Assignment 1

Miranda Smith

September 21, 2017

1 Question 1.1

Think up and write down a small number of queries for a web search engine. Make sure that the queries vary in length (i.e., they are not all one word). Try to specify exactly what information you are looking for in some of the queries. Run these queries on two commercial web search engines and compare the top 10 results for each query bydoing relevance judgments. Write a report that answers at least the following questions: What is the precision of the results? What is the overlap between the results for the two search engines? Is one search engine clearly better than the other? If so, by how much? How do short queries perform compared to long queries?

Answer

I created four search queries. "How long is a male asian water monitor" "What color should I paint my living room" "odu dragas hall location" "IT" and compared the first 10 results from Yahoo and Bing.

Query	Yahoo Precision	Bing Precision	Overlap
How long is a male asian water monitor	.70	.60	.30
What color should I paint my living room	.60	.90	.50
odu dragas hall location	.40	.40	.70
IT	.40	.40	.80

It does not appear that Bing or Yahoo is clearly better than the other. The only large difference occurs in "What color should I paint my living room" which bing got .90 revelent results while yahoo got .60. However it is a highly opinion based query, so it may just do better for that type of query. It appears that longer queries do much better than short queries. The two long queries got .60 and above for both search engines while the two short queries all got .40 precision.

Screenshots of the query results: Water Monitor Yahoo Water Monitor Bing Living Room Color Yahoo Living Room Color Bing Dragas Location Yahoo Dragas Location Bing IT Yahoo IT Bing

2 Question 1.2

Site search is another common application of search engines. In this case, search is restricted to the web pages at a given website. Compare site search to web-search, vertical search, and enterprise search.

Answer

Web Search's difference from site search is in the search domain. Site search is only searching the pages on one website while web search searches through all the pages on the internet that are archived by the particular search engine being used.

Vertical Search is searching a sub domain of the web that only relates to a specific topic. Site search is only restricted to one website, which may all be focused on a specific topic but that is determined by the site creators.

Enterprise search is searching only through files on a corporate intranet. While site search is restricted to searching a particular web site.

3 Question 1.4

List five web services or sites that you use that appear to use search, not including web search engines. Describe the role of search for that service. Also describe whether the search is based on a database or grep style of matching, or if the search is using some type of ranking.

Answer

Facebook.com Search on facebook is used to find people, photos, events, pages, marketplace listings, posts from friends, groups, an public posts that are on Facebook. The search is using a ranking system. It is separating out each category and the top results are always people and friends if possible.

Imgur.com Search on Imgur is used to find posts and users. It is a database based search, everything that matches the word will be selected. However, it sorts those results based on the post's points, relevancy, or recency based on the user's selection.

Amazon.com Search on Amazon is to help users find products based on the query. Amazon uses a database based search, it should find every item that would be relevant.

Spotify Search on spotify is meant to help users find songs, artists, and playlists. Spotify uses a ranked based search.It will return playlists that don't have the word being searched on any of the songs. It also categorizes the results between songs, artists, and playlists always keeping artists as the first result section.

Allrecipes.com Search on allrecipes is to help users find reciepes. Users are also able to narrow searches based off including and excluding certain ingredients.Allrecipes uses a grep based search.

4 Question 3.8

Suppose that, in an effort to crawl web pages faster, you set up two crawling machines with different starting seed URLs. Is this an effective strategy for distributed crawling? why or why not?

Answer

This is not an effective strategy for distributed crawling. The two computers aren't communicating. This means that eventually, even with different seed URLs, they would start overlapping pages they have crawled. They would need some way to separate out which machine should crawl which domains and transfer URLs between them.

5 Question 3.9

Write a simple single-threaded web crawler. Starting from a single input URL (perhaps a professors web page), the crawler should download a page and then wait at least five seconds before downloading the next page. Your program should find other pages to crawl by parsing link tags found in previously crawled documents.

Answer

```
#Modified from
#Stephen http://www.netinstructions.com/how-to-make-a-web-crawler-in-under-50-li
from html.parser import HTMLParser
from urllib.request import urlopen
from urllib import parse
import shutil
import time


class LinkParser(HTMLParser):

    # This is a function that HTMLParser normally has
    # but we are adding some functionality to it
    def handle_starttag(self, tag, attrs):
        if tag == 'a':
            for (key, value) in attrs:
                if key == 'href':
                    newUrl = parse.urljoin(self.baseUrl, value)
                    # add it to our colection of links:
                    self.links = self.links + [newUrl]
```

```
    def getLinks(self, url, index):
        self.links = []
        # Remember the base URL which will be important when creating
        # absolute URLs
        self.baseUrl = url
        # Use the urlopen function from the standard Python 3 library
        response = urlopen(url)
        # Make sure that we are looking at HTML and not other things that
        # are floating around on the internet (such as
        # JavaScript files, CSS, or .PDFs for example)
        if response.getheader('Content-Type')=='text/html':
            htmlBytes = response.read()
            filename = str(index) + "test.html"
            file_ = open(filename, 'wb')
            file_.write(htmlBytes)
            file_.close()
            htmlString = htmlBytes.decode("utf-8")
            self.feed(htmlString)
            return htmlString, self.links
        else:
            return "",[]


def spider(url, maxPages):
    pagesToVisit = [url]
    numberVisited = 0
    # The main loop. Create a LinkParser and get all the links on the page.
    # Also search the page for the word or string
    # In our getLinks function we return the web page
    # (this is useful for searching for the word)
    # and we return a set of links from that web page
    # (this is useful for where to go next)
    while numberVisited < maxPages and pagesToVisit != []:
        numberVisited = numberVisited +1
        # Start from the beginning of our collection of pages to visit:
        url = pagesToVisit[0]
        pagesToVisit = pagesToVisit[1:]
        try:
            print(numberVisited, "Visiting:", url)
            parser = LinkParser()
            data, links = parser.getLinks(url, numberVisited)
            time.sleep(5)
            # Add the pages that we visited to the end of our collection
            # of pages to visit:
            pagesToVisit = pagesToVisit + links
            print("**Success!**")
```

```
        except:
            print("**Failed!**")

spider("http://www.cs.odu.edu/~mln/", 10)
```

This program takes the supplied URL and a number for a max number of pages to collect. For a simple web crawler I didn't want it collecting a hundred pages of html that I wouldn't be using.It downloads and saves the page then finds all the links on the page. These collected links are then downloaded themselves, with a five second wait.