Web Search Engines — Homework 1

Anirudhan J Rajagopalan
N18824115
ajr619

Feburary 19, 2016

1 Problem 1

1.1 Optimality of processing postings by increasing size order

Processing the posting lists by the increasing order of the sizes of the posting list is infact optimal. A conjuctive query is made of AND clauses joined together. For example "abracadabra AND nyu" is a conjuctive query. Lets assume that NYU has more postings than abracadabra. Now if we process by the increasing order of posting size, we should fetch all the postings for abracadabra and the fetch all the postings for nyu. Since a conjuctive query returns all documents that contains both the terms, **the total result size should be lesser than or equal to the size of the smallest postings**.

If we assume nyu to have 1M postings and abracadabra to have 100 postings, by following the INTERSECT algorithm, we take a total of $O(1M + 100)$ for searching. But since we only keep the smaller list in memory, processing from smallest to largest lets us use less RAM.

1.2 Hashtable to speed up INTERSECT

Algorithm 1 INTERSECT algorithm using hashing to improve runtime

```
1:  function INTERSECT(t_1, ..., t_n)
2:      terms ←SORTBYINCREASINGFREQUENCY(< t_1, ..., t_n >)
3:      result ← postings(first(terms))
4:      terms ← rest(terms)
5:      for all term in terms do
6:          for all docId in result do
7:              if Hash<docID,term> == 0 then
8:                  remove(result, docId)
9:                  continue
10:     return result
```

The worst case time complexity of the above method will be that of the length of the shortest list times the number of terms. Since we order the terms by increasing order of freqency, this will lead to trimming of the results very fast.

2 Problem 2

2.1 Search Experiments

A number of experiments were carried out with the given query. This includes adding the international code, removing punctuation, removing part of the punctuation, removing the area code and so on. A sample of the observations is as given below:

2129983123 Home page does't show up

212-998-3123 Home page shows up as 4th result

+1-212-998-3123 A mention of Professor Ernest Davis as the 9th result at academia.edu website.

998-3123 No result for Ernest Davis or NYU

The results vary a lot when I search after signing off from my google account.

2.2 Difficulties with tokenizing worldwide telephone numbers

A search engine faces a difficult task of tokenizing worldwide telephone numbers. Phone numbers can be represented in a variety of ways, just as the way we used for searching above. A few issues with identifying telephone numbers are

1. Inconsistent use of punctuation such as brackets, hyphens, periods, and spaces.
2. Inconsistent use of international codes/area codes and other information that a human can generally assume.

Eventhough we face several issue while identifying the phone numbers we can use the other information available in the page to make a guess of the right format and the right phone number. Some of the visual clues are

1. HTML markups can provide clues as to which numbers are telephone numbers. Such as tags.
2. Locality of Phone numbers. Phone numbers are generlly located along side address. By identifying address or part of address we can generally infer phone numbers
3. The format of phone numbers can be domain/location specific. For example the ways of representing a phone number in India will differ from that in USA. But a particular format can generally be well used in a country. By identifying the language or the domain location, we can infer and tokenize phone numbers better.

The above few points summarize the difficulties in infering and tokenizing phone numbers. If we dont infer them properly we might end up tokenizing phone numbers in a different manner. Once we identify a phone number it is better to index them in a standard querable format so that we can convert them to whatever format necessary when we give mined information.

3 Problem 3

3.1 Reasonable Objective function

From the CMS book, we know that the probability of a web page getting changed follows a Poisson distribution. Hence we got the formula for age as:

$$Age(\lambda, t) = \int_0^t \lambda e^{-\lambda x}(t - x)dx \tag{1}$$

Equation (1) gives a reasonable value for the age of a particular web page. The second derivative of this expression gives the correct value for the cost of not crawling a web page. Similarly, we can get an expression of the age of a web page with respect to the amount of change $\Delta\sigma$. The rate of change ($\Delta\sigma$) is assumed to be constant for a particular web page. The random variable in the above case is the expression (t-x) which gives an approximate value of age. Similarly we can represent by $\Delta\sigma$ the random variable of amount of change of a website w.r.t time. Then the amount of change can be given as

$$Change(\lambda, \sigma, t) = \int_0^t \lambda e^{-\lambda x}\Delta\sigma dx \tag{2}$$

The first order derivative of the above equation is:

$$\frac{d}{dx}\int_0^t \lambda e^{-\lambda x}\Delta\sigma dx = \lambda e^{-\lambda x}\Delta\sigma \tag{3}$$

Minimizing the equation(3) w.r.t time gives us the optimal frequency with which we have to crawl a particular web site.

3.2 Information for better crawl performance

The crawler should collect the following information while crawling a webpage

1. The Δ difference between the sizes of the web page between the last crawl and current crawl.
2. The average refresh frequency of the web page
3. Stastic such as whether the webpage is a portal with multiple links to other websites.

These statistics help us figure out how frequently we should update our index for the webpage and the optimal crawling strategy. More importantly it helps us figure out all the variables required to minimize equation(3).

3.3 Missing Pages

If we dont crawl a web page, there is no way to know whether a new link to page B has been added. We can find if a page has changed by using the HEAD request which gives us the last modified time for a page and then use the response to decide on a full crawl. This will help us to crawl only when a page has changed.