Medi Crawl: A web search engine for diseases

1. Introduction

1.1 What is a Web Search Engine?

Internet is our everyday source for all kinds of information, like news, broadcast, music, pictures, or even movies. We are immersed in the flood of big data every day. However, the information we need is only a small portion of it and for different individuals the importance of information is hardly the same. A search engine will make it convenient for us to absorb information efficiently.

1.2 Crawling & Indexing

Search Engines have automated robots called crawlers that use links to scour the Internet, find web pages, and decipher page data that are indexed to be included in search engine results.

1.3 Ranking

Each search engine has different ranking factors, but they all have a few factors in common: keywords, content, and links. Keywords and content are arguably two of the most important factors that search engines look for when ranking pages. Because of this, it is imperative that you know what keywords are in highest demand within your market and incorporate those keywords into the content on your website. All your website's content will naturally create a collection of links; search engines use link analysis algorithms that look at the sources, number, and anchor texts of links to help determine their relevance in search queries. TF-IDF is one such method for ranking.

2. Scope of Work

2.1 Problem

Design a web search engine (Medi Crawl) for diseases.

2.2 Functionality

Try to beat the performance of FindZebra.
(ref.: https://www.findzebra.com/)
FindZebra: A search engine for rare diseases - https://arxiv.org/abs/1303.3229

2.3 Modules

1. Definition (Wikipedia and WebMD) - other names, types
2. Symptoms / Effects
3. Medicines / Vaccines (shopping?)
4. Treatment (hospitals / clinics?)
5. List of Doctors / researchers
6. News / Research / Books

2.4 Vision

We are going to use open-source libraries to build our web crawler to download the data from the internet, and we are using MongoDB to store the data, Pig to perform MapReduce (if required) on the database, and we are going to build a simple web interface as front-end layer for users to search text using Node & Express JS.

2.4.1 Accessibility & Indexing

For your website to be visible in search results, search engines must be able to find it. They do this by crawling the web and looking for relevant and indexable content, such as link structures and HTML features like alt tags for images.

2.4.2 Ranking factors

The different factors that we will be looking at are content to make sure that your content is relevant and utilizes keywords without being duplicated on multiple pages, keywords, and HTML markup to make sure that your source code is relevant to crawlers.

2.5 Benefits

2.5.1 Data Management

The data management plan to get the data for our prototype. We are planning to implement a search engine by building our own indexed database to collect webpages via Web crawling.

2.5.2 Query Implementation

- Query should be returned and displayed within 2s.
- Accuracy (confidence of results) should be more than 75%

3. Resources

OSS libraries + Free APIs: e.g. bootstrap, anaconda, beautiful soup, json, NumPy, sklearn, plotly etc.

3.1 Server

Express Server

3.2 Programming tools

Visual Code

3.3 Platform

MongoDB, NodeJS, MapReduce

4. Role of each member:

4.1 Devharsh Trivedi (dtrived5@stevens.edu) will be working on backend to build WebCrawler for the engine and indexing.

- Design you own Web "spider" to go through several levels deep starting from a specified page.
- Retrieved documents should be automatically indexed (inverted file format) and stored on a server.

4.2 Vaishnavi Gopalakrishnan (**vgopalak@stevens.edu**) will be working on MongoDB and Node JS to store and process the data.

- On the client-side, a simple query interface should provide the capability to perform simple Boolean queries.
- Results displayed should be ranked.

In terms of other work, like web interface, or some unpredictable small problems to fix, we will be working together.

5 Schedule:

Planning/Design:	Mar 15
Implement web crawler for diseases:	Apr 5
Use MongoDB to build our Database:	Apr 10
Implementing and Testing MapReduce on the Database:	Apr 20
Deal with the web interface & Ranking:	May 4
Performance and Accuracy improvements:	May 8
Documentation, Presentation & Project Report:	May 12