/S...

Articles by Jean-Yves Girard - http://iml.univ-mrs.fr/~girard/Articles.html About two dozen downloadable papers on linear logic by J.Y. Girard.

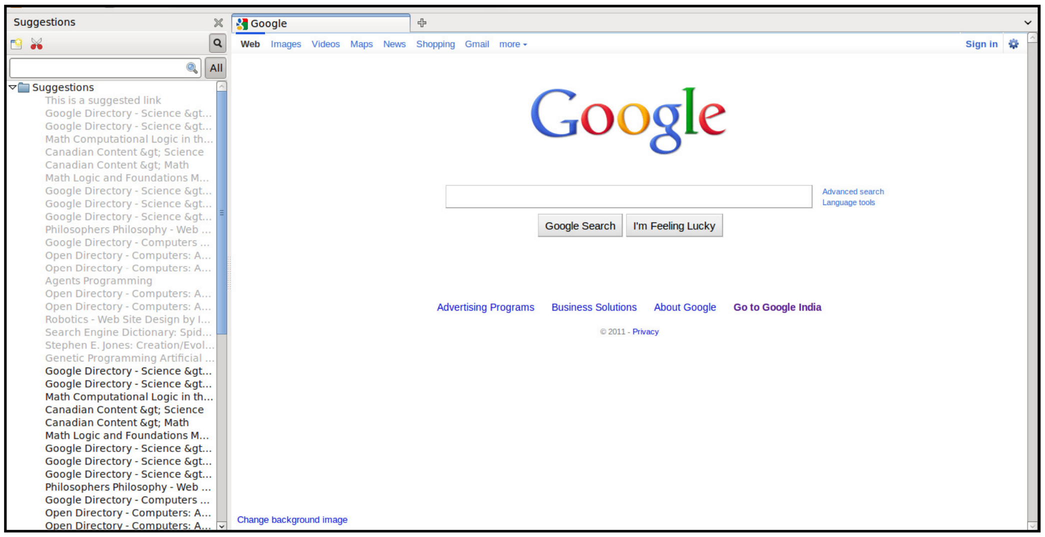
From Suggestions

Google Directory - Science > Math > Logic and Foundations ...

Articles by Jean-Yves Girard - http://iml.univ-mrs.fr/~girard/Articles.html About two dozen downloadable papers on linear logic by J.Y. Girard.

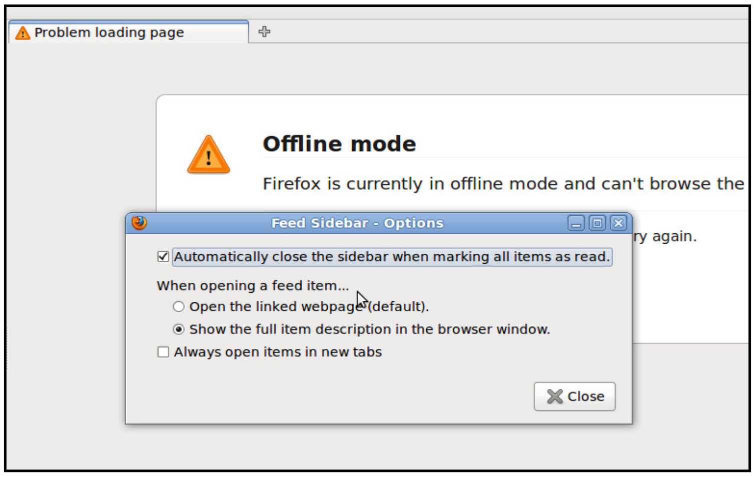
Permalink • Feed Link • Site Link • Published on Friday 27 May 2011 01:23:29 PM IST in Suggestions

5.3.4. Links When Marked as Read



Suggestions
This is a suggested link
Google Directory - Science &gt...
Google Directory - Science &gt...
Math Computational Logic in th...
Canadian Content &gt; Science
Canadian Content &gt; Math
Math Logic and Foundations M...
Google Directory - Science &gt...
Google Directory - Science &gt...
Google Directory - Science &gt...
Philosophers Philosophy - Web ...
Google Directory - Computers ...
Open Directory - Computers: A...
Open Directory - Computers: A...
Agents Programming
Open Directory - Computers: A...
Open Directory - Computers: A...
Robotics - Web Site Design by I...
Search Engine Dictionary: Spid...
Stephen E. Jones: Creation/Evol...
Genetic Programming Artificial ...
Google Directory - Science &gt...
Google Directory - Science &gt...
Math Computational Logic in th...
Canadian Content &gt; Science
Canadian Content &gt; Math
Math Logic and Foundations M...
Google Directory - Science &gt...
Google Directory - Science &gt...
Google Directory - Science &gt...
Philosophers Philosophy - Web ...
Google Directory - Computers ...
Open Directory - Computers: A...
Open Directory - Computers: A...

5.3.5. Options Pane to Set Configurable Parameters



Offline mode

Firefox is currently in offline mode and can't browse the

ry again.

[✓] Automatically close the sidebar when marking all items as read.

When opening a feed item...

○ Open the linked webpage (default).

[✓ Show the full item description in the browser window.]

☐ Always open items in new tabs

6. Project Plan

6.1. Proposed Plan for the Execution of Project

The Gantt chart below represents the split-up of our work schedule of our project. Following the design phase, the major part of our work was planned to be spent on implementation, as our team was inexperienced in the domain of the problem statement.

Activity	14 Jan	28 Jan	11 Feb	25 Feb	11 Mar	25 Mar	8 Apr	15 Apr	22 Apr	29 Apr	6 May	13 May	20 May	27 May
Project Preparations	🟦													
Feasibility Study	🟦	🟦	🟦											
Requirement analysis	🟦	🟦	🟦											
System Design			🟦	🟦										
Algorithm Design			🟦	🟦	🟦									
Add-On Implementation					🟦	🟦	🟦	🟦	🟦	🟦	🟦			
Algorithm Implementation					🟦	🟦	🟦	🟦						
Testing									🟦	🟦	🟦	🟦	🟦	
Documentation	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦

Figure 6.1– Gantt Chart for Proposed Plan

6.2. Realized Plan for the Execution of the Project

The Gantt diagram below represents the actual split-up of time spent in execution. Most of the split-up in the proposed plan was followed, barring the extra weeks spent in the continuous revision and development of the Add-on Implementation related issues.

Activity	14 Jan	28 Jan	11 Feb	25 Feb	11 Mar	25 Mar	8 Apr	15 Apr	22 Apr	29 Apr	6 May	13 May	20 May	27 May
Project Preparations	🟥													
Feasibility Study	🟥	🟥	🟥											
Requirement analysis	🟥	🟥	🟥											
System Design			🟥	🟥	🟥	🟥								
Algorithm Design			🟥	🟥	🟥	🟥								
Add-On Implementation					🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Algorithm Implementation						🟥	🟥	🟥						
Testing									🟥	🟥	🟥	🟥	🟥	🟥
Documentation	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

Figure 6.2– Gantt Chart for Realized Plan

7. Management Plan

7.1. Team Introduction

Our team consists of three members, none of us having any previous experience with the design and implementation of Browser Extensions. All of us had a working knowledge of JavaScript and database management systems.

7.2. Work Division

In our group, we assigned tasks and responsibilities as follows:

1) Kunal Sangwan – responsible for the design of the Ranking Algorithm and design and implementation of the Learning Engine.
2) Shashank Srikant – responsible for the design of the Ranking Algorithm and the design and implementation of the User-Interface Engine.
3) Sohil Arora – responsible for the design and implementation of the Semantic Analyzer and the Suggestion Engine.

Each was responsible for documenting and testing modules developed by him.

8. Development Cycle

The development strategy used was basically a combination of both the **agile software development model** and the **iterative and incremental software development model.**
The whole system was complex and included many functional parts to perform together to provide the system's required functionality. A lot of the functionality is part of the back-end and completely invisible to the user: intercepting page loads, generating the feature vector from each page, calculating and assigning weights to each topic using the developed algorithm and supporting parameters and the generation of suggestions for the user. Additionally, there are the front-end functionalities relating to display and manipulation of suggestions. Therefore, the functionalities were developed through various iterations, each iteration adding a new functionality to the system.
A component based development approach was employed and the project was broken down into smaller components, wherein each component provided exclusive functionality or supported other components to help them deliver their functionality. These components were developed as a part of various iterations.

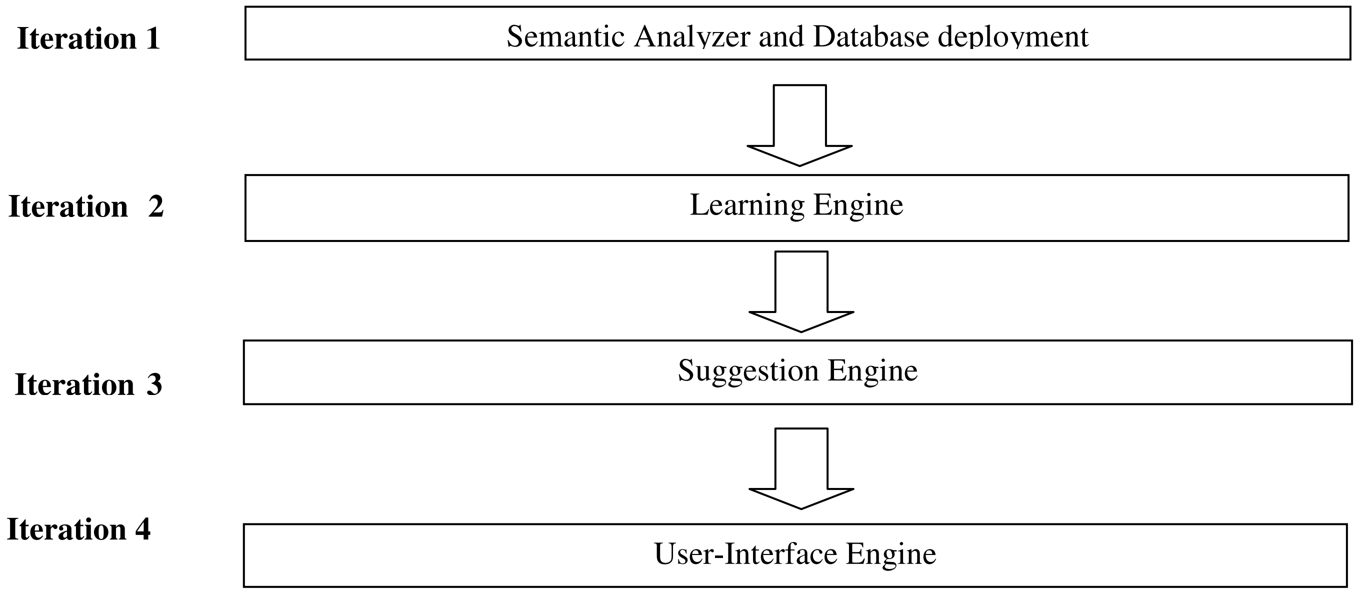


Figure 8.1– Development Cycle of the Project

The team was small, comprising of three members only. Each team member was, as a result, responsible for developing a complete component and following it up with testing and integrating the functionality with the overall system.

As a part of the first iteration, the page intercepting mechanism was implemented as was the mechanism for extracting topics from the intercepting page, thus completing the Semantic Analyzer. Also, an initial instance of the supporting database was put into place. This database instance witnessed many changes during other iterations, based on project requirements realized. The major components developed by distinguishable iterations are mentioned in Figure 8.1.

9. Implementation

9.1. The XPI

A Mozilla Firefox Extension is packaged as an XPI (Cross-Platform Installer). They are simply compressed ZIP files renamed to the XPI extension. The basic structure of an XPI is as follows:

9.1.1. XPI Contents

sampleextension1.xpi

- o *install.rdf* information needed to install add-on
- o *chrome.manifest* tells Firefox where to look for chrome files
- o *chrome* set of user elements outside of a window’s content area
    - ▪ *content* user interface and script files
        - browseroverlay.xul
        - browseroverlay.js
    - ▪ *locale* the text used in the extension
        - en-us
            - o browseroverlay.dtd
            - o browseroverlay.properties
    - ▪ *skin* files defining the look and feel of the UI
            - o browseroverlay.css

Each file in the XPI serves a specific purpose:

- **install.rdf**: Contains information needed to install the add-on.
- **chrome.manifest**: Tells Mozilla Firefox where to look for chrome files.
- **chrome**: The chrome folder contains the set of user elements outside of a window’s content area.
- **content**: The content folder contains user interface and script files.
- **browseroverlay.xul**: The basic user interface file, this provides a front-end for the user to view.
- **browseroverlay.js**: A JavaScript file, this provides all the functionality of the extension.
- **locale**: The locale folder contains language-specific content that is displayed in the extension.

- **skin**: The skin folder contains CSS files that define the look and feel of the user interface.

9.2. Languages Used

Mozilla Firefox Extensions are developed using a standard set of languages:

i. XUL:

This is an XML user interface markup language developed by the Mozilla project. Relying on other web technologies such as CSS and JavaScript, XUL is used for the design of the user interfaces on Mozilla applications and their extensions. In Suggestions4You, XUL elements Sidebar and Tree Hierarchy have been used for displaying suggestions to the user.

ii. JavaScript:

An object-oriented scripting language, JavaScript is used for all the functionality that an extension performs. In Suggestions4You, JavaScript is used for implementing the Semantic Analyzer and the Suggestion Engine.

iii. CSS (Cascading Style Sheets):

CSS is a style sheet language used to describe the look and formatting of web content. All data displayed is formatted using CSS.

iv. SQL:

Mozilla Firefox supports a Database Engine called SQLite. All the data being stored and retrieved by the Suggestions4You is managed using SQL operations on SQLite.

v. Java:

Java has been used for the development of the Learning Engine. It has been chosen for this purpose because of the platform independency it offers.

9.3. APIs used

Suggestions4You uses two Application Program Interfaces for its working:

9.3.1. Textwise

The Textwise API is used by the Semantic Analyzer to obtain the topics relating to every page visited by a user.

The API, when called with the URL of a page as a parameter, processes the content of the page and returns as an XML response which can then be parsed to obtain weight and index pairs. The index returned is mapped to the topics using a configuration file.

9.3.2. Bing

The Bing API [11] is used by the Suggestion Engine to obtain search results for custom-generated search queries. These search results are then used as suggestions for the user.

The API, when called with a search query as a parameter, returns the search results for that query in an XML format, which can then be parsed to obtain search results in any desired format.

9.4. Implementation Specifics

The following article discusses some of the implementation concerning each of the major components of our system.

9.4.1. Semantic Analyzer

The Semantic Analyzer is being used to process every page visited by the user and extract topics from it. It has been developed using JavaScript and uses a 3rd party API call to Textwise [2].

The component intercepts every page when it is loaded and sends its URL as a parameter in the Textwise API call, which in turn returns an XML response. This response is then parsed to obtain (weight, index) pairs.

Also, this component creates the parameters needed by the Learning Engine for ranking topics. It calculates the duration for which a page, and thus its topics, was active in the browser. Also, it keeps record of the parent of every page so that parent-child relationships between pages may be established.

All the data generated by the Semantic Analyzer is stored in the SQLite Database for use by the Learning Engine.

9.4.2. Leaning Engine

The Learning Engine is used to rank topics, given a set of supporting parameters. It has been developed using Java.

9.4.3. Suggestion Engine

The Suggestion Engine uses the final ranks generated by the Learning Engine and creates suggestions to be presented to the user. It has been developed in JavaScript and uses a 3rd party API call to Bing.

The component creates custom search queries for higher ranked topics and sends them as parameters in the Bing API call [11], which in turn returns an XML response. This response is then parsed to obtain the search results in clear text. These search results are then written onto the database for later use by the User-Interface Engine.

9.4.4. User-Interface Engine

The User-Interface Engine uses the search results generated by the Suggestion Engine and presents them to the user as suggestions. It has been developed using XUL and JavaScript.

The component uses XUL to create a handy sidebar which is displayed when a button on the Navigation Toolbar of Mozilla Firefox is clicked. Suggestions read by the user are hidden from the user's view, though he can choose to view them as well, and replaced by fresh suggestions.

10.Test Plan

10.1. Capturing session details

Test Case	Expected Output	Input	Result
New link is clicked on a web page and the parent of the web page is captured.	The url of the web page is stored in the database with the parent_url set to the current web page.	url on the page www.computerworld.com is clicked.	The url is inserted in the table session_details with parent_url set to the parent web page.
If New url is entered in the address bar then the database should be updated accordingly and the parent url is set to null.	The url of the web page is stored in the database with parent_url set to null.	url www.computerworld.com is entered in the address bar.	The session_details table contains a new entry that corresponds to the url entered with parent_url set to null.
Whenever a new url is visited the tags corresponding to the page visited are fetched from the semantic engine.	The tags corresponding to the url visited are fetched properly and inserted into the database.	url www.computerworld.com is entered in the address bar.	The tags are fetched with the help of the api and inserted into the table page_keywords.
The time spent on a particular web page is captured properly.	The page_open_time and the page_close_time for the currently opened page are updated properly.	url www.wikipedia.org is visited and 300 secs were spent on the page.	The page_open_time is updated at the time the page was visited and the page_close_time set to null. When the page is closed the page_close_time is updated to the current time.

10.2. Sidebar

Test Case	Expected Output	Input	Result
Clicking the button on the address bar toggles the sidebar.	The sidebar is toggled on clicking the sidebar.	The button on the address bar is pressed.	The sidebar state is toggled when the button is clicked.
The "All" option is pressed, the sidebar displays all the read and unread links in the sidebar.	The read links are displayed in faded font and the unread links are displayed in solid color font on clicking the "All" button.	"All" button is clicked.	The sidebar shows all the read and the unread links in respective colors.
In offline mode, double clicking any of the links opens up the description of the link in the content pane.	The description is opened in the new pane and the link should not be marked as read.	In offline mode, click one of the unread links.	The description is displayed in the content pane. The color of the link fades making it as read link.
In online mode, double click on an unread link opens up the link into the content pane.	The link is displayed in the content pane and the link is marked as read.	In online mode, click one of the unread links.	The link is displayed in the content pane and the link is marked as read link.
On marking a link read none of the new link added to the sidebar are read.	The new links added are not already in the list.	Keep marking links as read until the number of links in the sidebar starts decreasing.	The links are getting repeated.
On choosing the option "open all in tabs" and large number of tabs has to opened the browser prompts with a warning message.	The browser prompts with a warning message in case number of tabs to opened are large.	There are around 20 unread links, and "open all in tab option is chosen'.	The browser prompts with the appropriate message regarding the
Choosing "mark suggestion as read" option marks all the links in the sidebar as read.	The unread links in the sidebar gets converted to the read links.	Option "mark suggestion as read" is chosen.	The unread links gets faded to depict there read status.
When the number of	The read links should	The "maxReadCount"	When the browser is

link in the sidebar goes above maxReadCount preference then at next startup the number of such links should be removed.	get removed at the startup if the read likns count is more than the preference.	preference was set to 10 and the number of unread links is 12. In this state the browser was restarted.	restarted the number of unread links are 10 and the read links from the top are deleted.
When the number of unread links in the sidebar is less than the “suggestionsCount” saved in the preference, new links will be added to the list.	The total number of unread links is equal to the “suggestionsCount” preference.	Marked a suggestion or multiple suggestions as read.	The new suggestions are inserted into the sidebar.

10.3. Testing the algorithm

Test Case	Expected Output	Input	Result
The program is executing at fixed intervals.	The program is executed at the regular intervals and the new final weights are inserted into the database.	Wait for the time interval set.	The program is executing. The final_weights table is filled with the new weights.

11.Limitations and Future Scope

The project has immense scope for improvement and further development. Some of the major limitations of our project are –

- Browser Dependence: Given the current state of development facilities provided by various commercial web-browsers, we found that Mozilla Firefox has the best online community to support such an extension. As a result, all the work provided here is supported only on version of Mozilla Firefox.
- Semantic Engine Dependence: As we have not implemented the semantic analysis engine ourselves and depend on a commercial engine for our results, the entire tagging and the weights of the topics corresponding to a given page are calculated on the basis of the interpretation of this commercial engine. No modification to the algorithms can be performed from our side in order to better suit our problem domain.
- Effective Query Formation: The topics of interest realized can be organized and combined to form possibly effective queries. The current implementation witnesses a simple word/combination of words-based querying. This is scope for research in the field of language processing and string query formation.

The current status of our project qualifies to make it a prototype for such the suggestion system we initially conceived and proposed. If the above mentioned limitations are overcome, we may consider the following extensions to this project –

- An interactive feedback system: The User may be given a choice to provide a feedback on the nature of links being suggested to her. In the case that the suggestion engine has wrongly inferred topics of interest, this form of a supervised learning feature should help correct its inference and re-suggest links. The current implementation has been provided keeping in mind such a possible extension with enough support for its easy integration.
- Mobile-agent based engine: This shall involve enhancing a local classifier database by crawling through similar add-on data-stores in a network.
- Multiple browser integration: Based on the extension services provided by other web-browsers, we may develop the same service for them.

12.References

1. *Weka*, A Collection of machine learning algorithms for solving data mining problems implemented in Java and open sourced under the GPL, www.cs.waikato.ac.nz/ml/weka/
2. *TextWise*, is a Web service that automatically extracts tags from a piece of text. The tags are chosen based on both statistical and linguistic analysis of the original text, http://textwise.com/api
3. Thomas Hofmann, *Probabilistic Latent Semantic Indexing*, Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99), 1999
4. *IEEE 830-1998,* IEEE defined standards for the Software Requirements Specifications of a software product, http://standards.ieee.org/findstds/standard/830-1998.html, Last visited on 20 November 2010.
5. *IEEE 1016-2009,* IEEE defined standards for the Design Documentation of a software product, http://standards.ieee.org/findstds/standard/1016-2009.html, Last visited on 20 November 2010.
6. *StumbleUpon,* a discovery engine (a form of web search engine) that finds and recommends web content to its users. htrtp://www.stumbleupon.com/. Last visited on 20 April 2011
7. *Software Ideas Modeler*, an online, open-source, free CASE Tool, http:// www.softwareideas.net/, Last visited on 20 November 2010.
8. *Microsoft Visual Studio 2010, Student Edition,* A tool for generating Use-Cases for software systems, https://www.dreamspark.com/, Last visited on 1 November, 2010.
9. *MySQL Workbench,* A tool for generating E-R Diagrams of database schema, http://dev.mysql.com/downloads/workbench/, Last visited on 1 November, 2010.
10. *XUL, XML User Interface Language,* An introduction to XUL, https://developer.mozilla.org/en/XUL, Last visited on 20 May, 2011.
11. *Bing API,* An API to the Microsoft Bing Search Engine which provides search results over data formats like XML and JSON, http://www.bing.com/developers