Search Engine Project

Name:	Section:	Bench Number:
Amr Magdy Abdelwahed	1	32
Fares Atef	1	35
Amr Elsheshtawy	1	33

DuckDuckGoose

Introduction:

A fully functional crawler-based search engine, which starts with a set of seed URLs, follows their links to other pages, indexing their content along the way which results in an index of web pages that could be ranked according to their relevance to a given user query. In this document we describe the crawling and the indexing process in more detail as well as the ranking.

Design

In this section we will break down the architecture of our search engine, the different algorithms we used as well as the user interface.

Architecture

Crawler

This represents the core part of a crawler-based search engine and its method of action is as follows:

1. Start with a set of seed URLs that represent the starting point of your crawl.
2. For every seed URL collect all the URLs going out from it, as they will also be crawled
3. Repeat this process until you've collected enough URLs to build a decent database.

Important notes about the crawler:

- The crawler is multithreaded so that every thread starts with its own seed, which significantly increases performance.

- Our search engine only supports English which means that any URL that refers to non-English content will be ignored.
- The crawler saves its state with every URL so that if it's terminated prematurely, it can continue from it left off
- The crawler has to keep in mind the Robots.txt policies.

Indexer

This is considered the second stage after the crawling process and it goes as follows:

1. Receive all the documents which have been crawled.
2. Process each document, this processing includes: removing html tags, removing special characters and stop words, converting all words to lowercase, tokenizing the document and finally stemming all the remaining words.
3. For every word in the processed document calculate the Term Frequency and divide it by the total length of the document to get the Normalized Term Frequency.
4. When all the processing is over, create an inverted file which contains every single word obtained from all the documents and the documents in which they appeared.
5. After getting the inverted file, calculate the Inverse Document Frequency (IDF) for every word.
6. Finally, store this inverted file in the database.

Notes about the Indexer:

- The Indexer is also multithreaded to allow for a fast inversion of files and a fast total indexing time.

Query Processor:

This part is responsible for normalizing the given user query so that it can use to retrieve results from the database, it works similarly to how the indexer processes the document in terms of removing stop words, converting to lowercase, tokenizing and stemming.

Ranker

This module has two parts:

Page Ranker

This is responsible for giving every document in the database a score based on its popularity through an algorithm used by Google.

Results Ranker

This part ensures that all the documents retrieved by the query processor is sorted in order of relevance to the query, how it works is as follows:

1. Get all the results from the query processor.
2. For every URL in the results perform a weighted sum of TF*IDF and PageRank score obtained from the Page Ranker.
3. Sort the Results in descending order according to the final score.

Notes about the Results Ranker:

- The final relevance score for each document is multiplied by a weighting factor depending on several factors, these factors include: 1-the position of the word in the

document whether it be title, header or body 2-whether or not the query word is in the URL of that document 3-whether the document has the exact query word or it has a word which shares a stem with the query word.

- The Ranker also ensures that the results do not contain any spam in case the document has a very high Term Frequency.

User Interface:

We have designed a user-friendly interface with a big smiling duck at the front using HTML and CSS.

Additional Sub-Modules:

Webpage Scraper:

This sub-module is responsible for bringing the paragraph in which most of the query words appeared and also the title of the document.

Notes about the Webpage Scraper:

This module is generally very resource expensive therefore we designed it to be multi-threaded so as to improve performance.

Additional features:

Voice Search:

We have added the feature of using voice instead of typing to interact with the search engine.

I am feeling lucky:

We have added Google's "I am feeling lucky" search option which retrieves a result that the user might like without having to look through the results