(Web) Link Analysis

COSC–254 — April 10–??, 2019

Outline

The Web: data and their use

Crawling

Indexing

Ranking

HITS

PageRank

Data on the Web

The success of the Web is a *rare* phenomenon in *human history*

Its nature makes it a rich treasure trove of *diverse data types*:

- Web *content*: documents and links between them
- Web *usage*: patterns of user activities:
  - transactions, ratings, and user feedback
  - logs of user browsing behavior

Applications using web data

Content-centric:

DM apps: web pages as documents (*text analysis*, classification, *summarization*, …)

Discovery and Web *crawling*: how to find (new) webpages.

Search: find the pages that are *most relevant* to a query.

Linkage analysis: study the structure of the Web as a graph.

Usage-centric:

Recommender systems: "users who bought this product also bought …", friends recommendations, …

Web log analysis: model "normal" patterns to fight click-spam, optimize website design, …

Many of these applications use approaches we have studied

Outline

✔ The Web: types of data and their use

Crawling

Indexing

Ranking

HITS

PageRank

How to build a web search engine

1. Crawling: Find new webpages (or *new copies* of old ones), and *download* them

2. *Indexing*: *Parse* the webpages and *create a map* from words to webpages

3. Ranking: For *each possible query*, *compute the relevance* of each webpage to the query

4. Wait for users to leave Google and DuckDuckGo (What is Bing?)

Crawlers

Crawlers: programs that *find and download* webpages

A.k.a. spiders or robots

Must *download and store* the pages to process them as a set

Frequently, must:

refresh the set of downloaded pages;

delete the ones that do not exist anymore;

find new pages.

Building a simple crawler

Basic idea:

Start from a *seed set* S of pages;

Until S is not empty:

Choose a page p from S (*remove* p from S), and download p

Parse p to *find* pages that p links to, and *add* them to S

Aspects to decide:

Which page p to choose from S?

How to avoid fetching the same pages over and over?

When to stop? How to ensure that we fetch all web pages?

BasicCrawler

INPUT: seed *URLs* set S, selection algorithm $\mathcal{A}$

FrontierList $\leftarrow S$

repeat

Use algorithm $\mathcal{A}$ to select URL $X \in$ FrontierList

FrontierList $\leftarrow$ FrontierList $\setminus \{X\}$

Download URL X and save the corresponding page p_X

Parse p_X and add the URLs in it to the end of FrontierList

until termination criterion;

Filling the gaps

Which page p to *choose* from Frontier?

If $\mathcal{A}$ returns the *1st* element in FrontierList, then we are doing…
BFS from all sources in S at the same time.

Other possibilities: Give *higher* priority to

URLs of *frequently-changing* pages;

URLs with *high PageRank*;

URLs "*in topic*" (keywords in the page from where we extracted the URL);
…A *weighted combination* of the above is used in practice.

✔ Which URL to fetch next?

How to avoid fetching the same pages over and over?

When to stop? How to ensure that we fetch all web pages?

How to avoid fetching the *same* pages over and over?

What is the cause of fetching the same page multiple times?

Multiple pages link to it

A *non-satisfactory* "solution": keep a *hash table* H of URLs already fetched.

Add a URL to FrontierList only if it is not in H, and then add the URL to H.

How to avoid fetching the *same* pages over and over?

The hash-table approach is not entirely satisfactory because of:

1. If the page content changes, we want to fetch the page again.
   Solution: use a timer to purge H (different for different URLs)
2. *Spider-traps*: (final part) of URL "appended" to the URLs linked to.

   E.g., On `http://site.com/page1`, all linked URLS are in the form
   `http://site.com/page1/page`XXXX
   Allows for tracking of *sequences of user actions*
   Solution: URL *normalization* (rewrite the URL)
3. *(Near-) duplicate* pages at different URLS

How to remove near-duplicates?

Shingling:

Fix a natural number k, and consider a page p as the *set* of its k-*grams* (or k-shingles):

Now is the winter of our discontent

The set of 2-grams of p is:

$$S_p = \{\{\text{Now, is}\}, \{\text{is, the}\}, \{\text{the, winter}\}, \{\text{winter, of}\}, \{\text{of, our}\}, \{\text{our, discontent}\}\}$$

Shingles are less likely to appear in really-different pages than single words.

"Uniqueness" increases with k (usually $5 \leq k \leq 10$)

The similarity between two pages a, b is measured wit the *Jaccard* coefficient:

$$\mathrm{j}(a, b) = \frac{|S_a \cap S_b|}{|S_a \cup S_b|} \in [0, 1]$$

If $\mathrm{j}(a, b)$ is higher than a threshold, a and b are considered the same.

Outline

✔ The Web: types of data and their use

✔ Crawling

Indexing

Ranking

HITS

PageRank

Answering queries queries

After crawling, we have pages available locally.

We want to use them to to *answer queries*.

Given a query e.g., “what are the top-10 data mining algorithms”,
what are the steps to answer it?

1. Find the set S of pages containing the *important* words in the query;
   We need an *index* from words to pages

2. Sort S by quality / relevance / importance / trustworthiness.
   We need an *ranking function*

Page preprocessing

The 1st step in index construction is *preprocessing* the pages:

Stop-word removal: remove words giving no information: *"a", "the", …*

Stemming: consolidate *variations* of the same word with its *root*

e.g., replace *hope* and *hoping* with *hop*. Noisy!

Remove punctuation marks, digits, ….

Index construction

After stemming, we can see each page p as a pair

$$(\underbrace{p}_{\text{page ID}}, \underbrace{(w_1, w_2, \ldots)}_{\text{words in } p})$$

We can use MapReduce to build an *inverted index* from *words to page IDs*

map: maps $(p, (w_1, w_2, \ldots))$ to (w_1, p), (w_2, p), $\ldots$

reduce: identity (on $(w, (\underbrace{p_1, p_2, \ldots}_{\text{IDs of pages where } w \text{ appears}}))$)

Query processing

A query q is *preprocessed* (stopwords, stemming,...) to obtain $q = (w_1, w_2, \ldots)$
We then access the *inverted index* and obtain

$$(w_1, \underbrace{(p_1^{(1)}, p_2^{(1)}, \ldots)}_{D_1})$$

$$(w_2, \underbrace{(p_1^{(2)}, p_2^{(2)}, \ldots)}_{D_2})$$

$$\ldots$$

Often, *synonyms* of the words are also used.
The third step is taking the *(approximate) intersection* of the sets of documents:

$$Z = D_1 \cap D_2 \cap \ldots$$

The *approximate* intersection allows for documents not containing all the words in the query (and in the use of synonyms)

Outline

✔ The Web: types of data and their use

✔ Crawling

✔ Indexing

Ranking

HITS

PageRank

Ranking

There may be millions of documents containing all words of a common query.

User studies suggest that showing *about 10 results* is sufficient, with the top-3 being the most important.

How to choose them?Compute a *score* for the page w.r.t. the query.

$$\mathsf{score}(p, q) = f(\mathsf{contentScore}(p, q), \mathsf{reputation}(p))$$

Content Score: the relevance of the content of p to the query q.

Reputation: the quality of the page p.

Content score

Given a word w and a page p containing w, can we quantify how "important" is w in p?
(Hint: we do it all the time when looking at a newspaper, or a book)

Look at *where* in the page the word appears:

in the title, in the body, in the address, in a HTML header, in bold/italic text, in a link to another page, …

Count *how many times* the word appears;

We can keep track of this information when creating the index.

When the query contains multiple words, check if they appear close to each other.

Manipulating content scores

We don't trust all newspapers because anyone with enough money can print one

There is no cost in publishing on the Web:

pages with low-quality content are the norm, sometimes on purpose.

Website owners try to *manipulate the content score* of their pages:

1. *Content spamming*: fill pages with keywords (even invisible to the user)
2. *Cloaking*: show a different content to crawlers than to users

Crawlers (and browsers) send a *user-agent identifier* to servers, to let them know of their capabilities. A malicious publisher can check this identifier and serve different content depending on it.

Page reputation

How do we know if a newspaper has good reputation?

What about a scientific article?

1. *Page citation* (i.e., linking) mechanisms allows us to determine the quality of a page p.

   One link = one vote?

   No, must take into account the *quality of the pages linking to* p

2. *User behavior*: if a user clicks on p in the query results, then p must be good unless they come back and click on a different page!

Hacking the page reputation score

Personalization

Assume the query “how fast is a jaguar” comes in, what is the user really interested in?

Big cats? British cars? Jacksonville football players? Old Mac OSX versions?

Search engines *personalize* the ranking of results using:

- user history
- geolocation
- behavior of similar users
- recent events
- …

Ranking recap

To rank the pages that are relevant to a query, must take into account:

1. Content score
2. Page reputation
3. Personalization

These and many other ingredients are mixed according to a *continuously refined*, very secret recipe.

Outline

✔ The Web: types of data and their use

✔ Crawling

✔ Indexing

✔ Ranking

HITS

PageRank

HITS

Hypertext Induced Topic Search

query-dependent algorithm for computing the page reputation

Based on a phenomenon that was relatively common in the early Web of the '90s:

> Suppose that search engines do not work well.
> You care about baking. You have your own page with recipes.
> You know of many other pages about baking, which you *think are of good quality*.
>
> How do you enable other people to find those other pages?
>
> > You create a page p with a *links to all pages about baking that you know of*.
>
> p is a *hub* about backing

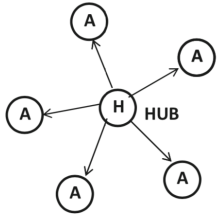
Hubs: manually-curated list of pages on a topic, considered of high quality.

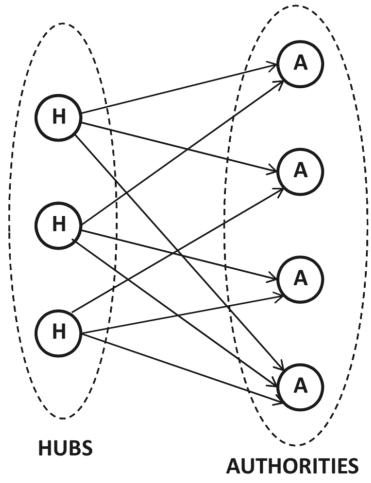
Hubs and Authorities

A hub links to pages that of high quality *according to the hub's owner*.

What can we say about a page p of *objectively* high quality?

Many hubs will link to p! p is an *authority* on the subject.


A
A
A
H
HUB
A
A
H
H
H
A
AUTHORITY
H
H


H
H
H
A
A
A
A
HUBS
AUTHORITIES

Scores

Good hubs points to good authorities that are pointed to by other good hubs.

For each page p relevant to a topic, keep *two scores*:

Hub score $\mathsf{h}(p) = \sum_{d\,:\,(p,d)\in G} \mathsf{a}(d)$

Authority score $\mathsf{a}(p) = \sum_{d\,:\,(d,p)\in G} \mathsf{h}(d)$

The $2n$ scores must be computed (n is the number of pages):
requires solving an *homogeneous* system of $2n$ linear eqs in $2n$ unknowns.

The iterative method

Solving a system of n linear eqs in n unknowns w/ Gaussian elimination takes time $O(n^3)$.

Iterative method: much *faster*, at the price of some *loss in precision*.

Idea: Let ε be a small *tolerance parameter*

$t \leftarrow 0$

Assign any initial value $x_i^{(0)}$ to each unknown x_i, $1 \leq i \leq n$

repeat:

$t \leftarrow t + 1$

For each variable x_i compute a new value $x_i^{(t)}$ *using* $x_j^{(t-1)}$, $j \neq i$, $1 \leq i \leq n$

until $|x_i^{(t)} - x^{(t-1)}| < \varepsilon$ for every i

The iterative method

"For each variable x_i compute a new value $x_i^{(t)}$ *using* $x_j^{(t-1)}$, $j \neq i$"

A *fixed point* of a function f is a point x^* such that

$$x^* = f(x^*)$$

If f "behaves well", the fixed point can be found by starting from any value of x and repeatedly apply f:

$$x^* = f(\underbrace{f(f(f(f(f(f(f(\cdots(f(x))))))))))}_{t \text{ appications give } x^{(t)}}$$

At each iteration t, the difference between x^* and $x^{(t)}$ decreases.

Few (tens of) iterations are often sufficient for convergence.

The iterative method for linear systems

System of n linear equations in n unknowns:

$$Ax = b$$

A is $n \times n$ matrix, x is $n \times 1$ vector of unknowns and b is $n \times 1$ vector of constants.

Let x^* be an *unknown* solution.

We want to build f such that $x^* = f(x^*)$.

Once we have f, we can use the iterative method to find x^*.

The iterative method for linear systems

Let's build f for linear systems.

Express A as $M - N$, where M is easy to invert. E.g., $M = I/z$, $z \in \mathbb{R}_{++}$.

For a solution x^*, it holds:

$$x^* = \underbrace{M^{-1}(Nx^* + b)}_{f(x^*)}$$

So we found a function f for which the solution x^* is a fixed point:

$$f(x) = M^{-1}(Nx + b)$$

We can then obtain x^* by applying the iterative method with f.

Iterative method for HITS

In HITS, for each page p relevant to a topic, keep *two scores*:

Hub score $\quad \mathsf{h}(p) = \sum_{d\,:\,(p,d)\in G} \mathsf{a}(d)$

Authority score $\quad \mathsf{a}(p) = \sum_{d\,:\,(d,p)\in G} \mathsf{h}(d)$

We want to express the system as $Ax = b$.

b is the $2n$-vector of zeroes (*homogeneous* system).

Let x_i be the unknown associated to $\mathsf{h}(p_i)$ for $1 \leq i \leq n$, and x_{n+i} the unknown associated to $\mathsf{a}(p_i)$

Iterative method for HITS

For every p_i let $\mathsf{t}(p_i)$ be the n-vector w/ jth entry $= 1$ if $(p_i, p_j) \in G$, 0 otherwise

For every p_i let $\mathsf{s}(p_i)$ be the n-vector w/ jth entry $= 1$ if $(p_j, p_i) \in G$, 0 otherwise

The i-th line of A, $1 \leq i \leq n$ is:

$$\underbrace{0 \ldots 0}_{i-1} 1 \underbrace{0 \ldots 0}_{n-i} \; -\mathsf{t}(p_i)$$

The $n + i$-th line of A, $1 \leq i \leq n$ is:

$$-\mathsf{s}(p_i) \; \underbrace{0 \ldots 0}_{i-1} 1 \underbrace{0 \ldots 0}_{n-i}$$

Notice that A has 1s along the diagonal.

Iterative method for HITS

We now have A, x, and $b = 0$ for HITS.

Use iterative method for linear systems:

$$f(x) = M^{-1}(Nx + b) = M^{-1}Nx, \text{ where } M - N = A.$$

Set $M = I$, so $N = I - A$, and $M^{-1} = I$, and $M^{-1}N = N = I - A$

We get

$$f(x) = (I - A)x$$

We have an f such that for a solution x^*, it holds $x^* = f(x^*)$.

Iterative method for HITS

$$f(x) = (I - A)x$$

$I - A$ is like $-A$, but with 0s along the diagonal.

The i-th line of $I - A$, $1 \leq i \leq n$ is:

$$\underbrace{0 \ldots 0}_{n} \; \mathsf{t}(p_i)$$

The $n + i$-th line of $I - A$, $1 \leq i \leq n$ is:

$$\mathsf{s}(p_i) \; \underbrace{0 \ldots 0}_{n}$$

We have $x^{(t+1)} = f(x^{(t)})$, i.e.:

$$\underbrace{x_i^{(t+1)}}_{\mathsf{h}^{(t+1)}(p_i)} = (I - A)_i x^{(t)} = \sum_{j:\,(p_i,p_j)\in G} \underbrace{x_{n+j}^{(t)}}_{\mathsf{a}^{(t)}(p_j)}$$

$$\underbrace{x_{n+i}^{(t+1)}}_{\mathsf{a}^{(t+1)}(p_i)} = (I - A)_{n+i} x^{(t)} = \sum_{j:\,(p_j,p_i)\in G} \underbrace{x_j^{(t)}}_{\mathsf{h}^{(t)}(p_i)}$$

Iterative methods for HITS

(Slight) annoyance:

A homogeneous system $Ax = 0$ has 1 solution equal to 0 or *infinite solutions*

To ensure the uniqueness of the solution, we add constraints:

we require the scores to be *normalized* so the sum of their squares is 1:

$$\sum_{i=1}^{n} {x_i}^2 = 1 \quad \text{and} \quad \sum_{i=1}^{n} {x_{n+i}}^2 = 1$$

Iterative method for HITS

Start with $\mathsf{h}^{(0)}(p) = \mathsf{a}^{(0)}(p) \leftarrow \frac{1}{\sqrt{|G|}}$

Repeat:

for each $p \in G$, $\mathsf{h}^{(t)}(p) \leftarrow \sum_{d:(p,d)\in G} \mathsf{a}^{(t-1)}(d)$

for each $p \in G$, $\mathsf{a}^{(t)}(p) \leftarrow \sum_{d:(d,p)\in G} \mathsf{h}^{(t-1)}(d)$

Normalize: Divide each $\mathsf{h}^{(t)}(p)$ by $\sum_{z\in G} \left(\mathsf{h}^{(t)}(z)\right)^2$ and similarly for $\mathsf{a}^{(t)}(p)$

until convergence

Attacking HITS

Suppose that I have a page about sea-lions and I want to increase its authority score.

I know a lot of really really high-quality pages about sea-lions.

I have the power to create as many webpages as I want.

How can I increase the authority score of my page?

I can create *many hubs* with links to:
all the best pages about sea-lions, and
a link to my page.

The best pages about sea lions will have *high authority scores*,
so my hubs will have *high hub scores*, and increase the authority score of my page.

Recap of HITS

HITS: a *topic-, query-*specific method to compute *relevance* of pages.

Pages could be *hubs* or *authorities*, or a mix: two scores per page.

Scores computed efficiently using the iterative method.

Requires to first find all pages relevant to a topic, and their link structure.

Relatively easy to attack.

Unclear solution: What exactly do you do with the two scores?

Outline

✔ The Web: types of data and their use

✔ Crawling

✔ Indexing

✔ Ranking

✔ HITS

PageRank

Outline

✔ The Web: types of data and their use

✔ Crawling

✔ Indexing

✔ Ranking

✔ HITS

✔ PageRank

Recap