

Web Mining

© 2007-2008 Mauro Brunato and Elisa Cilia

Academic Year 2007-2008, second semester
Last revision: April 15, 2008

Abstract

These lecture notes contain the same text as the slides, in a more readable and compact form. They are constantly updated (see the last revision date below the title), and may contain some material that hasn't been presented yet (mainly at the end).

Contents

1	Introduction	1
1.1	What is Web Mining	1
1.2	What is this course about?	2
1.3	Prerequisites	4
2	Web crawlers	5
2.1	Crawling principles	5
2.2	DNS	6
2.3	Concurrent web fetches	6
2.4	Link extraction	7
3	Web crawlers (I)	7
3.1	Robot exclusion	7
3.2	Trap avoidance	8
3.3	Page repository	8
3.4	Queues	8
4	Indexing	8
4.1	Direct and inverted indexing	10
5	Retrieval and Ranking	13
5.1	Recall and Precision	13
6	Relevance Ranking (I)	15
6.1	Vector-Space Model	15
6.2	Relevance feedback	17
7	Relevance and Ranking (II)	19
7.1	Probabilistic relevance feedback	19
7.2	Other issues	21

8 Similarity search	23
9 Similarity Search (II)	25
10 PageRank	27
11 HITS	29
12 Clustering	30
12.1 Introduction	30
12.2 Partitioning Approaches	31
12.2.1 Agglomerative clustering	31
12.2.2 k-Means Algorithm	33
13 Geometric Embedding Approaches	34
13.0.3 Multidimensional scaling	35
13.0.4 FastMap	35

Thursday, February 21, 2008

1 Introduction

Essential data

Course webpage:

<http://disi.unitn.it/~brunato/webmining/>

Teachers

- Mauro Brunato e-mail: brunato@disi.unitn.it
- Elisa Cilia e-mail: cilia@disi.unitn.it

Textbook

SOUMEN CHAKRABARTI

Mining the Web — Discovering Knowledge from Hypertext Data
Morgan Kaufmann Publishers, San Francisco, 2003.

1.1 What is Web Mining

What is Web Mining?

- The Web is an *unstructured* (or, at most, *semistructured*) collection of data.
- Data come in form of human-readable texts and images.
- Data are hyperlinked.

The Web is not a database

- A complete description of data items (a *schema*) is missing.

- Every word on a page can be an *attribute*

The Web is a messy —although exciting— collection of human-readable data and human-exploitable hyperlinks.

Exceptions

Web services

Web protocols and languages convey data between applications.

Data unreadable by humans, need application-level processing.

It ain't no web...

Semantic web

Web protocols associated to well-structured data encasings (schemas). Goal: let both computers and users understand the meaning of what is being displayed.

Most wanted!

Who cares about that?

Metadata-rich web pages

How to climb to the tops of Google listings: put keywords and relevant data into <meta> tags of HTML pages.

I'm too lazy or too busy to do that either.

1.2 What is this course about?

Topics covered by this course

- Crawling
- Indexing
- Retrieval and ranking
- Clustering
- Applications

Topics not covered

- Classification (partially covered in Machine learning course as *supervised learning*)
- Web usage mining (how users move between pages)

Crawling

One common need of hypertext processing: the ability of fetching and storing a large number of documents.

Crawlers, Spiders, Web Robots, Bots

An example

NAME

Wget - The non-interactive network downloader.

SYNOPSIS

wget [option]... [URL]...

Indexing and retrieval

Process collected documents into an index suitable for answering queries.

Big issue

Unlike a RDBMS, *order of answers is fundamental*: the user wants to see relevant data first!

In other words: maximize the probability that the first few answers will satisfy the user's needs, otherwise you will soon have no users to satisfy.

Search engines

The union of a web crawler and a web index is a *search engine*.

- Lycos: founded January 1994, operational June 1994, IPO April 1996.
- WebCrawler: research project, spring 1994.
- AltaVista (DEC): developed spring 1995, launched December 1995. Over 60 patents, 2M queries/day after 3 weeks.
- HotBot, Inktomi, Excite, **Google**, AskJeeves, MS Live Search. . .

Many of them have “More like this” type of user input.

Topic directories

Treelike structures (taxonomies)

- *Yahoo!*: started in 1994, company April 1995. “Yet Another Hierarchical Officious Oracle!” Taxonomy maintained by teams of editors.
- Open Directory Project (dmoz.org) Based on collaborative / volunteered input.

Clustering

Can topic directories be built without human intervention upon documents retrieved by a crawler?

Clustering / Unsupervised learning

Discovering groups in the set of documents so that documents *inside* the same group are *more similar* than documents in different groups.

Different people disagree on what documents should go into the same group:

They use different similarity measures, and such measures are difficult to guess due to the large number of attributes.

Classification

Mainly part of supervised and semi-supervised machine learning.

Applications

- At the heart of search engines: PageRank, HITS
- Attacking a search engine: link farms, topic contamination. . .
- Countermeasures

1.3 Prerequisites

ISO/OSI and TCP/IP protocol stacks

The ISO/OSI stack

7.	Application
6.	Presentation
5.	Session
4.	Transport
3.	Network
2.	Medium Access Data Link
1.	Physical

The TCP/IP stack

4.	Application
3.	Transport
2.	Internetworking
1.	Host to Network

HTML — HyperText Markup Language

The *language* of the Web

```
<html>
<head>
  <title>This is a webpage</title>
  <meta name="author" content="John Doe">
  <meta name="keywords" content="basic tags, simple page">
</head>
<body>
  <h1>This is a web page</h1>
  This page links to <a href="other.html">another page</a>
  and to <a href="http://www.microsoft.com/">Microsoft</a>'s
  homepage.
</body>
</html>
```

Mainly contains tags that modify the *appearance* and, secondarily, the *meaning* of the text.

Also contains *metadata* (feature not always used at its best).

HTTP — HyperText Transfer Protocol

Client connects to server, usually at TCP port 80.

The *protocol* of the Web

```
GET /thispath/thispage.html HTTP/1.1
Accept: */*
Accept-Language: it-it
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X; it-it)
  AppleWebKit/418.9.1 (KHTML, like Gecko) Safari/419.3
Connection: keep-alive
Host: www.pippo.it
```

Manages interaction between client and server.

2 Web crawlers

2.1 Crawling principles

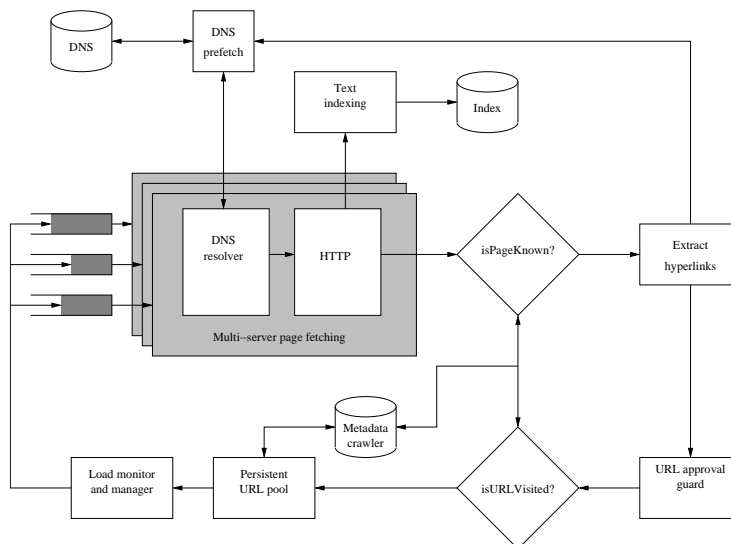
Basic crawling principles

- Start from a given set of URLs.
- Collect pages.

- Scan collected pages for hyperlinks to pages that have not been collected yet.

New URLs are potential new work and their set increases very fast.

A large-scale crawler



Engineering a large-scale crawler

- A single page fetch involves seconds of latency (\Rightarrow More fetches at the same time).
- Highly concurrent DNS (possibly multiple servers).
- No multithreading, better asynchronous sockets.
- Avoid duplicate URLs.

2.2 DNS

DNS usage

Crawler access is often spread through different domains to avoid overloading web servers (but being more demanding to DNS servers).

Cache can be slack with expiration times: better expired than late.

Problem

Standard `gethostbyname()` does not handle concurrent requests, so a custom DNS client is necessary (asynchronous sending and receiving).

Better solution: *prefetching* — do not wait for page request, but extract potential DNS queries from current pages.

2.3 Concurrent web fetches

Concurrent web fetches

Web-scale crawlers fetch $> 10^5$ pages per second. Page retrievals must proceed in parallel.

Two approaches:

- Exploit OS-level multithreading (one thread per page).
- Use non-blocking sockets and event handler.

What about multiprocessors?

Bottlenecks are network and disks, not CPU.

Concurrent fetches: multithreading

Multiple threads, *statically created* to avoid overhead.

Call to `connect()`, `send()` or `recv()` may block one thread while others run.

Pros

- Easy coding, complexity delegated to OS

Cons

- Synchronization problems and consequent IPC overhead
- Hardly optimized (OS assumes general purpose)
- One disaster spoils all threads (better with processes)

Concurrent fetches: non-blocking sockets

Single thread, vectors of non-blocking sockets, using `select()` to poll for received data.

While doing other business (indexing, saving to disk) incoming data are buffered until the next polling cycle.

Pros

- Fast, little overhead from OS
- Better control on overall status
- No need of protection or synchronization.

Cons

- Harder to code: need multiple data structures.

2.4 Link extraction

Link extraction

Web pages are parsed for hyperlinks. URLs must be canonicalized:

```
www.pippo.com/here/not/../there
      ↓
http://www.pippo.com:80/here/there/
```

Problems

Domain name - IP address relationship is many-to-many, due to load balancing needs and logical website mapping.

Avoiding repeated visits

Visited URLs must be stored to avoid unneeded duplicate visits: need of a fast memory-based *isUrlVisited?* function.

To save space, URLs are hashed, commonly by 2-level functions to exploit locality: (hostname,path).

Thursday, February 27, 2008

3 Web crawlers (I)

3.1 Robot exclusion

Robot exclusion

Hiding useless portion of a site

```
User-agent: Mauro's crawler
Crawl-delay: 1000
Disallow: /this/path
Disallow: /that/directory
```

```
User-agent: *
Disallow: /secrets
Disallow: /dynamic/page
Disallow: /ever/changing/path
```

3.2 Trap avoidance

Avoiding spider traps

Some web sites can be maliciously designed in order to crash spiders.

- Recursive links via soft aliases.
- Long URLs to overflow lexers and parsers.

3.3 Page repository

Page repository

Downloaded pages are hashed to easily spot duplicates with different URLs.

Pages are split into metadata (stored in relational form) and text (zipped for space efficiency).

3.4 Queues

Per-server working queues

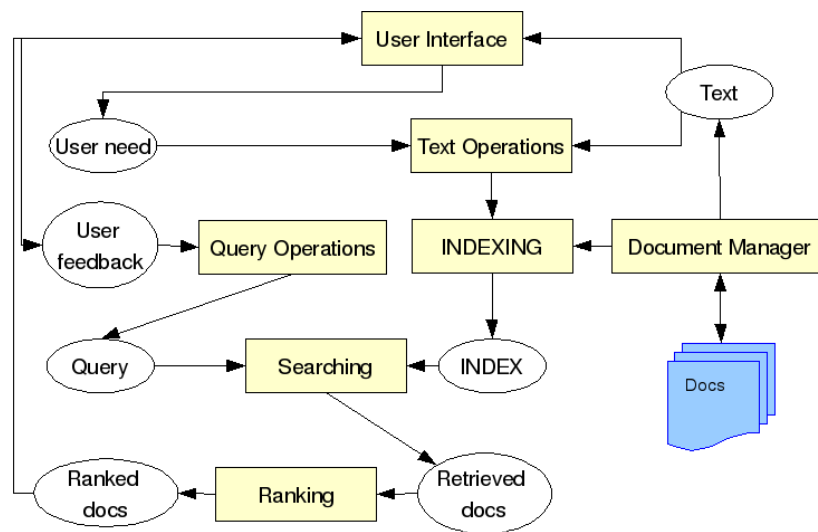
Web servers need to safeguard against DoS attacks.

- Crawlers must limit frequency of requests to the same server
- Span many different servers at once, but no more than n pages per second each (problem: DNS overload).
- Use queues.

Tuesday, February 28, 2008

4 Indexing

IR system architecture



Boolean queries

The simplest kind of query involves relationships between terms and documents:

- Documents containing the word “java”
- Documents containing the word “java” but not “coffee”
- Documents containing the phrase “java beans” or the word “API”

- Documents where “java” and “island” occur in the same sentence.

Proximity queries require the use of *inverted indices*.

Text Operations

- filter out HTML tags
- **tokenization**
 - simplest case: tokens are all nonempty sequences of characters not including spaces or punctuation marks.
- **stopword removal** (downcasing)
- **stemming**

	PLAY	
	PLAY	-S
	PLAY	-ING
	PLAY	-ED
RE-	PLAY	

- collapse variant forms (“am”, “is”, “are” all become “be”)

Beware the loss of information!

Example

Two sample documents:

Document d_1

My₁ care₂ is₃ loss₄ of₅ care₆, by₇ old₈ care₉ done₁₀.

Document d_2

Your₁ care₂ is₃ gain₄ of₅ care₆, by₇ new₈ care₉ won₁₀.

WILLIAM SHAKESPEARE — *The Life and Death of Richard the Second*, Act IV, Scene 1.

4.1 Direct and inverted indexing

Direct index

RDBMS table mapping term ID tid to document’s ID and position (did, pos):

	<u>tid</u>	<u>did</u>	<u>pos</u>
	my	1	1
	care	1	2
	is	1	3
	:	:	:
	new	2	8
	care	2	9
	won	2	10

Table “posting”:

A great waste of space...

SQL queries

1. `select did from posting where tid = 'java'`
2. `(select did from posting where tid = 'java') except (select did from posting where tid = 'coffee')`

If fast proximity search is not required, column `pos` can be discarded.

Inverted index

The following index transposes the previous one:

my	$d_1/1$	
care	$d_1/2,6,9$	$d_2/2,6,9$
is	$d_1/3$	$d_2/3$
loss	$d_1/4$	
of	$d_1/5$	$d_2/5$
by	$d_1/7$	$d_2/7$
old	$d_1/8$	
done	$d_1/10$	
your		$d_2/1$
gain		$d_2/4$
new		$d_2/8$
won		$d_2/10$

Problem

How to efficiently store this table?

Stopwords and stemming

Articles, prepositions and other forms of *function words* are useless in indexing (most documents contain them).

Definition

Such words are called *stopwords*.

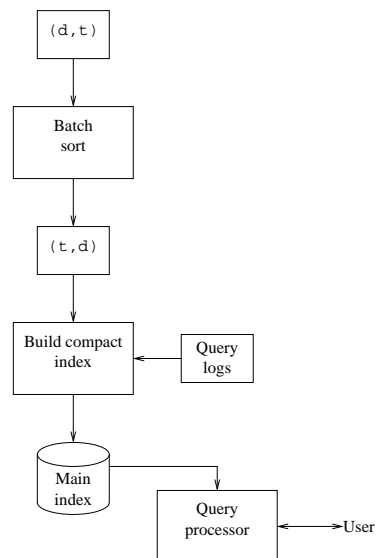
Stopwords are substituted with placeholders so offsets are unchanged.

Warning!

- Different phrases may be indexed together.
- Shakespeare again: “To be or not to be” becomes difficult to ask. . .
- Polysemy: compare “can” as verb (unimportant) with “can” as a noun (important index term).

Stemming implies morphological analysis and dictionary lookup.

A simple index



Documents in flux

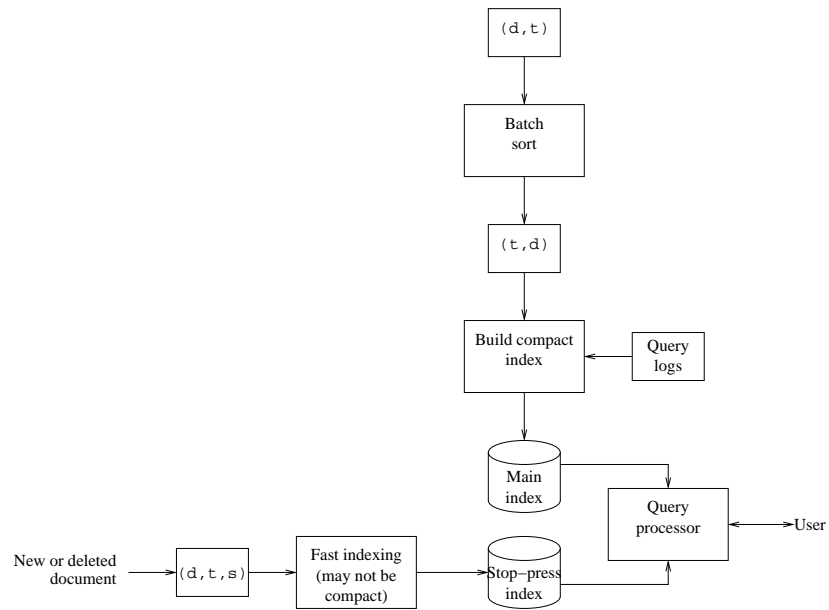
Two operations: deletion and insertion (modification is deletion followed by insertion).

Signed records (d, t, s) (s to represent which operation) compose the *stop-press* index.

Querying an index with stop-press updates

- Main index returns document set D_0 .
- Stop-press index returns matching new documents D_+ and matching deleted documents D_- .
- User obtains set $D = D_0 \cup D_+ \setminus D_-$.

Stop-press index



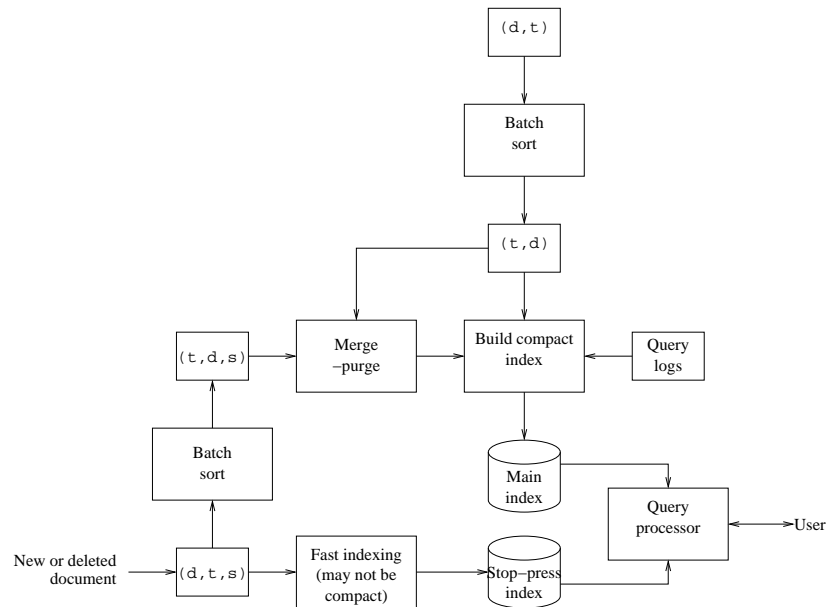
Updating the main index

Stop-press index must be built quickly.

When it is too large, the signed (d, t, s) records are sorted in (t, d, s) and merged/purged into the main index.

After this, stop-press index may be emptied.

Batch update



Index compression

Apart from stopwords, casing and punctuation, the index contains all info about a document. Thus it is large.

Solutions

- Sort doc IDs and *delta encode* them by storing *gaps*:

$$\text{word} \rightarrow (1234, 1324, 1335)$$

$$\Rightarrow \text{word} \rightarrow (1234, +90, +11)$$

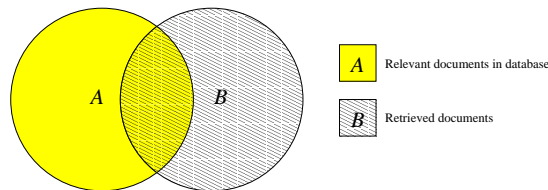
- Represent ID gap x with as few bits as possible:
 - Binary and unary encoding are suboptimal.
 - Unary code for $1 + \lceil \log_2 x \rceil$ followed by $x - 2^{\lceil \log_2 x \rceil}$ yields $1 + 2 \log_2 x$ bits (9 becomes **1110001**).
- *Lossy compression*: collect documents into buckets and index buckets separately.

Compression must be balanced with maintenance!

5 Retrieval and Ranking

5.1 Recall and Precision

Basic Definitions



- Retrieved relevant items (true positives): $A \cap B$
- Retrieved irrelevant items (false positives): $B \setminus A$
- Unretrieved relevant items (false negatives): $A \setminus B$

Precision

Which fraction of retrieved documents is relevant?

$$\frac{|A \cap B|}{|B|}$$

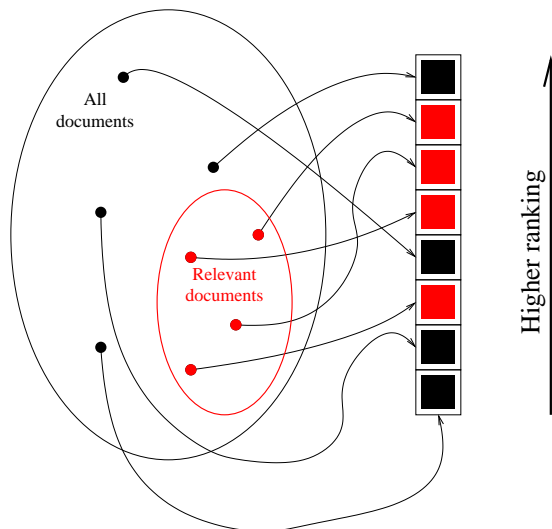
Recall

Which fraction of relevant documents was retrieved?

$$\frac{|A \cap B|}{|A|}$$

Intuition

Engine must rank relevant documents higher!



Red blocks should appear at the top.

More specialized definition with ranking

- D : corpus of $n = |D|$ documents;
- Q : set of queries.
- For query $q \in Q$, define $D_q \subset D$ as the set of all relevant documents (exhaustive, manually defined).
- Let $(d_1^q, d_2^q, \dots, d_n^q)$ be an ordering (“ranking”) of D returned by system in response to query q .
- Let $(r_1^q, r_2^q, \dots, r_n^q)$ be defined as

$$r_i^q = \begin{cases} 1 & \text{if } d_i^q \in D_q \\ 0 & \text{otherwise.} \end{cases}$$

Better definitions

Recall

Fraction of relevant documents found in the top k positions:

$$\text{recall}_q(k) = \frac{1}{|D_q|} \sum_{i=1}^k r_i^q$$

Precision

Fraction of top k documents that are relevant:

$$\text{precision}_q(k) = \frac{1}{k} \sum_{i=1}^k r_i^q$$

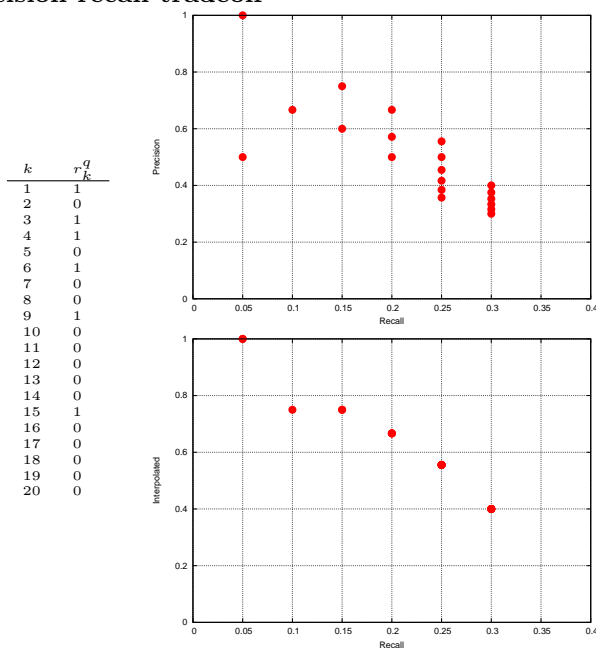
Average precision

$$\frac{1}{|D_q|} \sum_{i=1}^{|D_q|} r_k^q \text{precision}_q(k).$$

Precision-recall tradeoff

- Average precision is 1 iff all relevant documents are ranked before irrelevant ones
- Interpolated precision at recall = ρ : maximum precision for recall greater or equal to ρ .
- By convention, $\text{precision}_q(0) = 1$ and $\text{recall}_q(0) = 0$.
- Recall can be increased by increasing k , but then more and more irrelevant documents occur, driving down precision.
- Therefore, a recall-precision plot has a downward slope.

Precision-recall tradeoff



Wednesday, March 5, 2008

6 Relevance Ranking (I)

6.1 Vector-Space Model

The vector-space model

Representing document as points in a multi-dimensional space, each axis representing a term (token).

Coordinate of document d in direction of term t determined by:

Term frequency

$$\text{TF}(d, t) = \frac{n(d, t)}{\sum_{\tau} n(d, \tau)}$$

$$\text{TF}_{\text{SMART}}(d, t) = \begin{cases} 0 & \text{if } n(d, t) = 0 \\ 1 + \log(1 + \log n(d, t)) & \text{otherwise} \end{cases}$$

Inverse document frequency

$$\text{IDF}(t) = \log \frac{|D|}{|D_t|}$$

Coordinates in TFIDF-space

Document d represented by vector

$$\mathbf{d} = (d_t)_{t \in \text{terms}} \in \mathbb{R}_+^{\text{terms}}$$

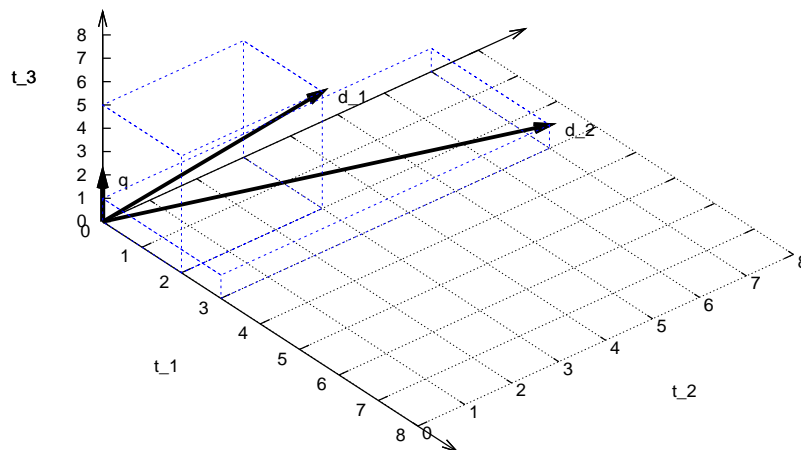
where component d_t is

$$d_t = \text{TF}(d, t) \text{IDF}(t)$$

Queries

A query q is a sequence of terms, therefore it admits a representation $\mathbf{q} = (q_t)$ in the same space as documents.

Geometrical Interpretation



Proximity measure in TFIDF-space

Problem

Given query \mathbf{q} and document \mathbf{d} , measure their proximity.

- Use Euclidean distance $\|\mathbf{d} - \mathbf{q}\|$. To avoid artifacts, vectors should be normalized: an n -fold replica of document \mathbf{d} should have the same similarity to \mathbf{q} as \mathbf{d} itself:

$$\left\| \frac{\mathbf{d}}{\|\mathbf{d}\|} - \frac{\mathbf{q}}{\|\mathbf{q}\|} \right\|.$$

- Angle between vectors \mathbf{d} and \mathbf{q} :

$$\arccos \frac{\mathbf{d} \cdot \mathbf{q}}{\|\mathbf{d}\| \|\mathbf{q}\|}.$$

TFIDF-based IR system

Information Retrieval system based on TFIDF coordinates:

- Build inverse index with $\text{TF}(t, d)$ and $\text{IDF}(t)$ information
- Given a query, map it onto TFIDF space
- Sort documents according to similarity metric
- return most similar documents

Are we done with this?

Now we are ready to refine the search!

6.2 Relevance feedback

Relevance feedback

The average web query is as few as two terms long!

After the first response, a *sophisticated* user learns how to improve his query.

For everybody else...

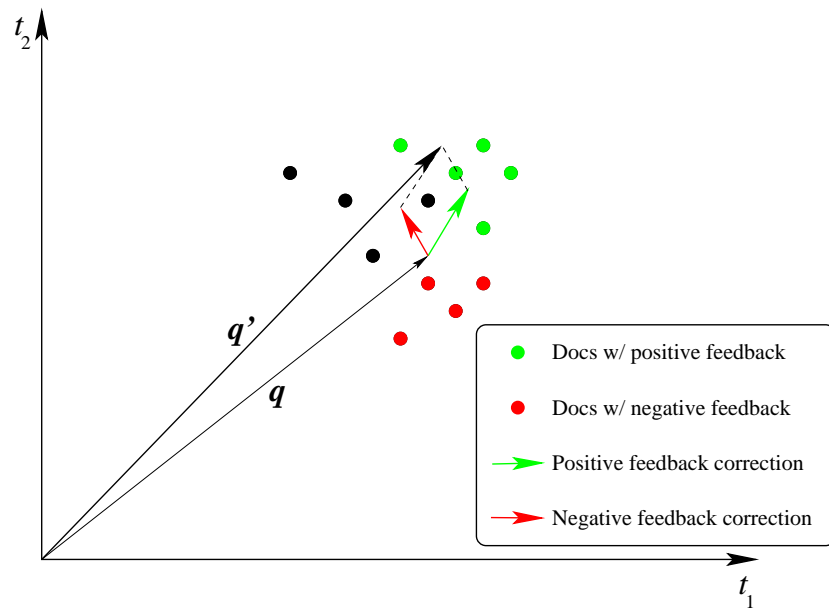
- Results page may include a rating form for documents (“Please mark documents that you have found useful”)
- User’s form submission is a form of *relevance feedback*.

Rocchio’s Method

Correct \mathbf{q} by pushing it nearer to useful docs and pulling it far from useless docs.

$$\mathbf{q}' = \alpha \mathbf{q} + \beta \sum_{\mathbf{d} \in D_+} \mathbf{d} - \gamma \sum_{\mathbf{d} \in D_-} \mathbf{d}.$$

Parameters α , β and γ control the amount of modification.



Pseudo-relevance feedback

If user input is absent:

Automatically build \$D_+\$ by assuming that a certain number (e.g., 10) of highest-ranked documents are more relevant than others.

Correction to Rocchio's formula

$$q' = \alpha q + \beta \overbrace{\sum_{d \in D_+} d}^{(1)} - \gamma \overbrace{\sum_{d \in D_-} d}^{(2)}.$$

One bad word may spoil it all

Not all words in documents in \$D_+\$ and \$D_-\$ should be used in sums (1) and (2).

For every document in \$D_+\$ and \$D_-\$ only take the 10 words with highest IDF index.

Relevance feedback on Google

None.

Why?

- Because user doesn't want to bother about checking boxes.
- Because \$q'\$ has many more components than \$q\$.

However, a "Documents like this" feature is present.

Thursday, March 6, 2008

7 Relevance and Ranking (II)

7.1 Probabilistic relevance feedback

A statistical approach

Problem

Rocchio's approach, metric ranking and PageRank are not a well justified, methodologically sound approach, but just a heuristically motivated and operational method.

A parametric method, with parameters related to the specific document set, would be more flexible.

Let R be a boolean (i.e., $\{0, 1\}$ -valued) random variable expressing the relevance of a document.

Statistical approach

Treat all quantities as random variables:

- the document is represented by random variable \mathbf{d} whose outcome is one of the possible documents in the corpus.
- same for query, represented by random variable \mathbf{q} .
- Relevance is represented by a boolean random variable R .

Odds ratio

$$\begin{aligned} \frac{\Pr(R|\mathbf{q}, \mathbf{d})}{\Pr(\bar{R}|\mathbf{q}, \mathbf{d})} &= \frac{\frac{\Pr(R, \mathbf{q}, \mathbf{d})}{\Pr(\mathbf{q}, \mathbf{d})}}{\frac{\Pr(\bar{R}, \mathbf{q}, \mathbf{d})}{\Pr(\mathbf{q}, \mathbf{d})}} = \frac{\Pr(R|\mathbf{q}) \Pr(\mathbf{d}|R, \mathbf{q})}{\Pr(\bar{R}|\mathbf{q}) \Pr(\mathbf{d}|\bar{R}, \mathbf{q})} \\ &\propto \frac{\Pr(\mathbf{d}|R, \mathbf{q})}{\Pr(\mathbf{d}|\bar{R}, \mathbf{q})}. \end{aligned}$$

- Occurrence of term t represented by boolean random variable x_t .
- Let x_{dt} be 1 iff term t appears in document d (0 otherwise).

Assume that term occurrences are independent given the query and the value of R

Therefore

$$\begin{aligned} \frac{\Pr(\mathbf{d}|R, \mathbf{q})}{\Pr(\mathbf{d}|\bar{R}, \mathbf{q})} &\approx \frac{\prod_t \Pr(x_t = x_{dt}|R, \mathbf{q})}{\prod_t \Pr(x_t = x_{dt}|\bar{R}, \mathbf{q})} \\ &= \frac{\prod_{t \in d} \Pr(x_t = 1|R, \mathbf{q}) \prod_{t \notin d} \Pr(x_t = 0|R, \mathbf{q})}{\prod_{t \in d} \Pr(x_t = 1|\bar{R}, \mathbf{q}) \prod_{t \notin d} \Pr(x_t = 0|\bar{R}, \mathbf{q})} \\ &= \frac{\prod_{t \in d} a_{tq} \prod_{t \notin d} (1 - a_{tq})}{\prod_{t \in d} b_{tq} \prod_{t \notin d} (1 - b_{tq})} \propto \prod_{t \in d} \frac{a_{tq}(1 - b_{tq})}{b_{tq}(1 - a_{tq})}, \end{aligned}$$

where $a_{tq} = \Pr(x_t = 1|R, \mathbf{q})$, $b_{tq} = \Pr(x_t = 1|\bar{R}, \mathbf{q})$.

Odds ratio ranking order

Let $a_{tq} = \Pr(x_t = 1 | R, \mathbf{q})$ and $b_{tq} = \Pr(x_t = 1 | \bar{R}, \mathbf{q})$

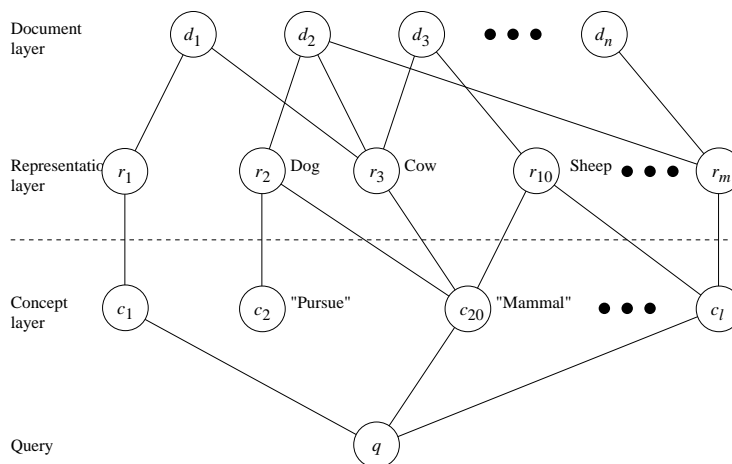
$$\frac{\Pr(R | \mathbf{q}, \mathbf{d})}{\Pr(\bar{R} | \mathbf{q}, \mathbf{d})} \propto \prod_{t \in d} \frac{a_{tq}(1 - b_{tq})}{b_{tq}(1 - a_{tq})}$$

The only parameters are a_{tq} and b_{tq} . They can be estimated by sampling on the user's relevance feedback.

Ranking as a Bayesian inference problem

- Different kinds of entities:
 - Documents
 - Terms
 - Concepts
 - Queries
- Entities are represented by the nodes of a DAG: a Bayesian inference network.

Bayesian inference networks



- In basic vector-space IR systems representation and concept layers coincide.
- Alternatively, hierarchies of concepts can be defined by users and administrators.

Using a Bayesian inference network

- Each node is associated to a random variable, with prior *belief* that variable is true.
- Directed arc: belief of child is function of parents' beliefs.

Belief inheritance

If node v has parents u_1, \dots, u_k , probability that v is true is

$$\Pr(v = \text{true} | u_1, \dots, u_k)$$

A 2^k -entry table defines the belief of v for all combinations of parents.

Example (or-node):

$$v = 1 - \prod_{i=1}^k (1 - u_i).$$

Relevance estimation

A Bayesian network and a query q are given.

Algorithm

- For every document d :
 - Set belief of d to 1, all others to 0.
 - Propagate beliefs through network
 - Set ranking order according to belief of q .

7.2 Other issues

Other issues

Spamming

Add text (invisible to the user) in the page.

Text is different

Headings, titles, anchor text in referring pages

E.g.: IBM may not state that it produces mainframes (is that still true?), but many referring links do. Same with Toyota.

Many search engines (Lycos, Google) index unfetched pages with anchor text in their referrers.

Phrase queries

```
socks -shoes +"network protocol"
```

Naïf solution

Build a separate index for phrases.

When is the two-word phrase $t_1 t_2$ relevant?

Frequency table:

$k_{00} = k(\bar{t}_1, \bar{t}_2)$	$k_{01} = k(\bar{t}_1, t_2)$
$k_{10} = k(t_1, \bar{t}_2)$	$k_{11} = k(t_1, t_2)$

If t_1 and t_2 are independent, then $\Pr(t_1 t_2) = \Pr(t_1) \Pr(t_2)$.

Checking word independence

Word independence (null hypothesis) can be checked via likelihood ratio test:

$$\lambda = \frac{\max_{p \in \Pi_0} H(p; k)}{\max_{p \in \Pi} H(p; k)}$$

$-2 \log \lambda$ is asymptotically χ^2 -distributed with degrees of freedom equal to the difference between dimensions of Π and Π_0 .

Likelihood of null hypothesis

Word independence entails only two probability values ($p_1 = \Pr(t_1)$ and $p_2 = \Pr(t_2)$):

$$H(p_1, p_2; k_{00}, \dots, k_{11}) \propto \frac{((1-p_1)(1-p_2))^{k_{00}} ((1-p_1)p_2)^{k_{01}}}{(p_1(1-p_2))^{k_{10}} (p_1 p_2)^{k_{11}}}$$

Likelihood of alternative hypothesis

Word dependency requires four probability values ($p_{00} = \Pr(\bar{t}_1 \bar{t}_2)$, $p_{01} = \Pr(\bar{t}_1 t_2)$, $p_{10} = \Pr(t_1 \bar{t}_2)$ and $p_{11} = \Pr(t_1 t_2)$).

$$H(p_{00}, \dots, p_{11}; k_{00}, \dots, k_{11}) \propto p_{00}^{k_{00}} \dots p_{11}^{k_{11}}.$$

For both hypotheses, find parameters that maximize likelihood of observations; compute $-2 \log \lambda$ and check against χ^2 table with the desired confidence value.

Example

$-2 \log \lambda$	Phrase
271	the swiss
264	can be
257	previous year
264	mineral water

Approximate matching

- Different dialects in the same language
- Different transliterations from foreign alphabets Peking \neq Beijing; Chaikovskij \neq Tchaikovsky.

Solutions?

- Collapse near homophones: soundex, used by phone companies Bradley, Bartle, Bartlett, Brodley \Rightarrow B634
- Decompose words into q -grams ($q = 2, 3, 4$) and check overlaps: *yang* and *wong* overlap by one 2-gram over three (*ng*) Collapsing vowels overlap is $2/3$.
- Allow use of wildcards: *univ** matches *universe*, *universal*, *university*...

Metasearch systems

Idea: use a portfolio of different search engines.

- Automatically reformulate queries for different syntaxes
- Overlap is little: common documents are likely to be relevant
- Improved recall

Problem

- Search engines usually don't provide ranking values.
- Even if they did, they would be on different scales.

Thursday, March 13, 2008

8 Similarity search

Similarity search

Similar documents should be clustered in TFIDF vector space.

Engines provide a “More documents like this” link.

Main issues:

- Eliminate near-equal copies

Handling documents as queries

Let d_q be a document upon which a “more like this” function must operate.
 d_q is a query.

Easy(?) solution

Find N documents in D for which $d \cdot d_q$ is largest.

Jaccard coefficient

Let $T(d)$ be the set of tokens appearing in document d .

$$r'(d_1, d_2) = \frac{|T(d_1) \cap T(d_2)|}{|T(d_1) \cup T(d_2)|}$$

- r' is a similarity measure: $r'(d, d) = 1$ and $r'(d_1, d_2) = r'(d_2, d_1)$.
- $1 - r'$ is a metric.

Precomputing $r'(\cdot, \cdot)$

- For each $d \in D$:
 - For each term $t \in T(d)$: put record (t, d) on file f_1 .
- Sort f_1 in (t, d) order and aggregate into form (t, D_t) .
- For each term t scanned from f_1 :
 - For each pair $d_1, d_2 \in D_t$: put record $(d_1, d_2, 1)$ on file f_2 .
- Sort f_2 on (d_1, d_2) and aggregate by adding on third field.

Reducing complexity

- Jaccard coefficient can be precomputed, or every document can be pre-associated with list of 10 more similar documents.
- Avoid very frequent terms (having low IDF)

Using permutations

Approximate Jaccard ratio by using probabilities. Given sets A and B :

$$\frac{|A \cap B|}{|A \cup B|} = \Pr(x \in A \cap B | x \in A \cup B).$$

To select random element from a set $S \subset \{1, \dots, N\}$:

- Select random permutation π on N elements;
- Select element in S such that its image in π is minimum:

$$x = \arg \min_{x \in S} \pi(x) = \arg \min \pi(S)$$

When applied to $A \cup B$, this method locates an element in the intersection if and only if

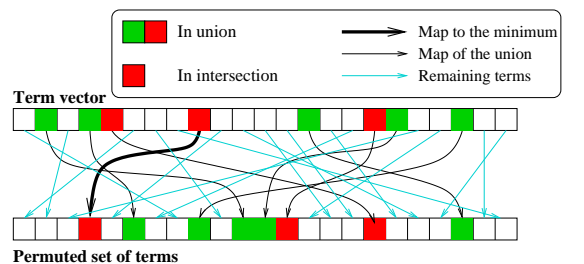
$$\min \pi(A) = \min \pi(B).$$

Why permutations work?

Given sets $A, B \subset \{1, \dots, N\}$, let us count how many permutations $\pi : \{1, \dots, N\} \rightarrow \{1, \dots, N\}$ (of the $n!$ possible) have the property

$$\min \pi(A) = \min \pi(B).$$

Let us build one such permutation



$$\binom{N}{|A \cup B|} \cdot |A \cap B| \cdot (|A \cup B| - 1)! \cdot (n - |A \cup B|)!$$

Let us build one such permutation

- The image of $A \cup B$ can be chosen in $\binom{N}{|A \cup B|}$ different ways.
- Within such image, the minimum element can be chosen within the $|A \cap B|$ elements that form the intersection.
- The remaining elements in the image of $A \cup B$ can be permuted in $(|A \cup B| - 1)!$ ways.
- The elements not in $A \cup B$ can be permuted in any of $(n - |A \cup B|)!$ ways.

Multiply all these, obtaining

$$N! \frac{|A \cap B|}{|A \cup B|}.$$

Random (inefficient) algorithm

- Generate a set Π of m permutations on the set of terms.
- $k \leftarrow 0$
- For each $\pi \in \Pi$:
 - if $\min \pi(T(d_1)) = \min \pi(T(d_2))$ then $k \leftarrow k + 1$.
- Estimate $r'(d_1, d_2) \approx \frac{k}{m}$.

Wednesday, March 19, 2008

9 Similarity Search (II)

Exploiting the same permutation for many documents

- For each random permutation π
 - For each $d \in D$ write pair $(s = \min \pi(T(d)), d)$ to file f_π .
 - Sort f_π on first field (blocks with same min are contiguous).
 - For each pair d_1, d_2 within a run of f_π with same s write (d_1, d_2) on g_π .
 - Sort g_π on (d_1, d_2) .
- Merge all g_π into g containing (d_1, d_2, n_{d_1, d_2}) counting number of occurrences of pair in all g_π 's.

Understanding the algorithm

For every permutation π , f_π maps each document to the minimum of its image; g_π contains all pair of documents that map on the same minimum.

After all g_π 's are computed, file g counts occurrences of each pair (d_1, d_2) in all g_π 's.

Near duplicates

Near-duplicate documents must be removed:

- Different hostnames in anchors
- Timestamps or maintainer names and signatures at the end of the page
- Customized formatting

Edit distance

Count the minimum number of edit operations (character deletions and insertions) to change from first to second document.

Proportional to product of document lengths.

Documents must be compared pairwise, no optimization possible.

Shingling

Solution

Use q -grams, a.k.a. *shingles* (sequences of q tokens/terms, not characters).

Given document d , let $S(d)$ be the set of its shingles (given length w).

Terms given by 32-bit number \Rightarrow 4-grams are 128-bit numbers.

Again, use Jaccard coefficient on shingle sets, this time to *remove* documents from list:

$$r(d_1, d_2) = \frac{|S(d_1) \cap S(d_2)|}{|S(d_1) \cup S(d_2)|}.$$

Collapsing mirror pages

- Similarity should not be limited to pages, but extended to web subgraphs.
- Replace `href` attributed in anchors by canonicalizing and removing stopwords
- Apply shingling and Jaccard coefficient estimation *after* URLs have been modified in such way.

Example of URL reduction

- Canonicalize, convert to lowercase.
- Collapse all punctuation.
- Remove stopwords (`http`, `cgi-bin`, `html...`).

10 PageRank

PageRank

Developed by Larry Page (hence the name) and Sergey Brin (Google founders) in 1996.

Basic idea

Define a “measure of prestige” such that the prestige of a page is proportional to that of pages that link to it.

Note: this is a recursive definition.

- Surfer moving through links forever, picking them uniformly at random.
- Let the starting page u be taken with probability p_u^0 .
- Let E be the adjacency matrix of the web: $(u, v) \in E$ (or $E_{uv} = 1$) iff link from page u to page v .

What is the probability p_v^i of being at page v after i clicks?

After one step...

What is the probability p_v^1 that surfer is at page v after 1 step? Let

$$N_u = \sum_v E_{uv}$$

be the *outdegree* of page u (sum of u -th row of E). Suppose no parallel edges exist,

$$p_v^1 = \sum_{(u,v) \in E} \frac{p_u^0}{N_u}$$

By normalizing E to have row sums equal to 1

$$L_{uv} = \frac{E_{uv}}{N_u}$$

we get

$$p_v^1 = \sum_u L_{uv} p_u^0 \quad \text{or} \quad \mathbf{p}^1 = L^T \mathbf{p}^0.$$

After i steps

$$\mathbf{p}^i = L^T \mathbf{p}^{i-1}.$$

If E is *irreducible* and *aperiodic* (but none is true), then

$$\lim_{i \rightarrow \infty} \mathbf{p}^i = \mathbf{p}$$

where \mathbf{p} is the principal eigenvector of L^T , aka its *stationary distribution*:

$$\mathbf{p} = L^T \mathbf{p} \quad (\text{eigenvalue is } 1).$$

Let p_u be the *prestige* of page u .

Dealing with bad properties

Surveys show that the Web is not strongly connected, and that random walks can be trapped into cycles.

Possible fix

Introduce “damping factor” corresponding to a user that occasionally stops following links: arbitrary probability d of going to a random page (even unconnected) at every step. Transition becomes:

$$\mathbf{p}^i = \left((1-d)L^T + \frac{d}{N}\mathbf{1}_N \right) \mathbf{p}^{i-1}.$$

Finding the eigenvector

- L is a very large matrix.
- Start with random vector $\mathbf{p} \leftarrow \mathbf{p}^0$.
- Repeat:

– Iterate vector:

$$\mathbf{p} \leftarrow \left((1-d)L^T + \frac{d}{N}\mathbf{1}_N \right) \mathbf{p}$$

– From time to time, normalize it:

$$\mathbf{p} \leftarrow \frac{\mathbf{p}}{\|\mathbf{p}\|_1}.$$

Notion of prestige is so fuzzy that nobody will ever care about the actual eigenvector!!!

Page says that 52 iterations are enough for $> 3 \times 10^8$ pages.

Handling nodes with no outlinks

If a page has no outgoing links...

First solution

Jump out of it with probability 1 and uniform destination (previously seen as “damping factor”).

Second solution

- Remove such pages from graph (needs iterated process: removing one such page can generate more of them).
- Compute PageRank for surviving nodes
- Propagate scores to eliminated nodes.

... **now what?**

The exact value of \mathbf{p} is not relevant, only the ranking order is.

PageRank is independent of page content.

Google's way of combining query and ranking is unknown.

Probably, empirical parameters and manual inspection are necessary.

Thursday, March 27, 2008

11 HITS

Hyperlink Induced Topic Search

Kleinberg, 1998

In a scientific community all good articles are either seminal (i.e., many others reference to them) or surveys (i.e., they reference to many others).

In the web

Pages may be *authorities* or *hubs*.

Two score sets:

$$\mathbf{h} = (h_u), \quad \mathbf{a} = (a_u).$$

HITS basics

Given query q , let R_q be the *root set* returned by an IR system.

Form the *expanded set* by adding all nodes linked to it:

$$V_q = R_q \cup \{u : ((u \rightarrow v) \vee (v \rightarrow u)) \wedge v \in R_q\}.$$

Let E_q be the induced link subset, $G_q = (V_q, E_q)$.

Recurrent relationship

Let hub score h_u be proportional to sum of referred authorities, authority score a_u be proportional to sum of referring hubs.

$$\begin{aligned} \mathbf{a} &= E^T \mathbf{h} \\ \mathbf{h} &= E \mathbf{a}. \end{aligned}$$

Iterated method

- Initialize \mathbf{a} and \mathbf{h} (e.g., uniformly)
- Repeat:
 - $\mathbf{h} \leftarrow E \mathbf{a}$
 - $\mathbf{a} \leftarrow E^T \mathbf{h}$
 - Normalize \mathbf{h} and \mathbf{a} .

HITS overall algorithm

1. Query the root set from an IR system
2. Expand the root set by radius 1
3. Run above algorithm
4. Report top-ranking authorities and hubs.

Problems

The principal eigenvector identifies the largest dense bipartite subgraph.

To find smaller sets, the other eigenvectors must be explored. There are iterative methods that remove known eigenvector from a system (reduce the search subspace once an eigenvector is identified).

The need of a bipartite core is bad if new pass-through pages are inserted.

12 Clustering

12.1 Introduction

Motivations for clustering

- Queries can be ambiguous, especially *web* queries (example: **star**)
- Mutual similarities in term vector space can help grouping similar documents together
- (i.e., find “clusters” of documents)
- Also a help for later manual classification à la Yahoo!.
- Assistance to fast similarity search by precomputed clusters.

The “Cluster Hypothesis”

If a user is interested in document d , he is likely to be interested in documents in the same cluster.

An example: The Scatter/Gather clustering system

key army war spangle banner

“The Star-Spangled Banner”, “Francis Scott Key”, “Fort McHenry”

film play career television

“Ellen Burstyn”, “Barbara Stanwyck”, “Milton Berle”

bright magnitude cluster constellation

“Star”, “The Galaxy”, “Extragalactic systems”

astronomer observatory astronomy

“Astronomy and astrophysics”, “Astrometry”, “Agena”

family species flower

“Blazing star”, “Brittle star”, “Bishop’s cap”

Definitions

- Corpus D of documents (or other entities) to be grouped together by similarity.
- Items $d \in D$ are characterized either *internally* by some intrinsic property (e.g., terms contained, coordinates in TFIDF space) or *externally* by a measure of distance $\delta(d_1, d_2)$ or similarity $\rho(d_1, d_2)$ between pairs.
- Examples: Euclidean distance, dot product, Jaccard coefficient.

What is a cluster?

Let D be partitioned into k disjoint subsets $\mathcal{G} = \{D_1, \dots, D_k\}$.

Criteria:

- Number of clusters $|\mathcal{G}|$
- Minimize intra-cluster distances:

$$\sum_{i=1}^k \sum_{d_1, d_2 \in D_i} \delta(d_1, d_2)$$

- Or minimize wrt cluster centroids D_i :

$$\sum_{i=1}^k \sum_{d \in D_i} \delta(d, D_i)$$

12.2 Partitioning Approaches

Partitioning Paradigms

Bottom-up

Agglomerative clustering

Top-down

k -means

12.2.1 Agglomerative clustering

Agglomerative clustering

Aka *hierarchical* clustering.

$\mathcal{G} \leftarrow \{\{d\} | d \in D\}$

Start with singleton partition

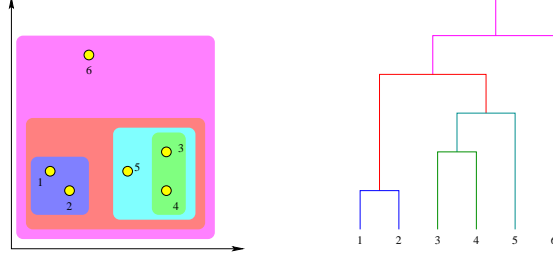
while $|\mathcal{G}| > 1$

 choose $\Gamma, \Delta \in \mathcal{G}$ maximizing $\rho(\Gamma, \Delta)$

$\Phi \leftarrow \Gamma \cup \Delta$

$G \leftarrow (G \setminus \{\Gamma, \Delta\}) \cup \{\Phi\}$

An example



Set similarity measures

$$\rho(\Gamma, \Delta) = s(\Gamma \cup \Delta)$$

where $s(\cdot)$ is a set's *self-similarity* measure:

$$s(\Phi) = \binom{\Phi}{2}^{-1} \sum_{d_1, d_2 \in \Phi} \rho(d_1, d_2).$$

Alternatives

$$\min_{d_1 \in \Gamma, d_2 \in \Delta} \rho(d_1, d_2), \quad \max_{d_1 \in \Gamma, d_2 \in \Delta} \rho(d_1, d_2),$$

$$\frac{\sum_{d_1 \in \Gamma, d_2 \in \Delta} \rho(d_1, d_2)}{|\Gamma| |\Delta|}$$

Similarity computation

Let \mathbf{d} be the vector representation of document $d \in D$. Suppose $\|\mathbf{d}\| = 1$.

Let us assume $\rho(d_1, d_2) = \mathbf{d}_1 \cdot \mathbf{d}_2$. Given $\Phi \subseteq D$, we maintain an unnormalized *group profile vector*:

$$\mathbf{p}(\Phi) = \sum_{d \in \Phi} \mathbf{d}.$$

Therefore:

$$s(\Phi) = \binom{\Phi}{2}^{-1} \left(\mathbf{p}(\Phi) \cdot \mathbf{p}(\Phi) - |\Phi| \right).$$

Computational time

$$\mathbf{p}(\Gamma \cup \Delta) \cdot \mathbf{p}(\Gamma \cup \Delta) = \mathbf{p}(\Gamma) \cdot \mathbf{p}(\Gamma) + \mathbf{p}(\Delta) \cdot \mathbf{p}(\Delta) + 2\mathbf{p}(\Gamma) \cdot \mathbf{p}(\Delta)$$

- Computation of new dot products is incremental.
- Maintain within each Γ a heap of all other Δ s sorted by $s(\Gamma \cup \Delta)$.

Starting with n singletons, compute pairwise similarities in $O(n^2)$ time, build heaps in $O(n^2 \log n)$ time.

Pick best pair to merge in $O(n)$ time (find best top of the heaps), delete them from heaps in $O(\log n)$ time.

Compute new similarity in $O(n)$ time, update all heaps in $O(n \log n)$ time.

Overall time complexity

Given $n - 1$ steps, time is $O(n^2 \log n)$.

Improvements

- Dot product time is almost constant, depending on number of terms (most coordinates are 0).
- Problem: towards the end of agglomeration, profile vectors become dense.
- Solution? Truncate profile vectors to, e.g., 1000 largest coordinates to maintain sparsity.

Thursday, April 3, 2008

12.2.2 k-Means Algorithm

k-Means Algorithm

Top-down Clustering

Iteratively refine the assignment of documents to a preset number of clusters.

k-Means with "hard" assignment

- Initial configuration: arbitrary (or chosen by a heuristic) grouping of documents into k groups
- Computation of the k corresponding centroids

k-Means Algorithm

initialize centroids to arbitrary vectors

while further improvement is possible

for each document d

 find cluster c whose centroid is **most similar** to d

 assign d to the cluster c

for each c

 recompute the centroid of c

k-Means with "soft" assignment

- no explicit assignment of documents to clusters
- each cluster is represented by a vector μ_c (not necessarily the centroid)

Goal

Find μ_c for each c that minimizes

$$\sum_d \min_c |d - \mu_c|^2 (\text{quantization error})$$

Iterative Algorithm

Basic idea:

bring the mean vectors closer to docs that they are closest to.

Repeat

for each document d
compute $\Delta\mu_c = \sum_d \begin{cases} \eta(d - \mu_c), & \text{if } \mu_c \text{ is closest to } d \\ 0 & \text{otherwise} \end{cases} \quad \mu_c \leftarrow \mu_c + \Delta\mu_c$

Alternative update rules

$$\Delta\mu_c = \eta \frac{1/|d - \mu_c|^2}{\sum_{\gamma} 1/|d - \mu_{\gamma}|^2} (d - \mu_c)$$
$$\Delta\mu_c = \eta \frac{\exp(-|d - \mu_c|^2)}{\sum_{\gamma} \exp(-|d - \mu_{\gamma}|^2)} (d - \mu_c)$$

Running Time

- bottom-up approaches can be used for the k -means algorithm initialization:
 - randomly select $O(kn)$ documents and subject them to bottom-up clustering $\rightarrow O(kn \log n)$
- n document compared against k centroids at each round $\rightarrow O(kn)$
- the number of rounds may be consider fixed (it is not strongly dependent on n or k)

Overall time complexity

$O(kn \log n)$

Wednesday, April 9, 2008

13 Geometric Embedding Approaches

Geometric Embedding Approaches

Self Organizing Maps (SOMs)

Multidimensional scaling

- FastMap

Latent Semantic Indexing (LSI)

13.0.3 Multidimensional scaling

Multidimensional scaling

- k -means and Kohonen maps require document placement in (vector) space, mutual distances are not enough.
- What if only distance (or similarity) is available?
- Useful for incorporating user feedback (“ i is quite similar to j but not to k ”).

Given data

A matrix of mutual distances d_{ij} between documents.

Goal

Embed documents in a low-dimensional space (just like Kohonen maps) so that mutual distances \hat{d}_{ij} differ as little as possible from those specified in the original matrix.

Multidimensional scaling

$$\text{stress} = \frac{\sum_{ij} (\hat{d}_{ij} - d_{ij})^2}{\sum_{ij} d_{ij}^2}.$$

How do we minimize the stress?

Iterative relaxation

1. Place all points at random (or by means of an external heuristic)
2. Iterate
 - (a) Take a point
 - (b) Move it slightly in a direction so that stress is reduced

$O(n)$ distances must be evaluated to move a point.

13.0.4 FastMap

Multidimensional scaling: FastMap

- FastMap, Faloutsos and Lin [1995]

Idea

Pretend that objects are indeed points in some unknown n -dimensional space and project these points on k mutually orthogonal directions.

Algorithm

Recursively

1. Project the objects on a carefully selected line (to obtain a coordinate)
2. Project the points to a hyperplane perpendicular to the line

until we obtain a k -coordinate representation for each object

FastMap: Key points

1. How to find a good direction or line
 2. How to "project" the original points onto the line
 3. How to project the points on the hyperplane
1. choose two pivot points a and b and the line passing through them
 2. projection of the points on the line are computed using *cosine law*

$$d_{b,x}^2 = d_{a,x}^2 + d_{a,b}^2 - 2x_1 d_{a,b} \Rightarrow x_1 = \frac{d_{a,x}^2 + d_{a,b}^2 - d_{b,x}^2}{2d_{a,b}}$$

3. Determine the distance function between two projections on the hyperplane

$$d'_{x',y'} = \sqrt{d_{x,y}^2 - (x_1 - y_1)^2}$$

FastMap: Final Considerations

- At the end we obtain a vector (x_1, \dots, x_k) for each point x in the original data set
- FastMap runs in $O(nk)$
 - for visualization tasks becomes linear in the size of the point set

Projections and Subspaces

Similarity computation in clustering algorithms:

- significant fraction of the running time is spent in it
- is proportional to the total number of non-zero components of the two vectors involved

Truncation

Only the largest components of the document vectors are retained

Examples

- a fixed number of components
- the smallest number of components that make up at least 90% of the norm of the original vector

Cutting down from 10^4 to 50 dims has no significant negative impact on clustering quality [Schutze, Silverstein]

Projections and Subspaces

How many dimensions are enough?

- Orthogonal Subspace Projection \Rightarrow look at the clustering outcome
- Non-orthogonal Subspace Projection

$$k \geq \frac{4}{\epsilon^2/2 - \epsilon^3/3} \ln n \quad \text{for } 0 < \epsilon < 1$$

grants that

$$(1 - \epsilon)\|\vec{x} - \vec{y}\|^2 \leq \|f(\vec{x}) - f(\vec{y})\|^2 \leq (1 + \epsilon)\|\vec{x} - \vec{y}\|^2$$

where n is the number of documents and $f : \mathcal{R}^d \rightarrow \mathcal{R}^k$

Data-sensitive random projections

1. select k^3 docs uniformly at random
2. find k^2 clusters (using partitioning approaches)
3. note the k^2 centroid vectors
4. for each doc d , find the projection of \vec{d} onto each of the centroid vectors
5. use the k^2 -real number vector as a representation of d
6. run a clustering algorithm on the k^2 -dimensional representation