Information Retrieval

Retrieval Models III: Pagerank, other models

Qianni Zhang

Retrieval Models III: Pagerank, others

Roadmap of the next two lectures:

- Pagerank
- Set based model
- Fuzzy set model
- Extended boolean model
- Generalised vector space

Page rank

Introduction

- The heart of Google's searching software is PageRank™, a system for ranking web pages developed by Larry Page and Sergey Brin at Stanford University
- Essentially, Google interprets a link from page v to page u as a vote, by page v, for page u.
- BUT these votes doesn't weigh the same, because Google also analyzes the page that casts the vote.

- The authority of a page is the sum of the authorities of the pages that reference the page.

Page rank

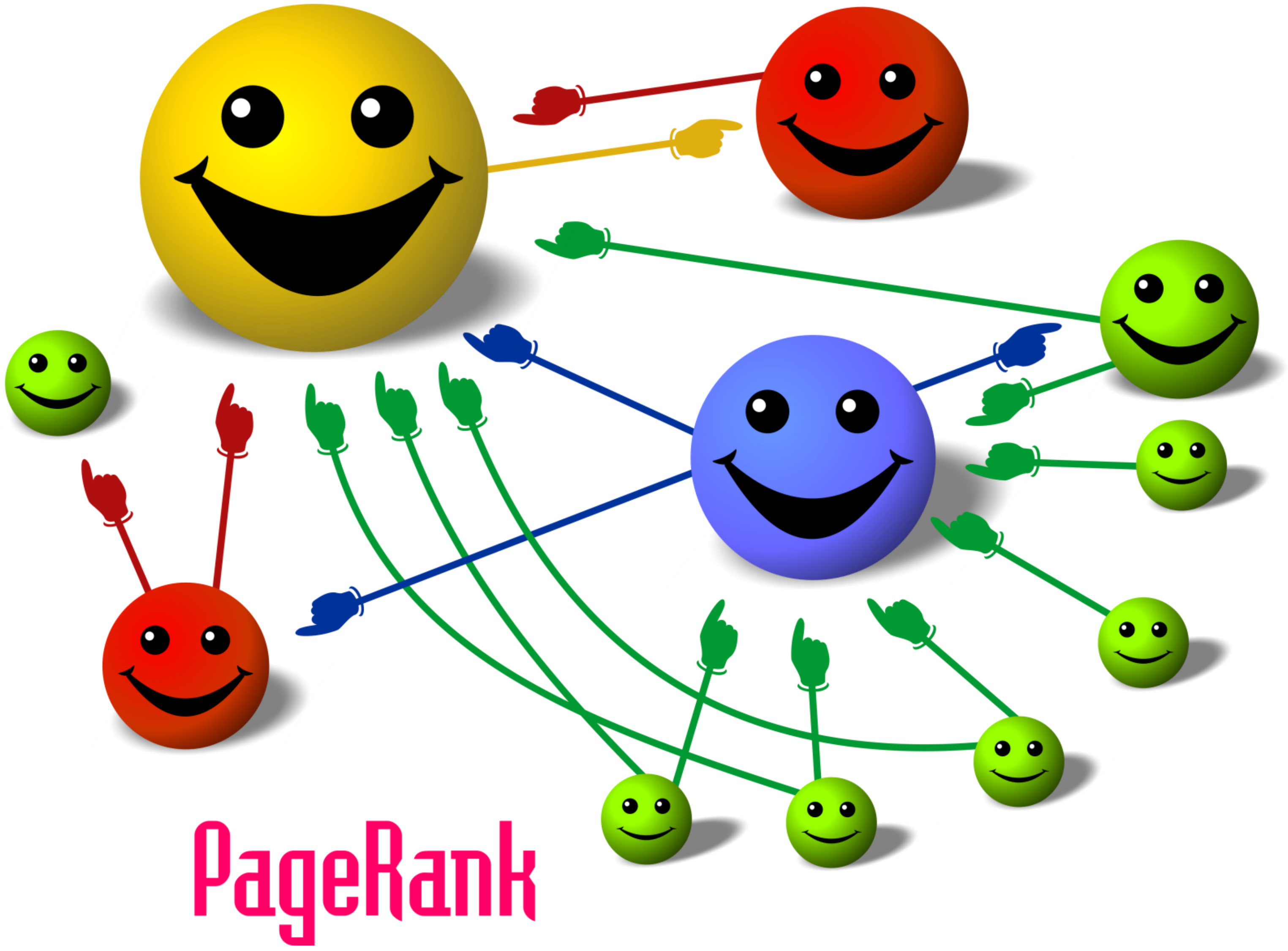
Introduction

PR	page rank reflecting its "authority", which can be used to rank pages
u, v	two pages
U	set of pages
N (*v*)	number of outgoing links in page *v*
d	a damping factor which can be set between 0 and 1

$$PR(u) := (1 - d) + d \cdot \sum_{v \to U} \frac{PR(v)}{N(v)}$$

- the summation is over the set of pages *v* that have a link to page *u*
- the page rank of page *u* is recursively defined by the page rank of those pages which link to page *u*

The size of each face is proportional to the total size of the other faces which are pointing to it.



Page rank

The Random Surfer Model

- Page rank is considered as a model of user behaviour, where a surfer clicks on links at random with no regard towards content.
- The random surfer visits a web page with a certain probability which derives from the page's page rank
- The probability that the random surfer clicks on one link is solely given by the number of links on that page.
- One page's page rank is divided by the number of links on the page.

Page rank

The Random Surfer Model

- So, the probability for the random surfer reaching one page is the sum of probabilities for the random surfer following links to this page.
- This probability is reduced by the damping factor *d*.
- The justification within the Random Surfer Model, therefore, is that
  - the surfer does not click on an infinite number of links,
  - but gets bored sometimes and jumps to another page at random.

Page rank

The damping factor *d*

- The probability for the random surfer not stopping to click on links is given by the damping factor *d*,
  - depends on probability
  - set between 0 and 1
- The higher *d* is, the more likely will the random surfer keep clicking links.
- Since the surfer jumps to another page at random after he stops clicking links, the probability therefore is implemented as a constant (1-*d*) into the algorithm.

Page rank

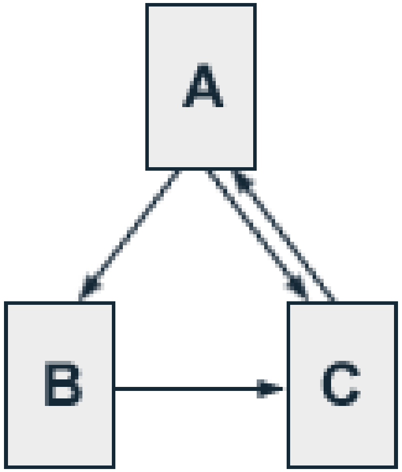
The damping factor *d*

- Regardless of inbound links, the probability for the random surfer jumping to a page is always (1-*d*)
- so a page has always a minimum page rank
- The extend of page rank benefit for a page by another page linking to it is reduced.

Page rank

An example

- A small web consisting of three pages A, B and C
- Page A links to the pages B and C, page B links to page C and page C links to page A
- The damping factor *d* is usually set to 0.85, but to keep the calculation simple we set it to 0.5.
  - PR(A) = 0.5 + 0.5 PR(C)
  - PR(B) = 0.5 + 0.5 (PR(A) / 2)
  - PR(C) = 0.5 + 0.5 (PR(A) / 2 + PR(B))



We get the following page rank values for the single pages.

- PR(A) = 1.07692308
- PR(B) = 0.76923077
- PR(C) = 1.15384615

The sum of all pages' PR is 3, equals the total number of web pages.

Page rank

The iterative computation of page rank

- For the simple three-page example it is easy to solve the according equation system to determine page rank values.
- In practice, the web consists of billions of documents and it is not possible to find a solution by inspection.
- Google search engine uses an approximating, iterative computation of page rank values.
    - Each page is assigned an initial starting value
    - The page rank of all pages are calculated in several computation circles based on the equations determined by the page rank algorithm.
- The iterative calculation shall again be illustrated by the three-page example, whereby each page is assigned a starting page rank value of 1.

Page rank

The iterative computation of page rank - example

Iteration	PR(A)	PR(B)	PR(C)
0	1	1	1
1	1	0.75	1.125
2	1.0625	0.765625	1.1484375
3	1.07421875	0.76855469	1.15283203
4	1.07641602	0.76910400	1.15365601
5	1.07682800	0.76920700	1.15381050
6	1.07690525	0.76922631	1.15383947
7	1.07691973	0.76922993	1.15384490
8	1.07692245	0.76923061	1.15384592
9	1.07692296	0.76923074	1.15384611
10	1.07692305	0.76923076	1.15384615
11	1.07692307	0.76923077	1.15384615
12	1.07692308	0.76923077	1.15384615

Page rank

Implementation - Dangling Links

- Links that point to any page with no outgoing links
- Most are pages that have not been downloaded yet
- Affect the model since it is not clear where their weight should be distributed
- Do not affect the ranking of any other page directly
- Can be simply removed before page rank calculation and added back afterwards

Page rank

Implementation

- Convert each URL into a unique integer and store each hyperlink in a database using the integer IDs to identify pages
- Sort the link structure by ID
- Remove all the dangling links from the database
- Make an initial assignment of ranks and start iteration
  - Choosing a good initial assignment can speed up the page rank
- Adding the dangling links back.

Page rank

Searching

- Two search engines:
  - Title-based search engine
  - Full text search engine
- Title-based search engine
  - Searches only the “Titles”
  - Finds all the web pages whose titles contain all the query words
  - Sorts the results by page rank
  - Very simple and cheap to implement
  - Title match ensures high precision, and page rank ensures high quality
- Full text search engine
  - Examines all the words in every stored document and also performs page rank (Rank Merging)
  - More precise but more complicated

Page rank

Summary

- Page rank is a global ranking of all web pages based on their locations in the web graph structure
- Page rank uses information which is external to the web pages - backlinks
- Backlinks from important pages are more significant than backlinks from average pages
- The structure of the web graph is very useful for information retrieval tasks.

Set Theoretic Models

- The Boolean model imposes a binary criterion for deciding relevance
- The question of how to extend the Boolean model to accommodate partial matching, i.e., a ranking for the documents retrieved has attracted considerable attention in the past
- We now discuss three alternative set theoretic models:
  - Set-Based Model
  - Extended Boolean Model
  - Fuzzy Set Model

Set-based models

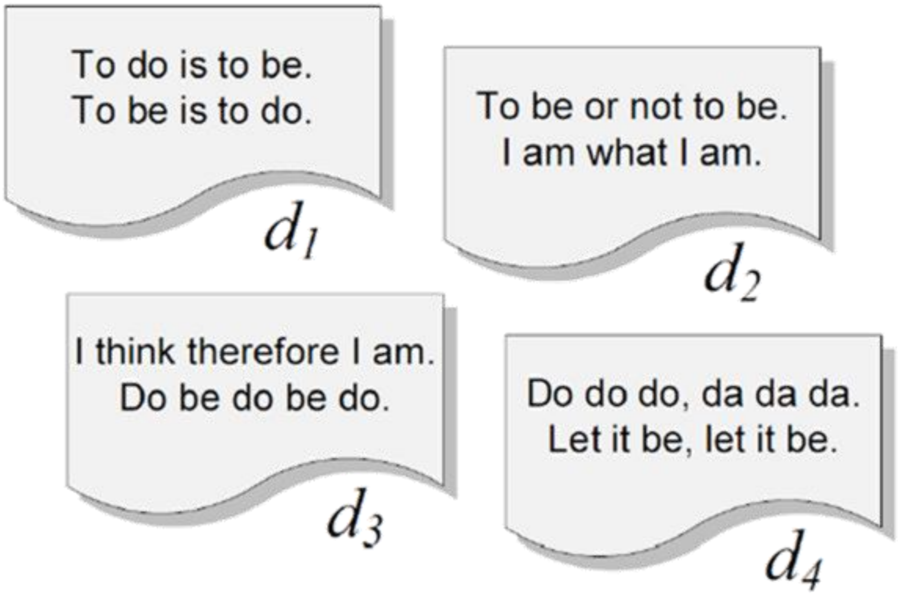
- Combines set theory with a vectorial ranking
- The fundamental idea is to use mutual dependencies among index terms to improve results
- Term dependencies are captured through termsets, which are sets of correlated terms
- The approach, which leads to improved results with various collections, constitutes the first IR model that effectively took advantage of term dependence with general collections

Termsets

- Termset is a concept used in place of the index terms
  - A termset $S_i = \{k_a, k_b, ..., k_n\}$ is a subset of the terms in the collection
  - If all index terms in S_i occur in a document d_j then we say that the termset S_i occurs in d_j
- There are 2^t termsets that might occur in the documents of a collection, where t is the vocabulary size
  - However, most combinations of terms have no semantic meaning
  - Thus, the actual number of termsets in a collection is far smaller than 2^t

Termsets

- Let t be the number of terms of the collection
  - Then, the set $V_S = \{S_1, S_2, ..., S_{2^t}\}$ is the vocabulary-set of the collection
- To illustrate, consider the document collection below

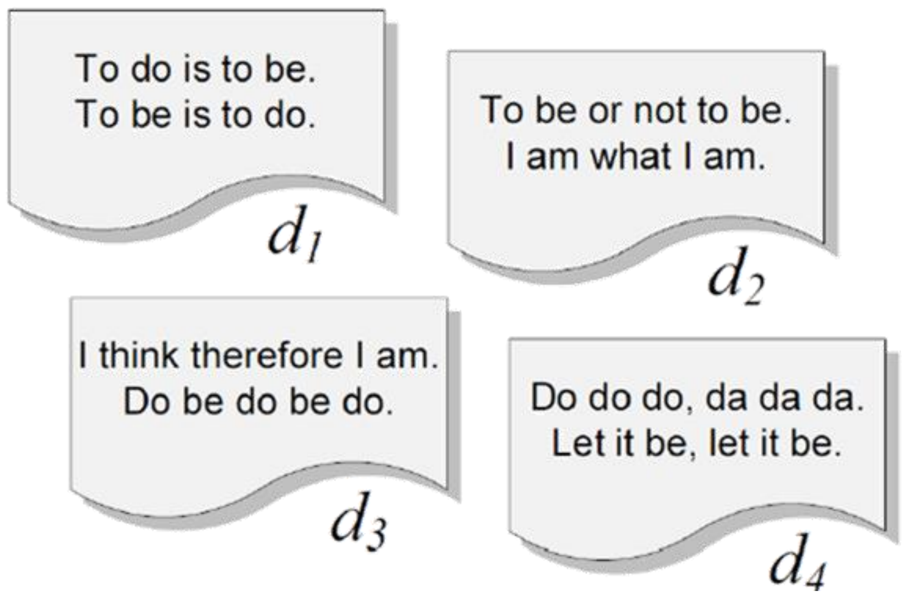


Termsets

- Define k_a = to, k_b = do, k_c = is, k_d = be, k_e = or, k_f = not, k_g = I, k_h = am, k_i = what, k_j = think, k_k = therefore, k_l = da, k_m = let, k_n = it

k_a = to	k_d = be	k_g = I	k_j = think	k_m = let
k_b = do	k_e = or	k_h = am	k_k = therefore	k_n = it
k_c = is	k_f = not	k_i = what	k_l = da	

- Further, let the letters *a...n* refer to the index terms $k_a ... k_n$, respectively



Termsets

- Consider the query q as “to do be it”, i.e. $q = \{k_a, k_b, k_d, k_n\}$
- For this query, the vocabulary-set is as below

Termset	Set of Terms	Documents
S_a	$\{a\}$	$\{d_1, d_2\}$
S_b	$\{b\}$	$\{d_1, d_3, d_4\}$
S_d	$\{d\}$	$\{d_1, d_2, d_3, d_4\}$
S_n	$\{n\}$	$\{d_4\}$
S_{ab}	$\{a, b\}$	$\{d_1\}$
S_{ad}	$\{a, d\}$	$\{d_1, d_2\}$
S_{bd}	$\{b, d\}$	$\{d_1, d_3, d_4\}$
S_{bn}	$\{b, n\}$	$\{d_4\}$
S_{abd}	$\{a, b, d\}$	$\{d_1\}$
S_{bdn}	$\{b, d, n\}$	$\{d_4\}$

Notice that there are 10 termsets that occur in our collection, out of the maximum of 16 termsets that can be formed with the terms in q

Termsets

- At query processing time, only the termsets generated by the query need to be considered
  - A termset composed of n terms is called an n-termset
  - Let N_i be the number of documents in which S_i occurs
- An n-termset S_i is said to be frequent if the number of documents S_i appears in is greater than or equal to a given threshold
  - This implies that an n-termset is *frequent* if and only if all of its $(n-1)$-termsets are also frequent
  - **Frequent termsets** can be used to reduce the number of termsets to consider with long queries

Termsets

- Let the threshold on the frequency of termsets be 2
- To compute all frequent termsets for the query
- $q = \{k_a, k_b, k_d, k_n\}$ ("to do be it")
  1. Compute the frequent 1-termsets and their inverted lists:
     - $S_a = \{d_1, d_2\}$
     - $S_b = \{d_1, d_3, d_4\}$
     - $S_d = \{d_1, d_2, d_3, d_4\}$
  2. Combine the inverted lists to compute frequent 2-termsets:
     - $S_{ad} = \{d_1, d_2\}$
     - $S_{bd} = \{d_1, d_3, d_4\}$
  3. Since there are no frequent 3-termsets, stop

Termset	Set of Terms	Documents
S_a	$\{a\}$	$\{d_1, d_2\}$
S_b	$\{b\}$	$\{d_1, d_3, d_4\}$
S_d	$\{d\}$	$\{d_1, d_2, d_3, d_4\}$
S_n	$\{n\}$	$\{d_4\}$
S_{ab}	$\{a, b\}$	$\{d_1\}$
S_{ad}	$\{a, d\}$	$\{d_1, d_2\}$
S_{bd}	$\{b, d\}$	$\{d_1, d_3, d_4\}$
S_{bn}	$\{b, n\}$	$\{d_4\}$
S_{abd}	$\{a, b, d\}$	$\{d_1\}$
S_{bdn}	$\{b, d, n\}$	$\{d_4\}$

Termsets

- Notice that there are only 5 frequent termsets in our collection
- Inverted lists for frequent n-termsets can be computed by starting with the inverted lists of frequent 1-termsets
  - Thus, the only indices required are the standard inverted lists used by any IR system
- This is reasonably fast for short queries up to 4-5 terms

Ranking Computation

- The ranking computation is based on the Vector Space Model
- Using termsets instead of index terms
- Given a query q, specified as a set of index terms, let
  - $\{S_1, S_2, \ldots\}$ be the set of termsets originated from q
  - N_i be the number of documents in which termset S_i occurs
  - N be the total number of documents in the collection
  - $F_{i,j}$ be the frequency of termsets S_i in document d_j , $F_{i,j} \neq 0$
- For each pair $[S_i , d_j]$, in which S_i occurs in d_j , we can compute a weight $W_{i,j}$, given by

$$W_{i,j} = (1 + \log F_{i,j}) \log(1 + \frac{N}{N_i}), \text{ if } F_{i,j} \neq 0$$

- For termsets that are not in the document, $W_{i,j} = 0$
- We also compute a $W_{i,q}$ value for each pair $[S_i , q]$

Ranking Computation

The weights of interest considering the query $q = \{k_a, k_b, k_d, k_n\}$ and the document d_3 are (assuming minimum threshold frequency of 1)

Termset	Weight	
S_a	$W_{a,3}$	0
S_b	$W_{b,3}$	(1+log3)*log(1+4/3)=0.544
S_d	$W_{d,3}$	(1+log2)*log(1+4/4)=0.39
S_n	$W_{n,3}$	0
S_{ab}	$W_{ab,3}$	0
S_{ad}	$W_{ad,3}$	0
S_{bd}	$W_{bd,3}$	(1+log2)*log(1+4/3)=0.4784
S_{bn}	$W_{bn,3}$	0
S_{abd}	$W_{abd,3}$	0
S_{bdn}	$W_{bdn,3}$	0

Termsets

Ranking Computation

- A document d_j and a query q are represented as vectors in a 2^t dimensional space of termsets

$$\begin{aligned}\vec{d_j} &= (\mathcal{W}_{1,j}, \mathcal{W}_{2,j}, \ldots, \mathcal{W}_{2^t,j}) \\ \vec{q} &= (\mathcal{W}_{1,q}, \mathcal{W}_{2,q}, \ldots, \mathcal{W}_{2^t,q})\end{aligned}$$

- The rank of d_j to the query q is computed using the similarity function as in the vector space model

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{S_i} \mathcal{W}_{i,j} \times \mathcal{W}_{i,q}}{|\vec{d_j}| \times |\vec{q}|}$$

- Normalisation needed to avoid counter-intuitive results
- Ranking of documents is done based on sim(d_j, q)