Social Network Project

Francesco Siani, Marco Mecchia

Università degli Studi di Salerno

September 11, 2016

Overview

Introduction

The purpose of our work is to test different algorithms in the three fundamental areas for assembling any search engine offering a Sponsored Search system:

Ranking of web documents

Matching of words inside documents

Auctions for acquiring advertisement slots.

Web Model

For our porpuses, we model the web as a graph $G = (V, E)$, where:

- V is the set of web pages.
- E are the hyperlinks connecting the pages.

The graph is directed since hyperlinks are unidirectional.

Web Transition Matrix

Graph can be easily rappresented as matrices.

One rappresentation is the *Transition Matrix*:

$$T(p,q) = \begin{cases} 0 & \text{if } (q,p) \notin E \\ 1/\omega(q) & \text{if } (q,p) \in E \end{cases}$$

Where $\omega(q)$ is the outdegree of node q.

PageRank

PageRank is a well known algorithm that uses link information to assign global importance scores to all pages on the web.

- The intuition behind PageRank is that a webpage is important if several other pages point to it.
- PageRank is based on *Mutual Reinforcement* between pages.

The PageRank score $r(p)$ of a page p is defined as:

$$r(p) = \alpha \cdot \sum_{q:(q,p)\in E} \frac{r(q)}{\omega(q)} + (1-\alpha) \cdot \frac{1}{n}$$

Generalized Pagerank

Given the transition matrix of a graph T, PageRank can be easily generalized in a matricial form.

$$\boldsymbol{r} = \alpha \cdot \boldsymbol{T} \cdot r + (1 - \alpha) \cdot \boldsymbol{d}$$

Where $\boldsymbol{d}$ is a *static score distribution vector* of arbitrary, non-negative entries summing up to one.

Topic-Sensitive Pagerank

- Since its introduction, a great number of improvements have been tought for PageRank.
- One of the main ideas is to use more than one vector of PageRank based on the topic in which the user is interested.
- In fact, one can use the *static score distribution vector* seen in previous slide and the initial ranks to weight some pages more than others.
- Motivations: different people have different interests, biased random walks.

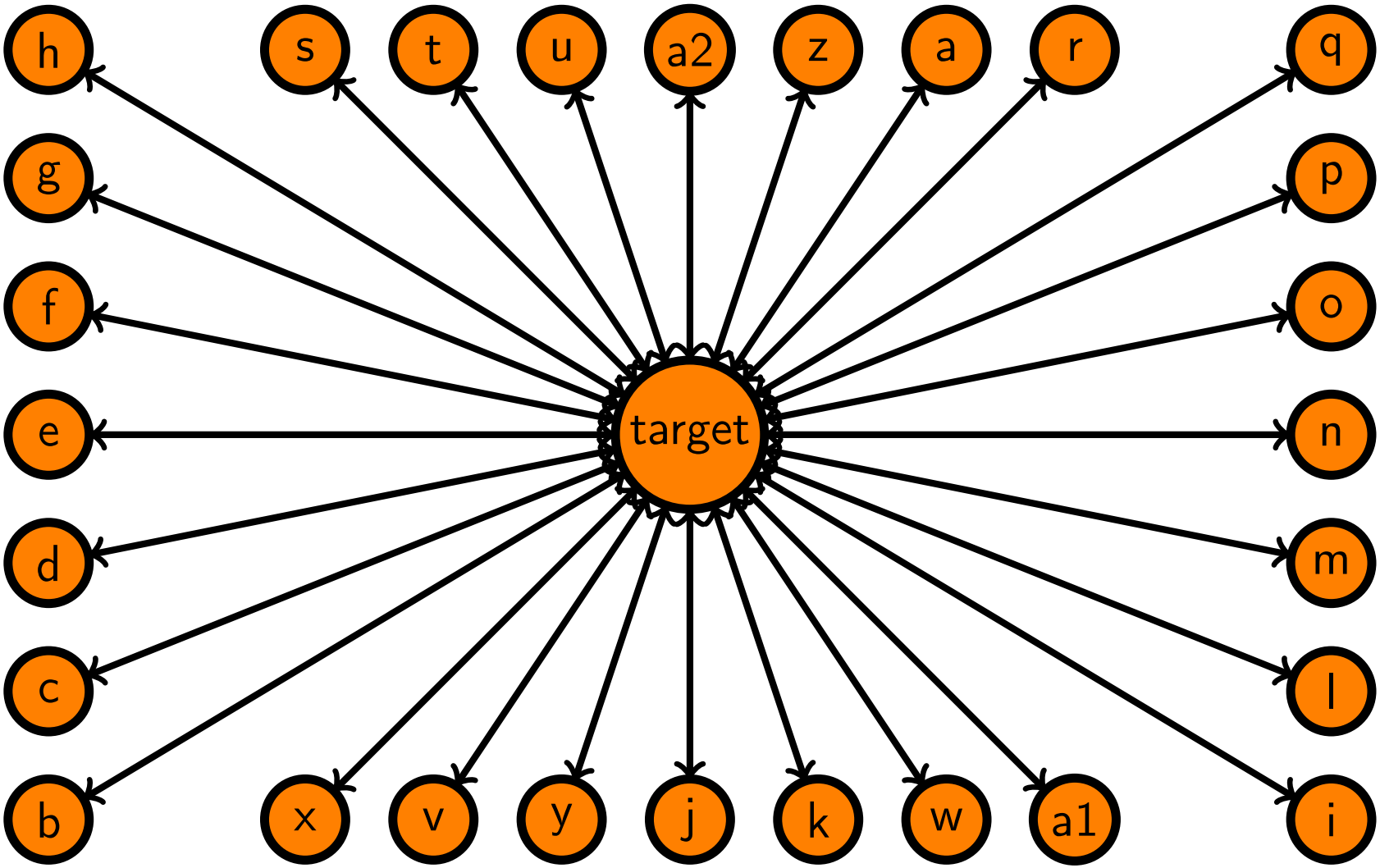
The spam farm problem

One of the main ideas to fool pagerank is the spam farm, which is made up by:

- A target page whose contents are likely to be matched by the search engine.
- A big set of spam pages whose contents are uninfluent, but they are two-ways linked to the target page.

With such architecture, PageRank gives the target page a very high score.

An example of spam farm



TrustRank: Overview

TrustRank is a particular form of topic-sensitive Pagerank.

- In this case the topic is a set of pages believed to be trustworthy.
- Useful to combat spam farms.

Intuition: is very unlikely that trusted pages point to spam.

TrustRank: Algorithm

Algorithm 1 TrustRank

Input: $\boldsymbol{T}, N, L, \alpha_\beta, M_B$
Output: $\boldsymbol{t}^*$

$\boldsymbol{s} =$ SelectSeed(...)
$\boldsymbol{\sigma} =$ Rank($\{1 \dots N\}$, $\boldsymbol{s}$)
$\boldsymbol{d} = \mathbf{0}_N$
for $i = 1$ to L **do**
    if $O(\sigma(i)) == 1$ **then**
        $\boldsymbol{d}(\sigma(i)) = 1$
    end if
end for
$\boldsymbol{d} = \boldsymbol{d}/|\boldsymbol{d}|$
$\boldsymbol{t}^* = \boldsymbol{d}$
for $i = 1$ to M_B **do**
    $\boldsymbol{t}^* = \alpha_\beta \cdot \boldsymbol{T} \cdot \boldsymbol{t}^* + (1 - \alpha_b) \cdot \boldsymbol{d}$
end for

Methods to selects seeds

There are two main ways to implement the function SelectSeed which returns the set $\boldsymbol{s}$ on which invoke the oracle:

1. Nodes with high pagerank score.
   In this way, if the target of a spam farm has a high pagerank is judged by the oracle.
2. Nodes with high inverse pagerank score.
   In this way, we use the oracle on nodes that have a lot of outgoing links.

Our implementation uses a hybrid tecnicque.

SpamMass

SpamMass is a useful tecnique to identify probably spam pages.

- It is based both on PageRanks and TrustRanks.
- Once the probably spam nodes are identified, they can be ignored by the search engine.

Given t_r and p_r the TrustRank and PageRank of the page r, the SpamMass s_r is given by:

$$s_r = \frac{p_r - t_r}{p_r}$$

The idea of Best Match

Given a query q, containing n query words, and a set of documents S, we define Best Match as a method that finds a subset of document S' such that:

- each document s_i of S' has a "reasonable" number of query words in it

According to this definition the basic Best Match consists of:

- counting how many query words documents have
  - we call this value "score" of a document
  - it is at maximum n
- ordering in decreasing order of score the documents (optional)
- return all documents whose score is "reasonable"
  - we use a threshold to define what is "reasonable"

Refining the Best Match

Two are the basic refinements to have a more efficient Best Match algorithm:

1. Use an inverted index.
   - in the form (word -> list of documents containing the word).
   - the keys of the dataset are the query words.
   - we can have in O(1) all the documents containing a determined word.
2. Use the frequency of the word in a document instead of assigning score 1.
   - defined as number of occurrences in document d / length(d).
   - it represents the *relevance* of a word in a particular document.

The main drawbacks of this refinements are the need of precomputation and special data structures.

Improved Best Match

We implemented also the following improved version of Best Match:

1. Sort offline documents in each inverted index in order of frequency of the term at which the inverted index refers.
2. For every query term define its possible impact on the score as the frequency of the most frequent document in its index
3. Sort the query terms in decreasing order of impact.
4. Consider the first 20 documents in the index of the first query term (if the first query term has an index with less than 20 documents, then complete with the first documents in the index of the next query term).

Improved Best Match

5. Compute the score for each of these documents.
6. Consider the first term in which there are documents that have not been scored.
7. Consider the first non-scored document in the index of this term.
8. If the frequency of the current term in the current document plus the sum of the impact of next terms is larger than the score of the 20-th scored document, then score this document and repeat from 7, otherwise consider the next.

Creating the Dataset

Our experiments ran on a set of approximatively 30000 pages created this way:

- We choosed a web-page for each of the 15 categories listed in https://www.dmoz.org/
- For every of these web pages we crawled 2000 pages by using the Wibbi online crawler
- Once generated the graph, from each pair of sets of 2000 pages, we choosed at random 10 pairs of vertices (u, v) with u being a page in the first set and v being a page in the second set and we added a link from u to v (if this link was absent).

Creating the Dataset

Here is the complete list of the websites chosen.

Category	Website	Description
Arts	www.imdb.com	The Internet Movie Database
Business	www.moodys.com	Corporate finance, banking
Computers	www.ibm.com	International Business Machines Corporation.
Games	www.ign.com	Videogame news
Health	www.who.int	World Health Organization
Home	www.allrecipies.com	Recipe search
Kids	www.cartoonnetwork.com	The home of cartoons online

Creating the Dataset

Category	Website	Description
News	www.foxnews.com	Breaking News
Recreation	www.lego.com	Producer of bilding blocks.
Reference	www.stackoverflow.com	Q & A site for computer science topics.
Science	www.researchgate.net	Researchgate is a network dedicated to science and research.
Shopping	www.ebay.com	One of the most known shopping website
Sports	www.nba.com	The official site of the National Basketball Association
Regional	www.lonelyplanet.com	Offers travel advice, detailed maps, travel news

Adding the spam farm

To test the utility of TrustRank and Spammass, we added to our dataset a spam farm in the following way:

1. We created a target page and 100 supporting pages.
2. The target page contains the 500 words with the smaller index among the one contained in the above 30000 nodes.
3. We created 30 random links from the original 30000 nodes to the target pages.

PageRank distribution

PageRank performance

TrustRank distribution

TrustRank performance

Experiment configuration

For obtaining the time comparison between BestMatch and ImprovedBestMatch we run both algorithms on 25 random queries of different length :

- 5 for each length from 3 to 7

Then we mean the results and plotted them on a graph

Time comparison BestMatch vs ImprovedBestMatch

Time comparison BestMatch vs ImprovedBestMatch

First of all Improved Best Match "wins" on all query lengths, moreover it seems to be constant. This is due to the fact that there is a huge difference on the times of the algorithm at least of 1 order of magnitude.

We can notice that times of query of length 5 in best match are smaller of the one on length 4; this can be attributable to the random generation of queries, in fact, if we take 5 very uncommon words there are few documents that have one of these word so the algorithm ends quickly.

Improved Best Match time

Search engine accuracy

Thank you for the attention.