PERSONALISED SEARCH ENGINE USING TWITTER DATA ANALYSIS

A PROJECT REPORT
Submitted by

Reshma Malla (1BM12IS058)
Suhas H V (1BM12IS085)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
INFORMATION SCIENCE AND ENGINEERING

Under the Guidance of

Dr. N Sandeep Varma
Assistant Professor, ISE,
BMSCE

BMS COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
MAY 2016

Department of Information Science and Engineering
B M S College of Engineering, Bull Temple Road, Bangalore 560019

BONAFIDE CERTIFICATE FOR EACH STUDENT

This is to certify that the project report titled, "**Personalized Search Engine using Twitter Data Analysis**" is a bonafide record of the project work done by **RESHMA MALLA** with University Seat Number No. 1BM12IS058 during the academic year 2015-2016.

Head of the Department Signature: Name: Seal: Date:	Guide Signature: Name: Designation: Date:

Submitted for the University Examination held on ……..

Internal Examiner Signature: Name: Designation: Date:	External Examiner Signature: Name: College / Designation: Date:

Department of Information Science and Engineering
B M S College of Engineering, Bull Temple Road, Bangalore 560019

BONAFIDE CERTIFICATE FOR EACH STUDENT

This is to certify that the project report titled, "**Personalized Search Engine using Twitter Data Analysis**" is a bonafide record of the project work done by **SUHAS H V** with University Seat Number No. 1BM12IS085 during the academic year 2015-2016.

Head of the Department Signature: Name: Seal: Date:	Guide Signature: Name: Designation: Date:

Submitted for the University Examination held on ……..

Internal Examiner Signature: Name: Designation: Date:	External Examiner Signature: Name: College / Designation: Date:

BMS COLLEGE OF ENGINEERING BANGALORE 560019

CERTIFICATE FROM THE SUPERVISOR/S (for the student batch)

Department: Information Science and Engineering

Candidates Degree Registered for: B.E.

Candidate details:

SL. NO.	Student Name	USN	Student's Signature
1.	Reshma Malla	1BM12IS058	
2	Suhas H. V	1BM12IS085	

Certified that these candidates are students belong to Information Science and Engineering of BMS College of engineering. They have carried out the project work of titled "Personalized Search Engine using Twitter Data Analysis" as final year dissertation project. It is in partial fulfilment for completing the requirement for the award of B.E. degree by VTU. The works is original and duly certify the same.

Sl No.	Supervisor Name	Department/Organization/Designation	Signature
1	Dr N Sandeep Varma	Information Science and Engineering Assistant Professor, BMSCE	
2	Dr Shambhavi B R	Information Science and Engineering Associate Professor, BMSCE	

HOD's Signature Seal

Date:

Department of Information Science and Engineering
B M S College of Engineering, Bull Temple Road, Bangalore 560019

CERTIFICATE FROM GUIDE

Certified that this project work **"Personalized Search Engine using Twitter Data Analysis"** is their true work**.** This project work was carried out under my supervision. This is to certify that I have gone through complete project report. The students have incorporated all the correction and suggestions made by me.

Guide 1

Signature:

Name:

Seal:

Date:

HOD

Signature:

Name:

Seal:

Date:

Acknowledgements

We take up this opportunity to express our gratitude to everyone who have supported us through the course of this project. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. We are sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

We express our warm thanks to our guides Dr N Sandeep Varma & Dr Shambhavi B R, Department of Information Science and Engineering, BMSCE, for their support and guidance provided throughout the project. We would like to thank Assistant Professor Shubha Rao for coordinating all our project schedules. We also want to thank Dr Gowri Shankar, HOD of the Department of Information Science and Engineering, BMSCE, and Dr Mallikarjuna Babu, Principal, BMSCE for providing us the facilities to work on this project.

Thank you,

Reshma Malla

Suhas H V

Synopsis

A web search engine is a software system that is designed to search for information on the World Wide Web. The search results are generally presented in a line of results often referred to as search engine results pages (SERPs). Search engines are the foundation of the Internet. Most users will turn to a search engine as the quickest way of finding the information, or product that they want. As many website owners rely on search engines to send traffic to their website, an entire industry has grown around the idea of optimizing Web content to improve one's placement in search engine results.

An internet user conducting a search has a specific intent and is looking for something in particular. The goal of the Search Engines is to provide users with search results that lead to "relevant" information on high-quality websites. We develop a Web Application that personalizes the results returned by Google Search Engine based on a user's Tweets. Twitter data analysis helps filter the results of his search to better determine the results expected by the user. The analysis performed will take into consideration various parameters based on his tweets. The results obtained will hence, be more personalized and apt.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

1. Introduction

1.1 Overview

The World Wide Web consists of tens of millions of Web servers (computers hosting Web sites) and billions of Web pages, all interconnected [1]. Each page is accessible by typing in an address (called a URL, for uniform resource locator). The problem the researcher faces is how to get the address of a page that may contain useful information.

Printed directories were a start, but no book could contain billions of addresses, and any printed material would be out of date before it could leave the printing plant.

Then the search engine was created. Because all of the pages are interconnected by hyperlinks, where one page links to another (or usually several others), it is possible to move all around the Web just by following links. Search engines surf automatically by using programs called bots or spiders.

The purpose of this project is to use the recent advances in the field of data sciences to create and optimise a search engine that will analyse the user's twitter data to predict his interests and preferences and uses this information to provide more personalised results.

1.2 Motivation

The task of a search engine spider is to visit as many sites as possible by following links (both within the site and from site to site), index the pages, and use this index to help users of the search engine find the pages they want.

Several factors determine how well a search engine performs, including the size of its index, the freshness of the content, and the ease of searching. Google excels in these areas and several more that are relevant to producing high quality results. However, these results don't take into account, the user's personal interests. They are generalised based on mass public opinion and lack a sense of personalisation. Hence, the core motivation of this project is to overcome these shortcomings of the existing search engines and design a web application, where Google search results are further refined and personalized based on the user's preferences. The user's preferences are obtained by carrying out an analysis on his tweets and tweets of people that he follow to better determine his interests.

1.3 Objective

- To develop a web application that personalises search.
- The application must be able to receive authorisation from the user to access the user's twitter timeline.
- To extract the tweets of the user and his followers using twitter api.
- To perform an analysis on the extracted tweets and store the analysed information in an organised format.
- To use this organised information to perform google search and personalise the results obtained.

1.4 Scope

Modern Search Engines are evolving every day and experts in the field such as Google, Yahoo, Bing etc. are trying to perfect their algorithms to perform an accurate search. This project will explore a new avenue to perform this task using twitter data analysis. The finished product can be deployed as a web application hosted on a server or the algorithm proposed to personalize search, can be incorporated into the existing search algorithms to provide the user with better results.

1.5 Existing System

Currently, existing search engines do not take into account the personal interests of the user, while performing the search. The results are displayed based on relevance and popularity. They use proprietary algorithms to determine the order of the results. They only take into account the user's previous searches to determine his interests.

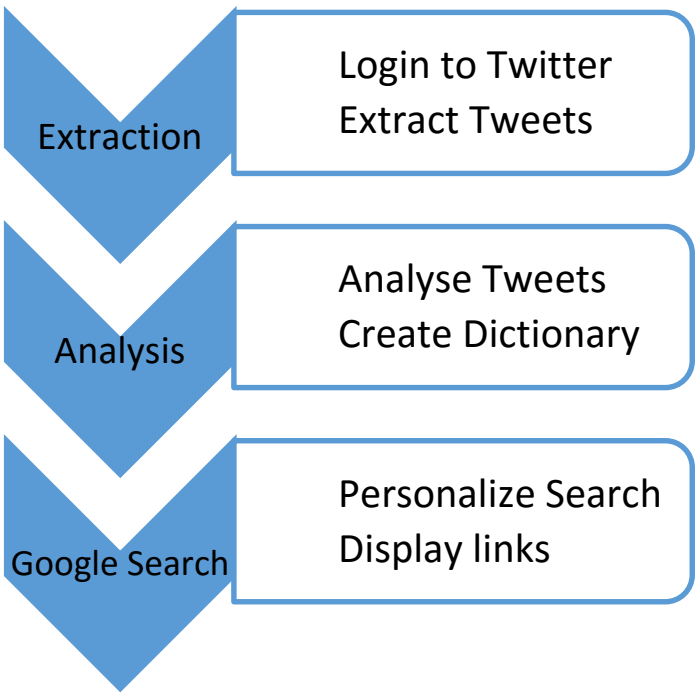
1.6 Problem Statement

To develop an application that personalizes Google search results using twitter data.

1.7 Proposed System

The following figure depicts the proposed system:

Figure 1.7.1: Proposed System



CHAPTER 2

LITERATURE SURVEY

2. Literature Survey

2.1 Search Engines

A search engine is simply a database of web pages, a method for discovering and storing information about new web pages, and a way to search that database. Therefore, in the simplest form, search engines perform three basic tasks:

1. Traverse the web and determine important words (crawling)
2. Build and maintain an index of sites' keywords and links (indexing)
3. Present search results based on reputation and relevance to users' keyword combinations (searching)

The primary goal is to effectively present high-quality, precise search results while efficiently handling a potentially huge volume of user queries.

For most users, Web search engines are the central starting point for their exploration of Web content. Search engines lead us to new websites we have never heard of, help us re encounter familiar websites and offer us a wide variety of content from the many sources of the Web, which we would not be able to discover with other tools. Most users use search engines every day, and the amount of queries entered into general-purpose Web search engines such as Google worldwide exceeds 100 billion queries per month. Even though most users use search engines every day, they know very little about them.

When discussing Web search engines, in most cases one arrives quickly at a discussion of Google. In fact, Google is often seen as synonymous with Web search. However, smaller companies are active, even though they usually focus on niche markets or business applications. A major reason for this is that while search may be highly profitable for smaller companies in these specialised areas of search, the high costs of building and maintaining a search engine on the scale of the Web lead to a concentration on the search engine market, with just a few major players left.

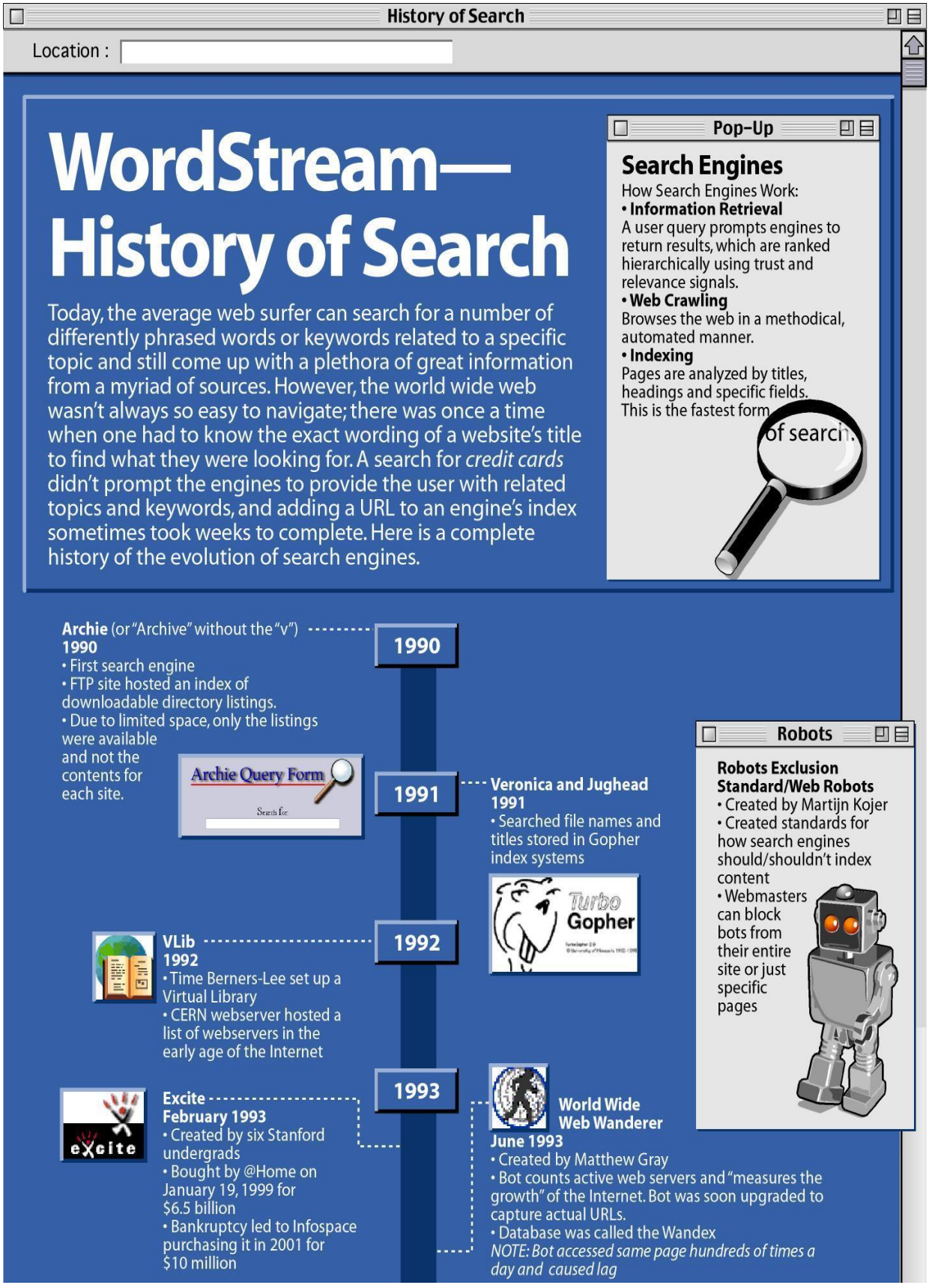
It may be irritating to see that many search engines claiming to search the "whole of the Web" are available on the market; however, only a few of them have their own, Web-scale index. Outside of these few, most search engines license search results from other search engines, the most famous example being Yahoo using results from Microsoft's Bing search

engine. When discussing the search engine market, it is often forgotten that while search engines are surely commercial enterprises, they also serve as facilitators of information, and therefore, that they serve the interests of the public. When considering that mainly one search engine is used, one has to ask whether this one search engine does indeed serve these interests.

Different internet search engines use different algorithms for determining which web pages are the most relevant for a particular search engine keyword, and which web pages should appear at the top of the search engine results page.

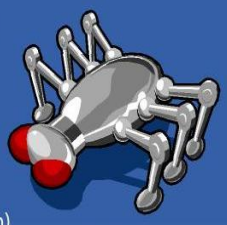
Relevancy is the key for online search engines – users naturally prefer a search engine that will give them the best and most relevant results. Search engines are often quite guarded with their search algorithms, since their unique algorithm is trying to generate the most relevant results. The best search engines, and often the most popular search engines as a result, are the ones that are the most relevant.

2.2 The History of Search Engines - An Infographic



ALIWEB
October 1993
Created by Martijn Koster
• Crawled meta info and allowed the user to submit the pages they wanted indexed along with a description
• No bot
• Not using excessive bandwidth
NOTE: people didn't know how to submit their sites

Primitive Web Search
December 1993
• JumpStation: Info about page's title and header using simple linear search
• World Wide Web Worm: Indexed titles and URLs
(these two listed results in the order they were found without ranking)
• RBSE Spider (had a rating system)
(unless exact title was a match it was extremely hard to find anything)



1994

Infoseek
January 1994
• Webmasters could submit a page in realtime
December 95: Netscape began using them as their default search

E|Net Galaxy
January 1994
• Efficient in its utilization of different web search features
• Unnecessary library due to small size of Internet

WebCrawler
April 1994
• First crawler that indexed entire pages
• Too popular to be used during daytime
• **June 1995:** AOL buys WebCrawler
• **1997:** Excite bought out WebCrawler
• AOL begins using Excite

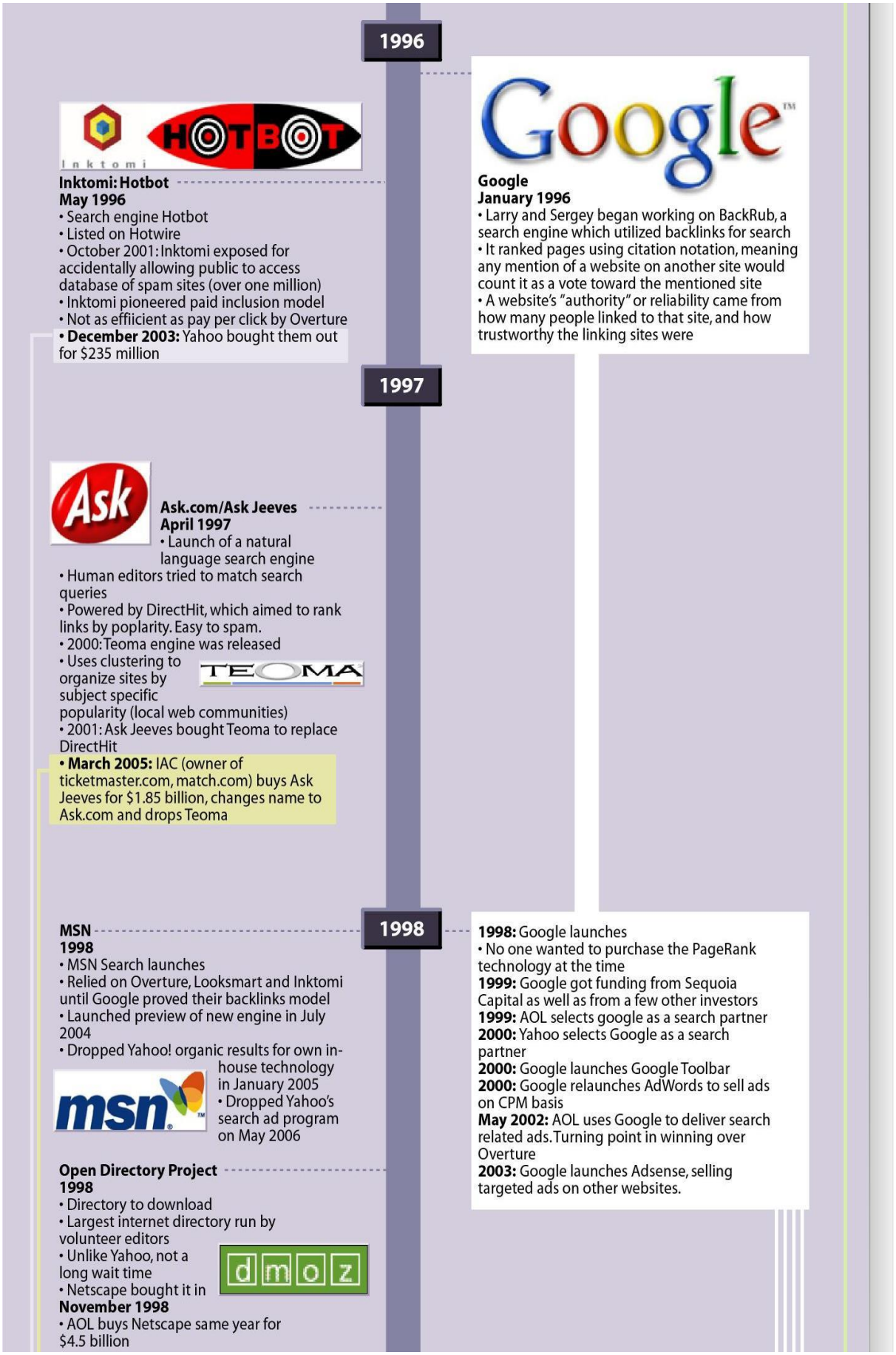
Lycos
July 1994
• Went public with catalog of 54,000 documents
• Ranked Relevance retrieval
• Prefix matching and word proximity
• August 1994: Had identified 394,000 documents; 1.5 million by January 1995
• November 1996: Had identified 60 million documents (more than any other search engine)
• October 2004: Lycos was sold to Daum Communications, the second largest Internet portal in Korea

AltaVista
January 1994
• Unlimited bandwith (for the time)
• First to allow natural language queries
• Advanced searching techniques
• Add or delete your own URL within 24 hours
• Search tips and new features
• Inktomi and Google sent them into irrelevancy
February 2003: Overture intends to purchase AltaVista for $80 million in stock, $60 million in cash
• Yahoo bought Overture in 2003

Yahoo! Directory
April 1994
• Created by David Filo and Jerry Yang
• Began as a collection of favorable web pages
• Increasing size influenced them to become a searchable directory
• A man-made description with each URL
• Informational sites added for free, but they expanded to include commercial sites. This is still available for $300 a year.
• Long wait time to be included
• Yahoo! didn't work on their search engine until **2002** when they began acquiring other search directories. Until that time they had outsourced their search services

1995

LookSmart
1995
• Competed with Yahoo by increasing inclusion rates back and forth
• **1998:** Bought non-commercial directory Zeal for $20 million
• **2002:** Transitioned into a pay per click provider, destroying reliability. Began depending on MSN by syndicating links through their portal
• **2002:** Bought WiseNut
• **2003:** Felt the sting of rejection when it was dumped by Microsoft and lost more than 65% of its annual revenue
• **2006:** Shut down Zeal


1996
Inktomi: Hotbot
May 1996
• Search engine Hotbot
• Listed on Hotwire
• October 2001: Inktomi exposed for accidentally allowing public to access database of spam sites (over one million)
• Inktomi pioneered paid inclusion model
• Not as effiicient as pay per click by Overture
• December 2003: Yahoo bought them out for $235 million
Google
January 1996
• Larry and Sergey began working on BackRub, a search engine which utilized backlinks for search
• It ranked pages using citation notation, meaning any mention of a website on another site would count it as a vote toward the mentioned site
• A website's "authority" or reliability came from how many people linked to that site, and how trustworthy the linking sites were
1997
Ask.com/Ask Jeeves
April 1997
• Launch of a natural language search engine
• Human editors tried to match search queries
• Powered by DirectHit, which aimed to rank links by poplarity. Easy to spam.
• 2000: Teoma engine was released
• Uses clustering to organize sites by subject specific popularity (local web communities)
• 2001: Ask Jeeves bought Teoma to replace DirectHit
• March 2005: IAC (owner of ticketmaster.com, match.com) buys Ask Jeeves for $1.85 billion, changes name to Ask.com and drops Teoma
1998
MSN
1998
• MSN Search launches
• Relied on Overture, Looksmart and Inktomi until Google proved their backlinks model
• Launched preview of new engine in July 2004
• Dropped Yahoo! organic results for own in-house technology in January 2005
• Dropped Yahoo's search ad program on May 2006
Open Directory Project
1998
• Directory to download
• Largest internet directory run by volunteer editors
• Unlike Yahoo, not a long wait time
• Netscape bought it in November 1998
• AOL buys Netscape same year for $4.5 billion
1998: Google launches
• No one wanted to purchase the PageRank technology at the time
1999: Google got funding from Sequoia Capital as well as from a few other investors
1999: AOL selects google as a search partner
2000: Yahoo selects Google as a search partner
2000: Google launches Google Toolbar
2000: Google relaunches AdWords to sell ads on CPM basis
May 2002: AOL uses Google to deliver search related ads. Turning point in winning over Overture
2003: Google launches Adsense, selling targeted ads on other websites.


1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
overture
Overture Services
• Formerly Goto.com
• The first company to successfully provide a pay-per-click placement search service
• Bought out AltaVista and AllTheWeb
• 2003: Acquired by Yahoo! Search Marketing as the beginning of their pay-per-click Internet advertising services
alltheweb
find it all
AllTheWeb
May 1999
• Sleek interface with advanced features
• February 2003: Bought by Overture for $70 million
• ATW was eventually rolled into Yahoo! Search
Snap
October 2005
• Overture owner Bill Gross launches Snap search engine
• Shows search volumes ,revenues, and advertisers
• Proved to be too complicated, not simplistic enough for the average web surfer
Classic Web Search
Enhanced Web Search
snap
LiveSearch
September 2006
• Microsoft announces launch of Live Search Product
Live Search
Web
Images
News
Local
QnA Beta
cuil
Cuil
June 2008
• Managed and developed by former Google employees
• Has indexed 127 billion web pages as of Feb. 2009
bing
Bing
June 2009
• Rebranding of MSN/live search
• Inline search suggestions for related searches directly in result set

2.3 Google

2.3.1 Background

The web is big. In 1998, the first Google index had 26 million pages and reached the 1 billion mark by 2000. As of July 2008, that number multiplied by a thousand, or, to be exact, 1 trillion. In late 2012, Google claimed to have indexed over 30 trillion unique individual live URLs on the web. Search has penetrated the very fabric of our globalized society. According to comScore, more than 158 billion searches were performed worldwide each month in 2011. This number equals approximately 5.2 billion searches performed every day, and 61,000 searches performed every single second every day [2]. The way we work, play, shop, and interact have changed, and the high demand for search will continue to escalate. Profit and non-profit organizations, as well as individuals looking to have a successful presence on the web need to adapt the way they create, publish, and distribute information. Search engines are closely tied to success in the new web economy.

2.3.2 Page Rank

Google's rise to success was in large part due to a patented algorithm called PageRank that helps rank web pages that match a given search string. When Google was a Stanford research project, it was nicknamed BackRub because the technology checks backlinks to determine a site's importance. Previous keyword-based methods of ranking search results, used by many search engines that were once more popular than Google, would rank pages by how often the search terms occurred in the page, or how strongly associated the search terms were within each resulting page. The PageRank algorithm instead analyzes human-generated links assuming that web pages linked from many important pages are themselves likely to be important. The algorithm computes a recursive score for pages, based on the weighted sum of the PageRanks of the pages linking to them. PageRank is thought to correlate well with human concepts of importance [3]. In addition to PageRank,

A Web page's PageRank depends on a few factors:

a. The frequency and location of keywords within the Web page: If the keyword only appears once within the body of a page, it will receive a low score for that keyword.

b. How long the Web page has existed: People create new Web pages every day, and not all of them stick around for long. Google places more value on pages with an established history.
c. The number of other Web pages that link to the page in question: Google looks at how many Web pages' link to a particular site to determine its relevance.

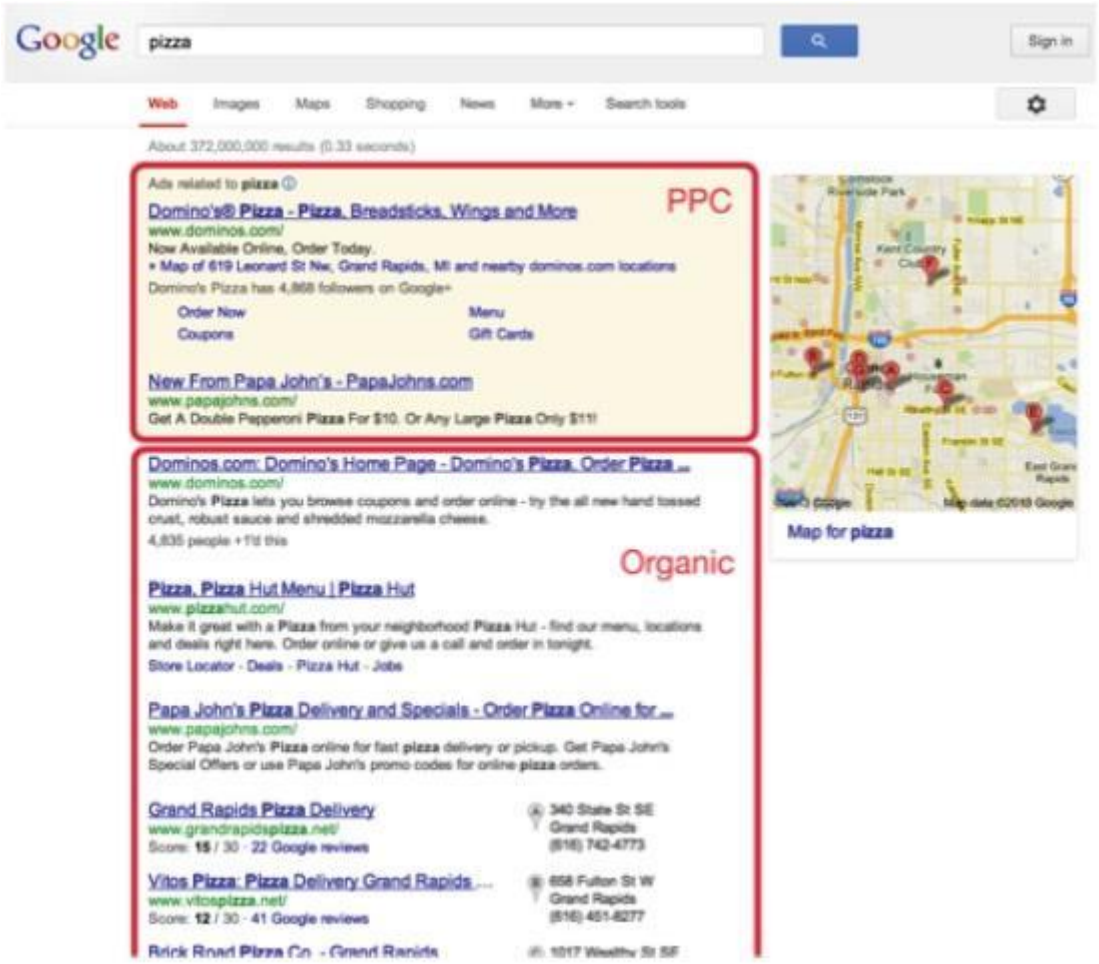
Google, over the years, has added many other secret criteria for determining the ranking of pages on result lists, reported to be over 250 different indicators, the specifics of which are kept secret to keep spammers at bay and help Google maintain an edge over its competitors globally.

2.3.3 Search Engine Optimisation

Search Engine Optimization is the process of increasing the number of visitors to a website by achieving high rank in the search results returned by a search engine. The higher a website ranks in the results pages, the greater the chance of users visiting the website. To do this, a website uses a set of optimization methods that manipulate dozens or even hundreds of its markup elements.

A search engine results page (SERP) is the listing of results returned by a search engine in response to a keyword query, and it contains two different sections: organic and PPC (Pay-per-click). The Organic section of a SERP contains results that are not sponsored or paid for in any way they rely strictly on search algorithms. The PPC section, on the other hand, contains text ads purchased from either Google AdWords or Microsoft AdCenter, using a bidding system to determine placements among other competing text ads. Figure 2.3.3.1 shows an SERP from Google.

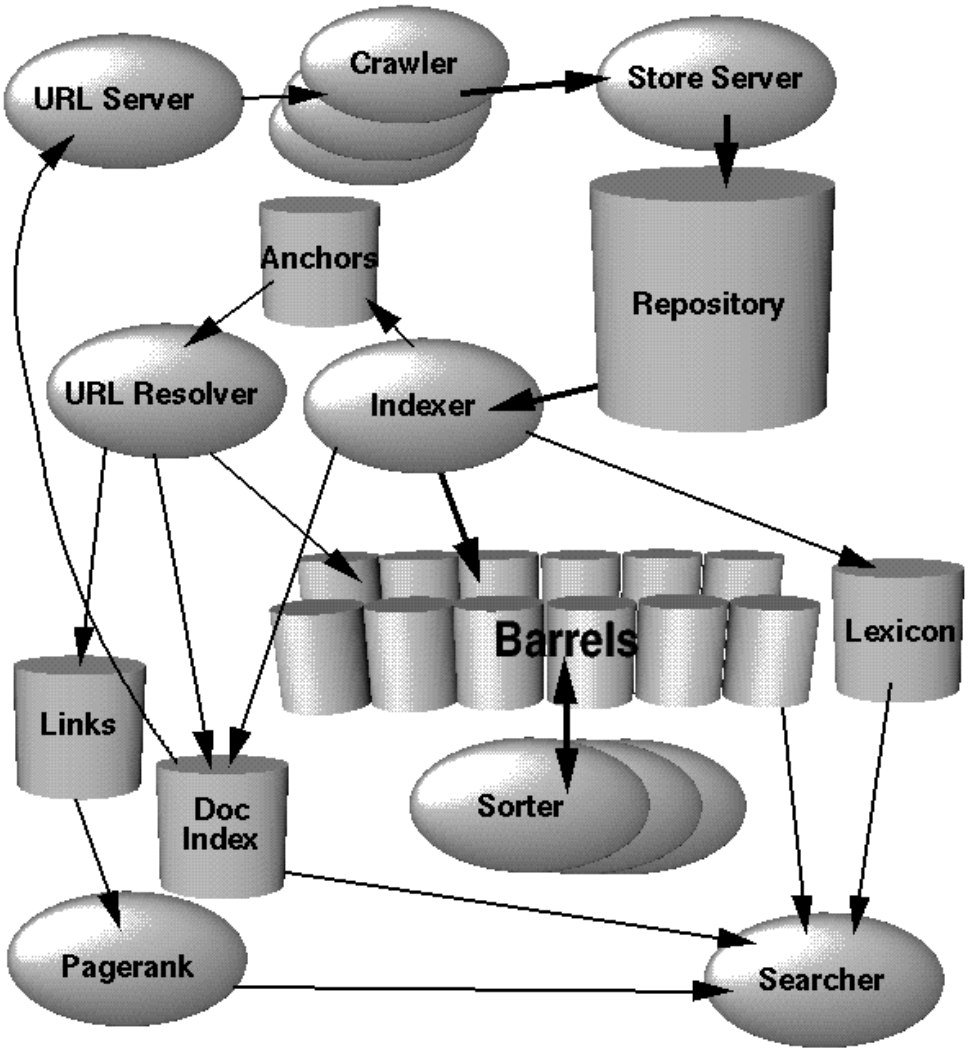
Fig 2.3.3.1: SERP from Google



2.3.4 High-Level Architecture

In order to deliver fast and relevant search results, Google must first discover the hundreds of millions of web pages that exist on the World Wide Web and organize them in an efficient manner. Google utilizes special application programs called web crawlers or spiders to automatically download a copy, or cache, of a web page and follow any links it may have. After downloading, a search engine will analyze the content of each page to discover lists of words and their occurrence, structural analysis, hyperlinks, and HTML validation.

Fig 2.3.4.1: High Level Architecture



2.3.5 Limitations of Search Engines

Search engines are far from perfect:

a. They rank websites in part according to concepts such as intrinsic authority, which in many cases are flawed, and which allow rankings to be manipulated.
b. They filter results according to the information that a particular user has given them, which rarely provides an accurate reflection of a user's real interests.
c. Although it is less of a problem than it used to be, search engine rankings have always been manipulated by keyword–stuffing: the inclusion of unnaturally large numbers of search terms into web pages.

d. Rankings are almost certainly affected in some way by the search engine companies' commercial interests: their dependence on paid advertising, and their promotion of their own products.

Part of the blame falls on the search engines' users:

a. People who use search engines, and people who own websites, underestimate the vast numbers of individual web pages that often touch on a particular topic.
b. Search engine rankings depend fundamentally upon the search terms that are used. Most users of search engines do not know how to create the most appropriate search terms.

Search engine rankings are frequently inaccurate. In order to understand why they are less than perfect, it is necessary to understand how the rankings are calculated. Search engines attempt to rank websites mainly according to two factors:

a. Significance, which can be increased by skilled search engine optimisation,
b. Popularity, which is largely out of the hands of the website's owner and its designer.

2.4 Data Analytics

Data Mining is primarily used today by companies with a strong consumer focus — retail, financial, communication, and marketing organizations, to "drill down" into their transactional data and determine pricing, customer preferences and product positioning, impact on sales, customer satisfaction and corporate profits. With data mining, a retailer can use point-of-sale records of customer purchases to develop products and promotions to appeal to specific customer segments. The following is a list of 14 other important areas where data mining is widely used:

➔ Future Healthcare

Data mining holds great potential to improve health systems. It uses data and analytics to identify best practices that improve care and reduce costs. Researchers use data mining approaches like multi-dimensional databases, machine learning, soft computing, data visualization and statistics. Mining can be used to predict the volume of patients in every category. Processes are developed that make sure that the patients receive appropriate care at the right place and at the right time. Data mining can also help healthcare insurers to detect fraud and abuse.

➔ Market Basket Analysis

Market basket analysis is a modelling technique based upon a theory that if you buy a certain group of items you are more likely to buy another group of items. This technique may allow the retailer to understand the purchase behaviour of a buyer. This information may help the retailer to know the buyer's needs and change the store's layout accordingly. Using differential analysis comparison of results between different stores, between customers in different demographic groups can be done.

➔ Education

There is a new emerging field, called Educational Data Mining, concerns with

developing methods that discover knowledge from data originating from educational Environments. The goals of EDM are identified as predicting students' future learning behaviour, studying the effects of educational support, and advancing scientific knowledge about learning. Data mining can be used by an institution to take accurate decisions and also to predict the results of the student. With the results the institution can focus on what to teach and how to teach. Learning pattern of the students can be captured and used to develop techniques to teach them.

➔ Manufacturing Engineering

Knowledge is the best asset a manufacturing enterprise would possess. Data mining tools can be very useful to discover patterns in complex manufacturing process. Data mining can be used in system-level designing to extract the relationships between product architecture, product portfolio, and customer needs data. It can also be used to predict the product development span time, cost, and dependencies among other tasks.

➔ CRM

Customer Relationship Management is all about acquiring and retaining customers, also improving customers' loyalty and implementing customer focused strategies. To maintain a proper relationship with a customer a business need to collect data and analyse the information. This is where data mining plays its part. With data mining technologies the collected data can be used for analysis. Instead of being confused where to focus to retain customer, the seekers for the solution get filtered results.

➔ Fraud Detection

Billions of dollars have been lost to the action of frauds. Traditional methods of fraud detection are time consuming and complex. Data mining aids in providing meaningful patterns and turning data into information. Any information that is valid and useful is knowledge. A perfect fraud detection system should protect

information of all the users. A supervised method includes collection of sample records. These records are classified fraudulent or non-fraudulent. A model is built using this data and the algorithm is made to identify whether the record is fraudulent or not.

➔ Intrusion Detection

Any action that will compromise the integrity and confidentiality of a resource is an intrusion. The defensive measures to avoid an intrusion includes user authentication, avoid programming errors, and information protection. Data mining can help improve intrusion detection by adding a level of focus to anomaly detection. It helps an analyst to distinguish an activity from common everyday network activity.

➔ Lie Detection

Apprehending a criminal is easy whereas bringing out the truth from him is difficult. Law enforcement can use mining techniques to investigate crimes, monitor communication of suspected terrorists. This filed includes text mining also. This process seeks to find meaningful patterns in data which is usually unstructured text. The data sample collected from previous investigations are compared and a model for lie detection is created. With this model processes can be created according to the necessity.

➔ Customer Segmentation

Traditional market research may help us to segment customers but data mining goes in deep and increases market effectiveness. Data mining aids in aligning the customers into a distinct segment and can tailor the needs according to the customers. Market is always about retaining the customers. Data mining allows to find a segment of customers based on vulnerability and the business could offer them with special offers and enhance satisfaction.

➔ Financial Banking

With computerised banking everywhere huge amount of data is supposed to be generated with new transactions. Data mining can contribute to solving business problems in banking and finance by finding patterns, causalities, and correlations in business information and market prices that are not immediately apparent to managers because the volume data is too large or is generated too quickly to screen by experts. The managers may find these information for better segmenting, targeting, acquiring, retaining and maintaining a profitable customer.

➔ Corporate Surveillance

Corporate surveillance is the monitoring of a person or group's behaviour by a corporation. The data collected is most often used for marketing purposes or sold to other corporations, but is also regularly shared with government agencies. It can be used by the business to tailor their products desirable by their customers. The data can be used for direct marketing purposes, such as the targeted advertisements on Google and Yahoo, where ads are targeted to the users of ever day search engines by analysing their search history and emails.

➔ Research Analysis

History shows that we have witnessed revolutionary changes in research. Data mining is helpful in data cleaning, data pre-processing and integration of databases. The researchers can find any similar data from the database that might bring any change in the research. Identification of any co-occurring sequences and the correlation between any activities can be known. Data visualisation and visual data mining provide us with a clear view of the data.

➔ Criminal Investigation

Criminology is a process that aims to identify crime characteristics. Actually crime analysis includes exploring and detecting crimes and their relationships with criminals. The high volume of crime datasets and also the complexity of

relationships between these kinds of data have made criminology an appropriate field for applying data mining techniques. Text based crime reports can be converted into word processing files. These information can be used to perform crime matching process.

➔ Bio Informatics

Data Mining approaches seem ideally suited for Bioinformatics, since it is data-rich. Mining biological data helps to extract useful knowledge from massive datasets gathered in biology, and in other related life sciences areas such as medicine and neuroscience. Applications of data mining to bioinformatics include gene finding, protein function inference, disease diagnosis, disease prognosis, disease treatment optimization, protein and gene interaction network reconstruction, data cleansing, and protein sub-cellular location prediction.

2.5 Twitter

Successful micro blogging services such as Twitter have become an integral part of the daily life of millions of users. In addition to communicating with friends, family or acquaintances, micro blogging services are used as recommendation services, real-time news sources and content sharing venues.

A user's experience with a micro blogging service could be significantly improved if information about the demographic attributes or personal interests of the particular user, as well as the other users of the service, was available [4]. Such information could allow for personalized recommendations of users to follow or user posts to read; additionally, events and topics of interest to particular communities could be highlighted.

The digital age introducing social media gave social networks an unprecedented impulse to flourish. Even though quite a few studies have been carried out on social media and social networks, Twitter has become one of the most widely used online social network for scientific analysis [5]. Most of the research in this field is based on data gathered from Twitter. Although research has been done based on other social networks and media like Facebook and YouTube, this work has not led to new insights, new approaches or novel contributions, as compared to Twitter.

From a developer perspective, Twitter is particularly suited for data mining because of three key attributes.

- Twitter's API is well designed and easy to access.
- Twitter data is in a convenient format for analysis.
- Twitter's terms of use for the data are relatively liberal. It is generally accepted that tweets are public and accessible to anyone, hence the asymmetric following model that allows access to any account without request for approval.

The potential of Twitter's self-organizing, ever-growing pool of data offers direct insight into trends and interests on both a personal and collective scale, but it has yet to fully capture the imagination of developers. And, in terms of the value that could come from data mining social media, Twitter is just the tip of the iceberg.

CHAPTER 3

REQUIREMENT ANALYSIS AND SPECIFICATIONS

3. Requirement analysis and Specification

3.1 Functional Requirements:

a. Authenticate User using his Twitter Account.

b. Successfully log into Application.

c. Display Search results by taking into consideration his tweets.

3.2 Non Functional Requirements:

a. The application developed should be able to display search results in a fast manner, there should be no delay.

b. The application developed should be able to handle large number of tweets.

c. The application should take into consideration his tweets as well as the tweets of people he follows.

CHAPTER 4
DESIGN

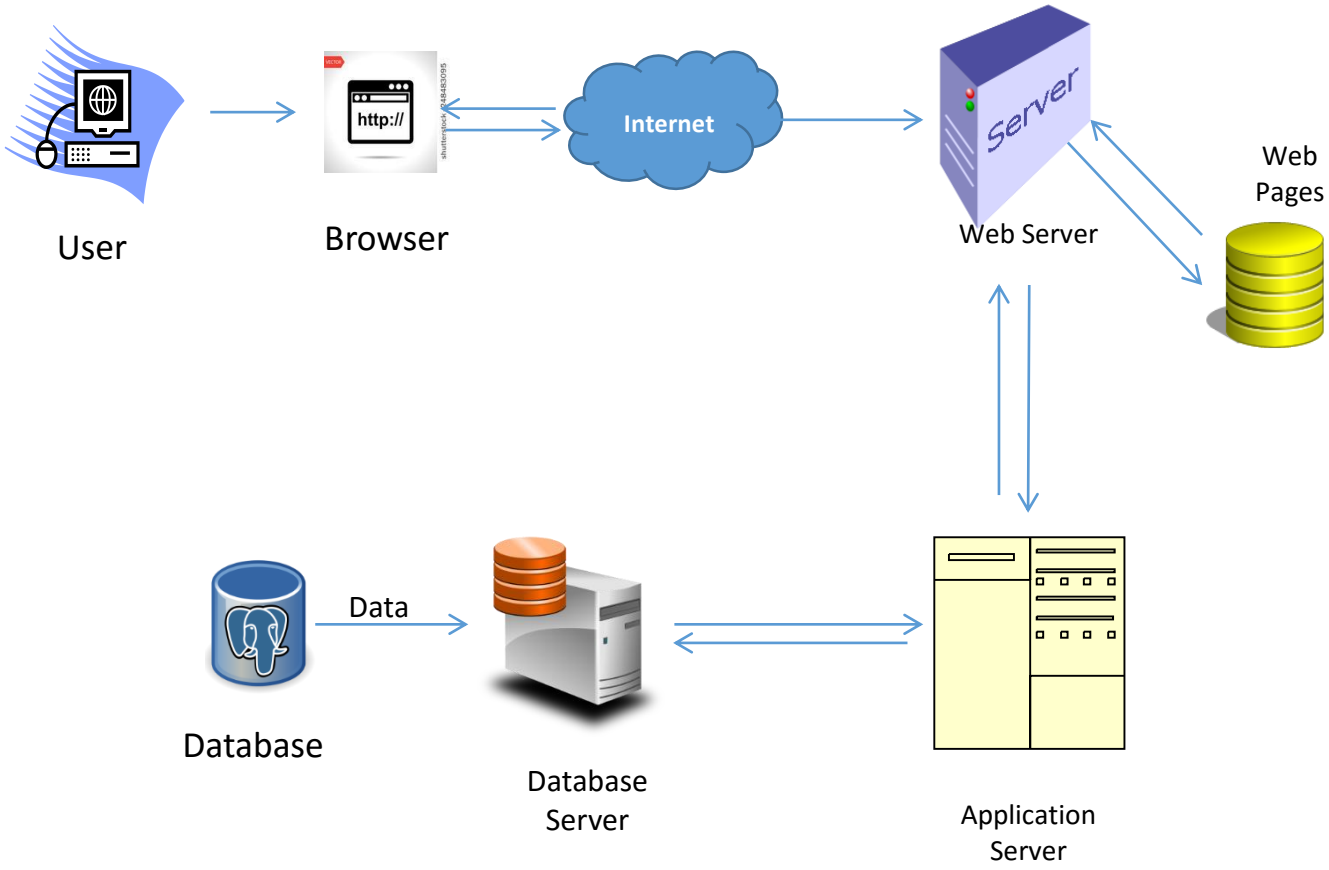
4. Design

4.1 High Level Design

4.1.1 Web Application

The architecture diagram provides an overview of the entire system identifying the main components and their interfaces. The following diagram represents the high level architecture for the web application:

Fig 4.1.1: High Level Architecture for web application



4.1.2 System Architecture

As illustrated by Fig. 4.1.2, the Personalised Search Engine will have 4 main components:

1. Authorisation module: Responsible for authenticating the user and authorising the web application to access his tweets.

2. Analysis module: Pre-processing of the tweets to eliminate unwanted information followed by extraction of entities and topics to create a dictionary.

3. Search module: Reads the search query from the user and performs Google search along with the associated keywords in the dictionary.

4. Search Results module: Extracts the links and descriptions from the results returned by Google API and reorders them to populate the results template.

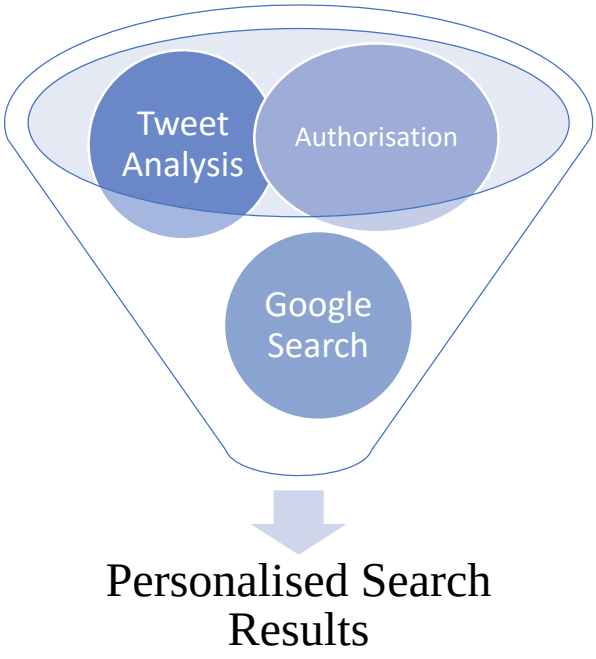


Fig 4.1.2: System Architecture

4.1.3 Use Case Diagram

Use case diagram depicts the interaction among the elements of the system. The Use Case is analyzed to identify, clarify and organize the system requirement.

There are two actors in our proposed system. They are:

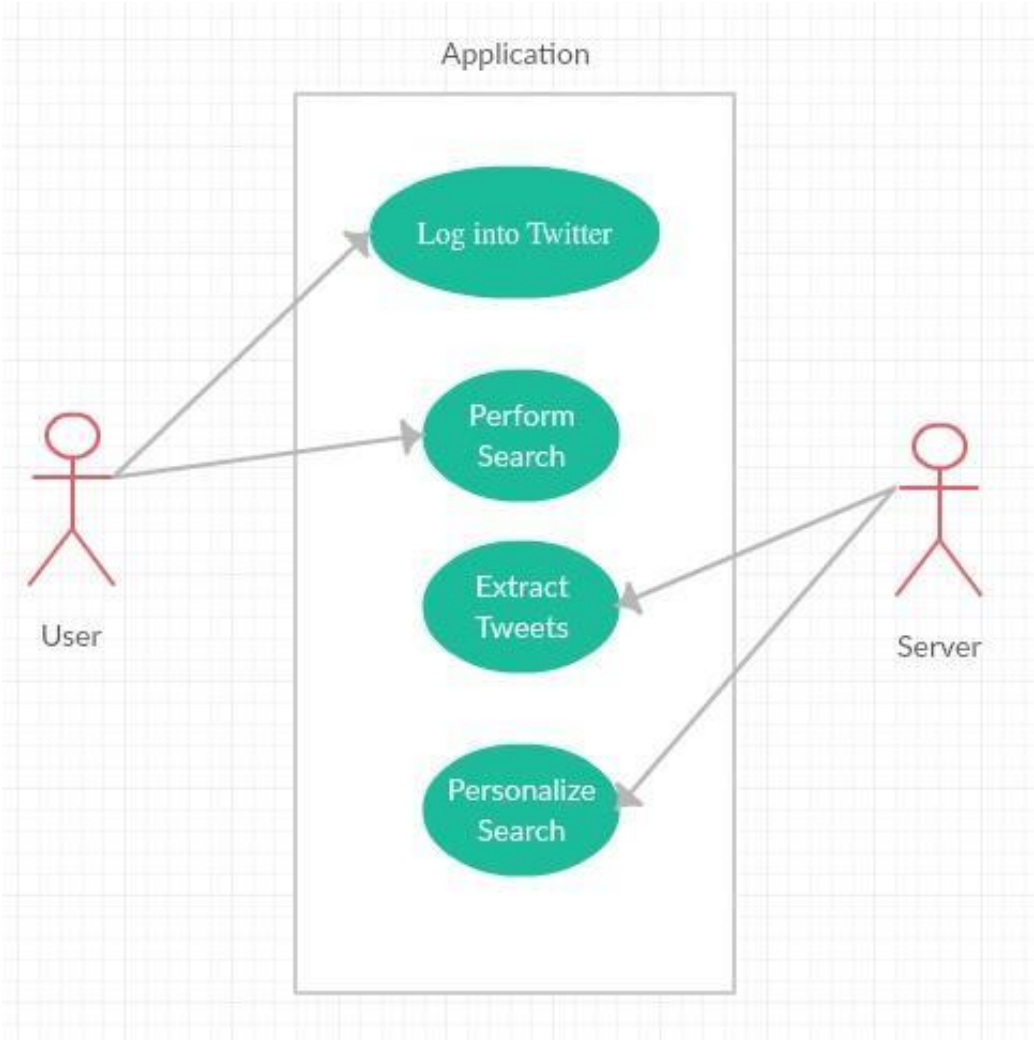
1. **User**

The functions of the user include logging into Twitter and performing search.

2. **Server**

   The server authenticates the user, extracts Tweets, personalizes searches and return html web pages.

Figure 4.1.3: Use Case Diagram



4.2 Low Level Design

4.2.1. Sequence Diagram

Let us say we want to create a new service that displays all tweets related to OAuth from a particular user's Twitter timeline, called OAuthTwitter or OAT. This sequence will consist of 5 players:

1. The user or **Resource Owner (RO)**.
2. Chrome, the user's browser or **User Agent (UA)**.
3. OAT which is the service of the web application that wants access tweets or the **Client**
4. Twitter, where users tweets are stored also called the **Resource Server (RS)**.
5. Twitter is also the **Authorization Server (AS)**.

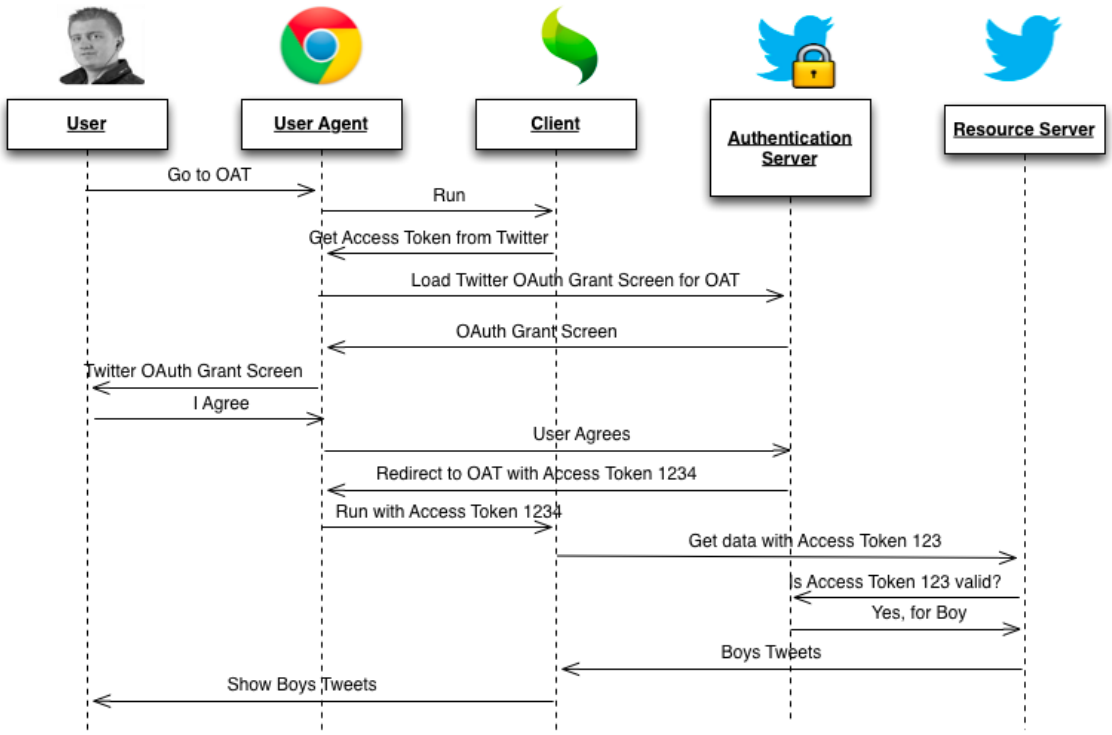


Fig 4.2.1: Sequence Diagram for Twitter User authentication using OAuth

Now when the user opens the url **https://oat.example.com**, Chrome (**UA**) loads the page and executes the python script. When the **Client** service sees that it doesn't have an **Access Token** to get the data from Twitter, it redirects Chrome (**UA**) to Twitter with a **Client ID and a redirect URL**. When the user grants OAT access to his data, Twitter (**AS**) redirects him back with: **https://oat.example.com/index.html#access_token=x**. The **Client** is now authorised and hence, makes an API call to **RS** for the data. The Twitter API (**RS**) checks for this token on in its authentication store (**AS**), sees that it's valid for the user and returns the data.

CHAPTER 5
IMPLEMENTATION

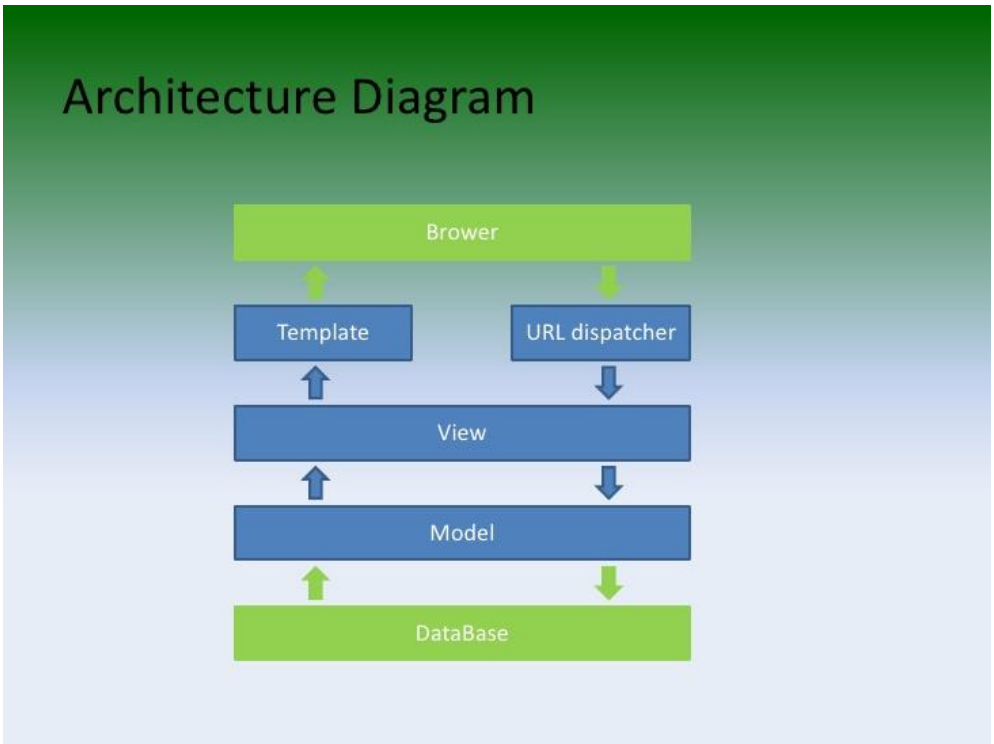
5. Implementation

5.1 Overview of Technologies Used:

5.1.1 Django-Python

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design [6]. Built by experienced developers, it takes care of much of the hassle of Web development. It's free and open source. Django is a "MTV" framework – that is, "model", "template", and "view." We use Django to build our Web application.

Fig 5.1: Architecture of Django



5.1.2. Twitter API

Rest API's: The REST APIs provides programmatic access to read and write Twitter data. This includes: author a new Tweet, read author profile and follower data. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

OAuth: Twitter uses OAuth to provide authorized access to its API and hence the developed web app makes use of OAuth to authenticate twitter users.[7]

Features of OAuth

1. Secure - Users are not required to share their passwords with 3rd party applications, increasing account security.
2. Standard - A wealth of client libraries and example code are compatible with Twitter's OAuth implementation.

To use OAuth, an application must:

1. Obtain access tokens to act on behalf of a user account.
2. Authorize all HTTP requests it sends to Twitter's APIs.

5.1.3. Text Razor

The TextRazor API helps extract and understand the Who, What, Why and How from tweets with unprecedented accuracy and speed [8]. It combines state-of-the-art natural language processing techniques with a comprehensive knowledgebase of real-life facts to help rapidly extract the value from documents, tweets or web pages. It provides Entity Extraction, Disambiguation and Linking, Keyphrase Extraction, Automatic Topic Tagging and Classification.

5.1.4. Google

Google API is used to perform a Google search of the search query given by the user [9]. This query is modified based on the associated values present in the dictionary created by the application. If such values do not exist, a normal Google search is performed on the query given by the user and the results are displayed without modification.

5.1.5. Beautiful Soup

Beautiful Soup is a Python library for pulling data out of HTML and XML files [10]. It is a HTML parser written in the Python programming language. The name is derived more from the unrelated 'tag soup', the unstructured and almost unparseable HTML found on most websites. Beautiful Soup doesn't fetch the web page, which must be done separately. We use Beautiful Soup to scrape the HTML page returned by Google in order to extract links, title and description.

5.2 Implementation Details of Modules

5.2.1. Authorization

This module consists of the user logging into our web app with his twitter credentials. Our Web app makes use of OAuth, redirects to Twitter Authentication Server to obtain an access token. After successful logging in, the application has access to the user's tweets as well as the tweets of people the user follows.

5.2.2 Analysis

The Analysis module, extracts the tweets using the Twitter API. The Twitter API returns the tweets in the form of a JSON, this JSON is parsed to get only the text part. Text pre-processing is carried out on the tweets. The processed text is then analysed using the Text Razor API. This API helps extract entities and topics. The topics are analysed again to get its entities. The result of the analysis is stored in the form of a Dictionary. The dictionary consists of a key and a list of values. The values are a list of entities that the key belongs to.

5.2.3 Search

The search module is a HTML page that reads the query entered in by the user and searches the query for keywords. These words are looked up in the dictionary and the list of values associated with that key are used to personalize the search and predict the result expected by the user based on his interests. In the case that keywords in the query are not found in the dictionary, the result displayed will be the same as Google Search Engine.

5.2.4 Search Results

The result is a search engine results page and thus consists of a title, link and descriptions of the hits matching the search. Beautiful Soup is used to parse the HTML page returned by Google API and get the necessary information. This information is populated in a template and displayed as the result.

5.3 Algorithm

5.3.1 Pseudo Code

The following is the pseudo code of the algorithm that is used to personalise the search results:

Input: *Twitter authentication details, search string*

Output: Personalized search results

Assumptions:

User is a frequent user of Twitter

1. Authenticate User as valid Twitter user by using OAuth API.
2. Authorize application to be able to read tweets.
3. Retrieve user tweets and timeline tweets by the use of Twitter API.
4. Store tweets as a JSON.
5. alltweets →Fetch "text" from JSON
6. response → process(alltweets) using TextRazor to retrieve list of all entities and topics.
7. Foreach entity in response.entities do:
    1. If entity is classified:
        1. If entity.confidence_score > 0.7:
            1. If entity already exists in dictionary:
                1. Increase frequency_score of entity
            2. Else:
                1. Add "type" of entity and entity.name to dictionary
                2. frequency_score(enitity) = 1
6. Foreach topic in response.topics do:

1. If topic is classified:
    1. If topic.confidence_score > 0.5:
        1. If topic does not exist in dictionary:
            1. Add “type” of entity and entity.name to dictionary
            2. frequency_score(enitity) = 1
7. Load home page of search engine with a search box.
8. search_string → query entered by the user.
9. response → process(search_string) using TextRazor to retrieve list of all entities and topics.
10. Foreach entity in response.entities do:
    1. If entity.type exists in dictionary:
        1. For values associated with entity,type do:
            1. search_entity → value with next highest ‘frequency_score’
            2. Results → Google(search_string + search_entity) and store top 3 results
11. Display search results in decreasing order of frequency_score

5.3.2 Code Snippets

1. Urls.py

```
from django.conf.urls import patterns, include, url

from django.contrib import admin

admin.autodiscover()


urlpatterns = patterns('',
   url(r'^admin/', include(admin.site.urls)),
   url('', include('social.apps.django_app.urls', namespace='social')),
   url(r'^$', 'django_social_app.views.login'),
```

```
    url(r'^home/$', 'django_social_app.views.home'),
    url(r'^logout/$', 'django_social_app.views.logout'),
    url(r'^results/$', 'django_social_app.views.results')
)
```

2. Results.html

```
<!DOCTYPE html>
<html >
 <head>
  <meta charset="UTF-8">
  <title>Personalized Search Engine</title>
  {% load static%}
    <link rel="stylesheet" href="{% static "css/results.css" %}" type="text/css"
   charset="utf-8">
  </head>
<body>
   <div>
        <h1>Results </h1>
       </br>
       {% for p in together %}
            <div>
            <ol>
                <li><span>{{ forloop.counter }}.</span>
              <p> Link: <a href= {{ p.0 }} > <b>{{ p.1 }}</b> </a> </br>{{
       p.2 }}  </p></li>
```

```
                    </ol>
                    </div>
                {% endfor %}
        </div>
</body>
</html>
```

3. OAuth Login using Twitter Account

```
def login(request):
   return render(request, 'login.html')

@login_required(login_url='/')

def home(request):
    consumer_key = 'RmYhC39yrTsRahPrxpbTNIk9m'
    consumer_secret
='SEjo3pHKsuK6jnXuLoOfX7cZMbRMAzRlfiiRs4anWirhYbnGbU'

    social = request.user.social_auth.get(provider='twitter')
    access_token = social.extra_data['access_token'].get('oauth_token')
    access_secret = social.extra_data['access_token'].get('oauth_token_secret')

    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)
 return render(request,'home.html')

def logout(request):
    auth_logout(request)
    return redirect('/')
```

4. Twitter API – Tweepy to extract Tweets

```
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth)
alltweets = ""
public_tweets = api.home_timeline(count=500)
for tweet in public_tweets:
        alltweets+=tweet.text
user_tweets = api.user_timeline(count=100)
for tweet in user_tweets:
        alltweets+=tweet.text
process_or_store(alltweets)
```

5. Text Razor- Categorization of Tweets

```
textrazor.api_key =
"813c3fc408c749a28006cca97f5865dca86c569f77ed08728b151bc0"
client = textrazor.TextRazor(extractors=["entities", "topics"])
alltweets = re.sub(r"(?:\@|https?\://)\S+", "", alltweets)
client.set_classifiers(["textrazor_iab"])
response = client.analyze(alltweets)
for entity in response.entities():
        if entity.confidence_score > 0.7:
                if entity.dbpedia_types:
                for e in entity.dbpedia_types:
                        if entity.id not in entities_dictionary[e]:
```

```
                                entities_dictionary[e].append(entity.id)
                        if entity.id in score:
                                score[entity.id]+= 1
                        else:
                                score[entity.id] = 1
                topics=""
        for topic in response.topics():
                topics+= topic.label + "\n"
        response = client.analyze(topics)
        for entity in response.entities():
                if entity.confidence_score > 0.7:
                if entity.dbpedia_types:
                for e in entity.dbpedia_types:
                        if entity.id not in entities_dictionary[e]:
                                entities_dictionary[e].append(entity.id)
                                score[entity.id] = 1
```

6. Google Search

```
def results(request):
    if 'search_string' in request.GET:
        search_string = request.GET['search_string']
     together = []
    appends=[value for key, value in entities_dictionary.items() if search_string ==
key.lower()]
    myscores={}
    sorted_appends=[]
    if appends:
```

```
        for e in appends[0]:
            myscores[e] = score[e]
        sorted_appends = [key for key, value in sorted(myscores.iteritems(),
key=lambda (k,v): (v,k), reverse=True)]
        for e in sorted_appends:
            r = google_search(search_string + " " + e)
            r1=[]
            if r[0][2]:
                r1.append(r[0])
            if r[1][2]:
                r1.append(r[1])
            if r[2][2]:
                r1.append(r[2])
            together.append(r1)
    else:
            together = google_search(search_string)
            return render(request, 'results1.html', {'together':together})
    return render(request, 'results.html', {'together':together})
```

7. Beautiful Soup

```
def google_search(term):
    br = mechanize.Browser()
    br.set_handle_robots(False)
    br.set_handle_referer(False)
    br.addheaders= [('User-agent','mychrome')]
    term=term.replace(" ", "+")
    query="http://www.google.com/search?num=5&q=" + term
    htmltext=br.open(query).read()
    soup=BeautifulSoup(htmltext,"lxml")
    search= soup.findAll('div',attrs={'id':'search'})
    searchtext=str(search[0])
    soup1=BeautifulSoup(searchtext,"lxml")
    list_items=soup1.findAll('div', attrs={'class':'g'})
```

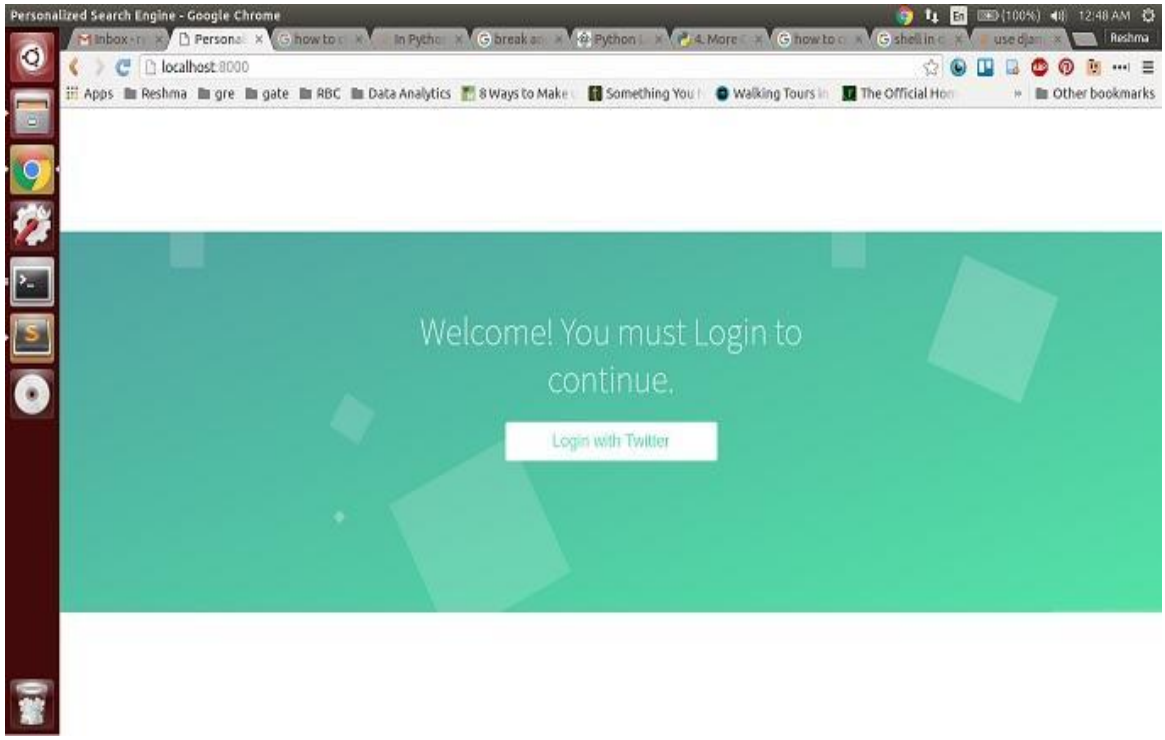
```
for div in list_items:
    soup2= BeautifulSoup(str(div),"lxml")
    links = soup2.find('cite')
    mylink = re.sub('<[^>]*>', '', str(links))
    if mylink is None:
        continue
    else:
        desc_items=soup2.find('span',attrs={'class':'st'})
        mydesc = re.sub('<[^>]*>', '', str(desc_items))
        title= soup2.find('h3',attrs={'class':'r'})
        mytitle= re.sub('<[^>]*>', '', str(title))
        val1 = mylink.startswith('http')
        val2 = mylink.startswith('https')
        if not val1 and  not val2:
            mylink = "http://"+mylink
        row=[]
        row.append(str(mylink))
        row.append(str(mytitle))
        row.append(mydesc)
       results.append(row)
return
```

CHAPTER 6
TESTING AND RESULTS

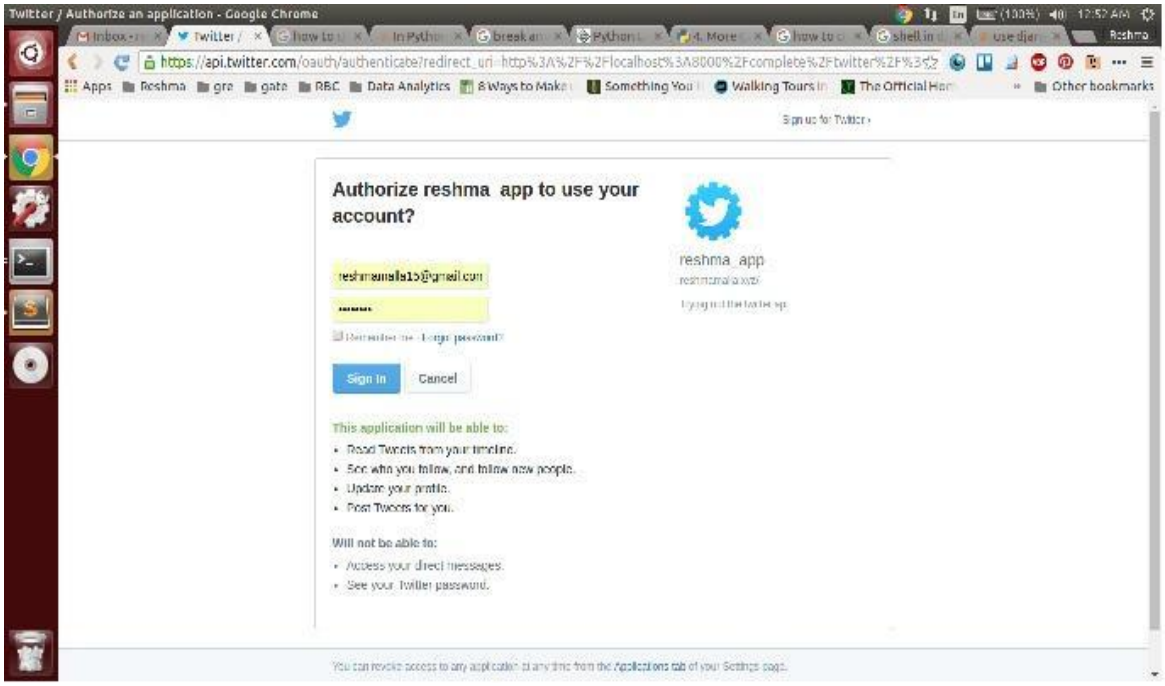
6. Testing and Results

6.1 Screenshots

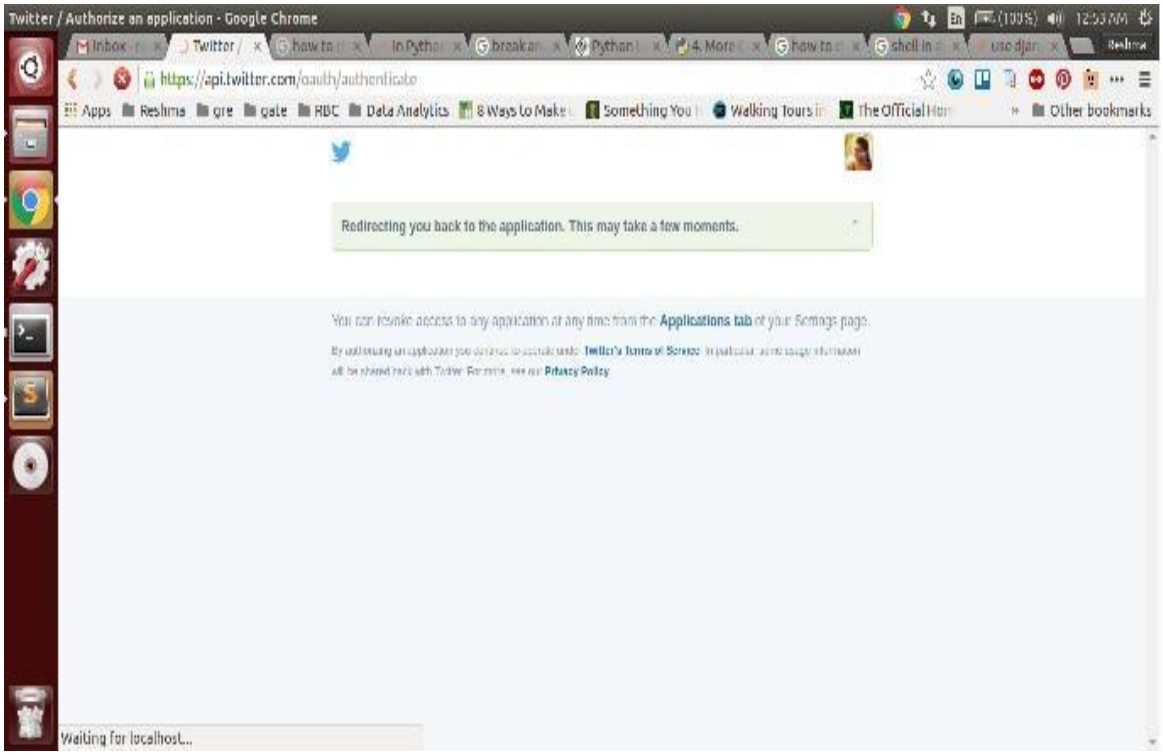
1. Login



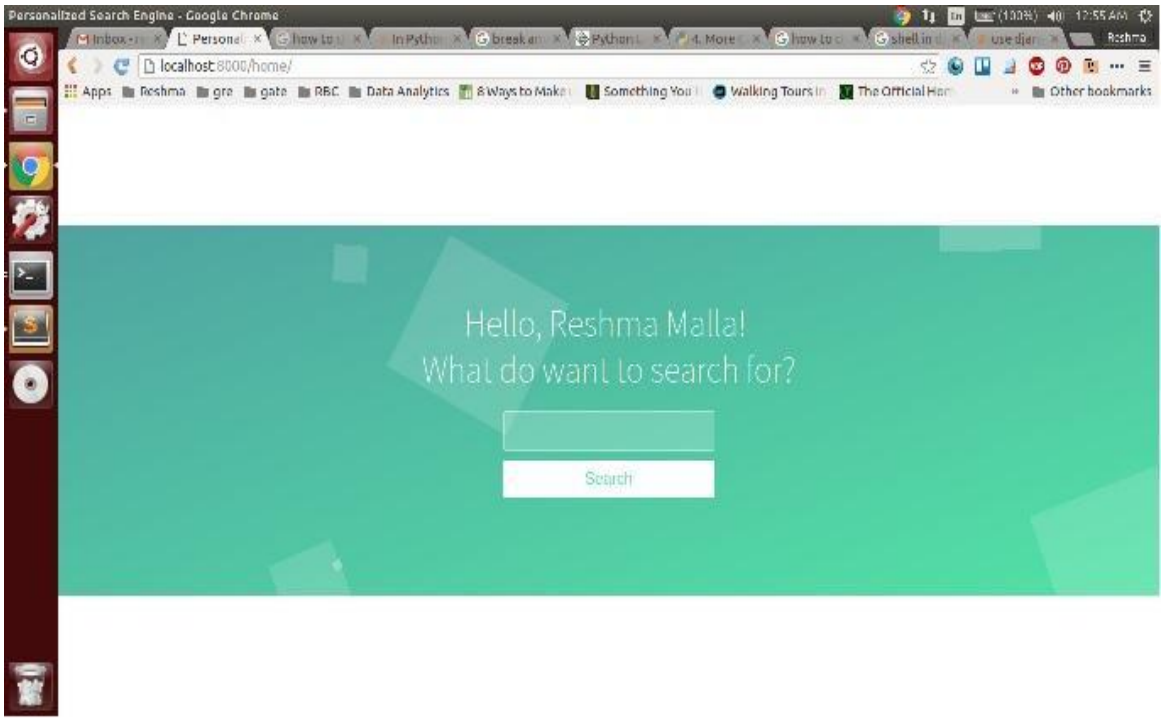
2. Authorize Application



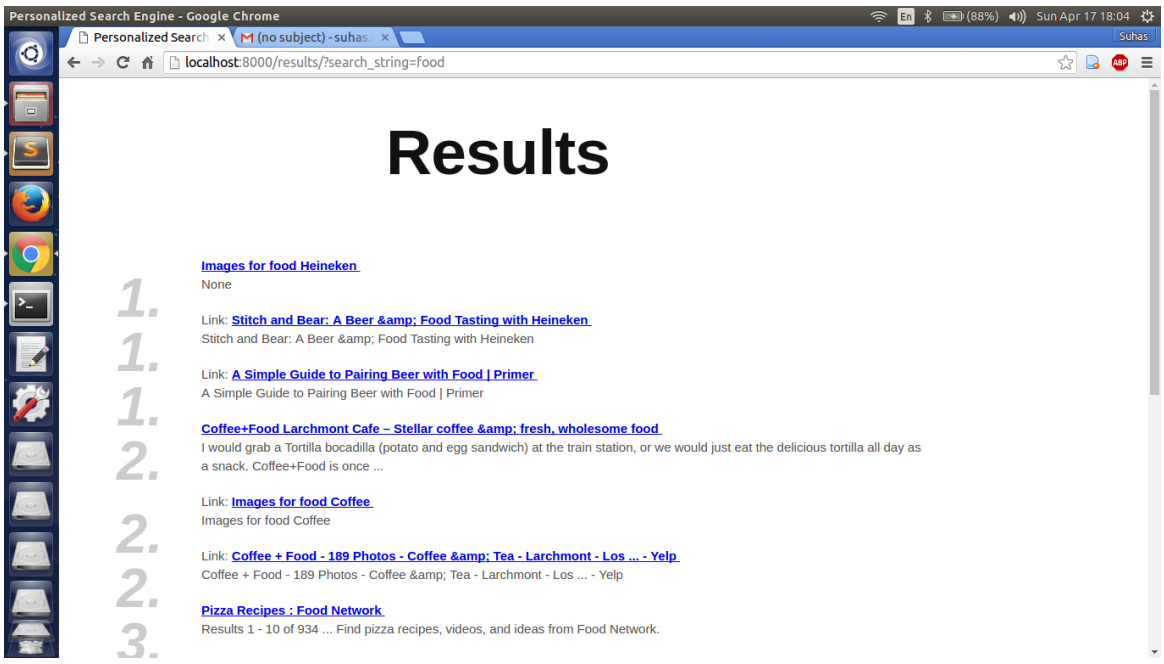
3. Redirect Back to application



4. Search Bar



5. Results


Personalized Search Engine - Google Chrome
Sun Apr 17 18:04
Personalized Search
(no subject) - suhas...
Suhas
localhost:8000/results/?search_string=food
Results
1. Images for food Heineken
None
1. Link: Stitch and Bear: A Beer & Food Tasting with Heineken
Stitch and Bear: A Beer & Food Tasting with Heineken
1. Link: A Simple Guide to Pairing Beer with Food | Primer
A Simple Guide to Pairing Beer with Food | Primer
2. Coffee+Food Larchmont Cafe – Stellar coffee & fresh, wholesome food
I would grab a Tortilla bocadilla (potato and egg sandwich) at the train station, or we would just eat the delicious tortilla all day as a snack. Coffee+Food is once ...
2. Link: Images for food Coffee
Images for food Coffee
2. Link: Coffee + Food - 189 Photos - Coffee & Tea - Larchmont - Los ... - Yelp
Coffee + Food - 189 Photos - Coffee & Tea - Larchmont - Los ... - Yelp
3. Pizza Recipes : Food Network
Results 1 - 10 of 934 ... Find pizza recipes, videos, and ideas from Food Network.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7. Conclusion and Future Enhancements

7.1 Conclusion

The developed web application makes use of Twitter OAuth for the authentication of Twitter users. The tweets of the user as well as the tweets of people followed by the user are extracted. Then tweets are then analysed using Text Razor API. Text Razor helps perform text preprocessing and a set of entities and topics are identified. The topics are analyzed further to determine their entities. For each entity, Text razor provides a list of domains it could belong to and a confidence score associated with it. This score is used to filter out the likely incorrect results.

Following the use of Text Razor, a dictionary is created to store the result of the analysis carried out. When a search is carried out, the dictionary is looked up for a matching key and the list of values for that key are associated with the carried out search. The google search returns a HTML page, Beautiful Soup is used to extract data from html tags and used in the construction of our results.

This way, the search is modified based on his tweets and interests. This occurs at a small expense of time and the search results are personalized to better suit his interests. The results are hence more apt.

7.2 Future Work

1. Other Social media data such as Facebook can be added to the analysis process to enhance the dictionary created.
2. Analysis of the tweets can be improved with better pre-processing and meticulous extraction of entities and topics.
3. Building a wider and comprehensive web of the user's interests will better personalise his/her search results.
4. The ordering system proposed in the algorithm can be improved to provide better organised results.
5. The application's user interface can be improved to make it more appealing.

REFERENCES

[1] Solihin, Niko, "Search Engine Optimization: A Survey of Current Best Practices" (2013). Technical Library. Paper 151. http://scholarworks.gvsu.edu/cistechlib/151

[2] "comScore Releases August 2011 U.S. Search Engine Rankings ..." 2012. 28 Apr. 2013, http://www.comscore.com/Insights/Press_Releases/2011/9/

[3] Brin, Sergey, and Lawrence Page. "The anatomy of a largescale hypertextual Web search engine." Computer networks and ISDN systems 30.1 (1998): 107-117.

[4] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon, "What is Twitter, a Social Network or a News Media?" International World Wide Web Conference Committee (IW3C2)

[5] MarketingGum, http://www.marketinggum.com/ twitter-statistics-2011-updated-stats/.

[6] Django Official Documentation, www.docs.djangoproject.com

[7] OAuth Authentication- Real Python, https://realpython.com/blog/python/adding-social-authentication-to-django/

[8] Text Razor Python API, www.textrazor.com/docs/python

[9] Google API, www.developers.google.com/web-search-docs/

[10] Beautiful Soup, www.en.wikipedia.org/wiki/Beautiful_Soup