Web Information Retrieval

Textbook by
Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze
Notes Revised by X. Meng for SEU
May 2014

Acknowledgement

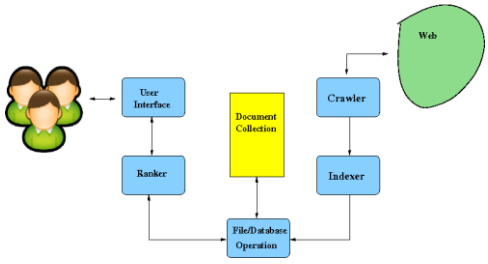
Contents of lectures, projects are extracted and organized from many sources, including those from Professor Manning's lecture notes from the textbook, notes and examples from others including Professor Bruce Croft of UMass, Professor Raymond Mooney of UT Austin, Professor David Yarowsky of Johns Hopkins University, Professor David Grossman of IIT, and Professor Brian Davidson of Lehigh.

In this segment:

- Web search engine architecture
  - Text collection
  - Text processing
  - Indexing
  - Ranking
  - User interface
  - Storage
- Link analysis
  - Hubs and authorities: HITS
  - PageRank

Search Engine Architecture

System Architecture



Collecting Text

Basic algorithm:

- *Initialize queue with URLs of known seed pages*
- *Repeat*
  - *Take a URL from queue*
  - *Fetch and parse page specified by the URL*
  - *Extract URLs from page*
  - *Add URLs to queue*
- *Until the stop condition is met*

Possible stop conditions:

- Page count reached, time limit reached, or a combination of these conditions

Fundamental assumption: The web is well connected.

Processing Text

- Parsing text
- Removing stop words
- Selecting index terms
- Stemming terms

Processing Text – parsing text

- In English words are separated by delimiters such as space
- In Chinese, it is a bit more challenge to separate words (or characters)

和尚

The two characters can be treated as one word meaning 'monk' or as a sequence of two words meaning 'and' (和, 以及) and 'still, not yet (尚未).'

8

Chinese: No White Space

莎拉波娃现在居住在美国东南部的佛罗里达。今年４月９日，莎拉波娃在美国第一大城市纽约度过了１８岁生日。生日派对上，莎拉波娃露出了甜美的微笑。

Processing Text – selecting index term

- Converting documents to a collection of *index terms*, which is typically a subset of all terms.
- Why?
  - Matching the exact string of characters typed by the user is too restrictive
    - i.e., it doesn't work very well in terms of effectiveness
  - Not all words are of equal value in a search
  - Sometimes not clear where words begin and end
    - Not even clear what a word is in some languages
      - e.g., Chinese, Korean

Processing Text – removing stop words

- Function words, e.g., *the, so*, have little meaning on their own
- High occurrence frequencies, e.g., *computer* in a computer science textbook
- Treated as *stopwords* (i.e., removed)
  - reduce index space, improve response time, improve effectiveness
- Some special cases have to be considered
  - e.g., "to be or not to be"

Processing Text – removing stop words

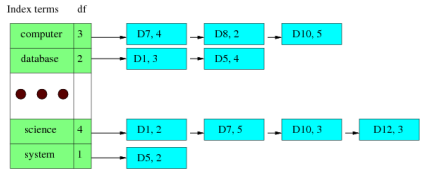
- Stopword list can be created from high-frequency words or based on a standard list
- Lists are customized for applications, domains, and even parts of documents
  - e.g., "click" is a good stopword for anchor text
- Best policy is to index all words in documents, make decisions about which words to use at query time

Text Processing -- stemming

- Many morphological variations of words
  - *inflectional* (plurals, tenses)
  - *derivational* (making verbs nouns etc.)
- In most cases, these have the same or very similar meanings
- Stemmers attempt to reduce morphological variations of words to a common stem
  - usually involves removing suffixes
- Can be done at indexing time or as part of query processing (like stopwords)

Building Inverted Index

- After selecting index terms and applying stemming algorithm, if necessary, an inverted indexing system is built



A simple inverted indexing system

14

Ranking

- Many results may "match" the query terms
- How to rank these results, possible solutions
  - No ranking
  - Ranking by the number of times a query term appears in documents
  - Ranking by some weighted average
  - …

15

User Interface

- Display one huge list 1 … n?
- Display k results per page? Allow the user to jump to certain pages directly?
- Allow user feedback for refining the search results?

16

Experiment with Code

- Let's now examine the code example for a simple server program that can take a user input

17

Link Analysis

18

Link Analysis

- Links are a key component of the Web
- Important for navigation, but also for search
  - e.g., Example website
  - "Example website" is the anchor text
  - "http://example.com" is the destination link
  - both are used by search engines

Anchor Text

- Used as a description of the content of the *destination page*
  - i.e., collection of anchor text in all links pointing to a page can be used as an additional text field
- Anchor text tends to be short, descriptive, and similar to query text
- Retrieval experiments have shown that anchor text has significant impact on effectiveness for *some types of queries*
  - i.e., more than PageRank

Authorities

- *Authorities* are pages that are recognized as providing significant, trustworthy, and useful information on a topic.
- *In-degree* (number of pointers to a page) is one simple measure of authority.
- However in-degree treats all links as equal.
- Should links from pages that are themselves authoritative count more?

Hubs

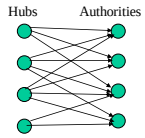
- *Hubs* are index pages that provide lots of useful links to relevant content pages (topic authorities).
- Hub pages for IR are included in the course home page:
  - http://www.cs.utexas.edu/users/mooney/ir-course
  - http://www.eg.bucknell.edu/~xmeng/webir-course/common-files/sortingWebIRCourses.html

HITS

- Hyperlink Induced Topic Search
- Algorithm developed by Kleinberg of Cornell in 1998.
- Attempts to computationally determine hubs and authorities on a particular topic through analysis of a relevant subgraph of the web.
- Based on mutually recursive facts:
  - Hubs point to lots of authorities.
  - Authorities are pointed to by lots of hubs.

Hubs and Authorities

- Together they tend to form a bipartite graph:



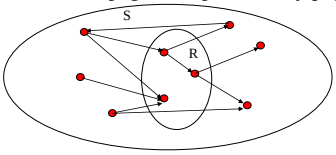
HITS Algorithm

- Computes hubs and authorities for a particular topic specified by a normal query.
- First determines a set of relevant pages for the query called the *base* set *S*.
- Analyze the link structure of the web subgraph defined by *S* to find authority and hub pages in this set.

25

Constructing a Base Subgraph

- For a specific query *Q*, let the set of documents returned by a standard search engine (e.g., google) be called the *root* set *R*.
- Initialize *S* to *R*.
- Add to *S* all pages pointed to by any page in *R*.
- Add to *S* all pages that point to any page in *R*.



26

Base Limitations

- To limit computational expense:
  - Limit number of root pages to the top 200 pages retrieved for the query.
  - Limit number of "back-pointer" pages to a random set of at most 50 pages returned by a "reverse link" query.
- To eliminate purely navigational links:
  - Eliminate links between two pages on the same host.
- To eliminate "non-authority-conveying" links:
  - Allow only m ($m \cong 4$–8) pages from a given host as pointers to any individual page.

27

Authorities and In-Degree

- Even within the base set *S* for a given query, the nodes with highest in-degree are not necessarily authorities (may just be generally popular pages like Facebook or Amazon).
- True authority pages are pointed to by a number of hubs (i.e., pages that point to lots of authorities).

28

Iterative Algorithm

- Use an iterative algorithm to slowly converge on a mutually reinforcing set of hubs and authorities.
- Maintain for each page $p \in S$:
  - Authority score: a_p (vector $\boldsymbol{a}$)
  - Hub score: h_p (vector $\boldsymbol{h}$)
- Initialize all $a_p = h_p = 1$
- Maintain normalized scores:

$$\sum_{p \in S} (a_p)^2 = 1 \qquad \sum_{p \in S} (h_p)^2 = 1$$

29

HITS Update Rules

- Authorities are pointed to by lots of good hubs:

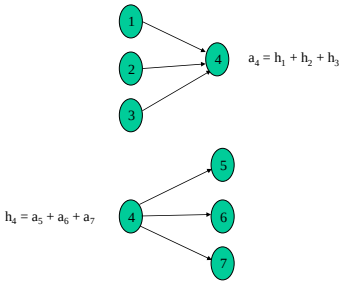
$$a_p = \sum_{q:q \to p} h_q$$

- Hubs point to lots of good authorities:

$$h_p = \sum_{q:p \to q} a_q$$

30

Illustrated Update Rules



31

HITS Iterative Algorithm

Initialize for all $p \in S$: $a_p = h_p = 1$

For i = 1 to k:

For all $p \in S$: $a_p = \sum_{q:q \to p} h_q$ *(update auth. scores)*

For all $p \in S$: $h_p = \sum_{q:p \to q} a_q$ *(update hub scores)*

For all $p \in S$: $a_p = a_p / c$ c: $\sum_{p \in S} (a_p / c)^2 = 1$ *(normalize **a**)*

For all $p \in S$: $h_p = h_p / c$ c: $\sum_{p \in S} (h_p / c)^2 = 1$ *(normalize **h**)*

32

Convergence

- Algorithm converges to a *fix-point* if iterated indefinitely.
- Define *A* to be the adjacency matrix for the subgraph defined by *S*.
  - $A_{ij} = 1$ for $i \in S, j \in S$ iff $i \to j$
- Authority vector, ***a***, converges to the principal eigenvector of $A^T A$
- Hub vector, ***h***, converges to the principal eigenvector of AA^T
- In practice, 20 iterations produces fairly stable results.

33

Results (late 1990s)

- Authorities for query: "Java"
  - java.sun.com
  - comp.lang.java FAQ
- Authorities for query "search engine"
  - Yahoo.com
  - Excite.com
  - Lycos.com
  - Altavista.com
- Authorities for query "Gates"
  - Microsoft.com
  - roadahead.com

34

Result Comments

- In most cases, the final authorities were not in the initial root set generated using Altavista.
- Authorities were brought in from linked and reverse-linked pages and then HITS computed their high authority score.

35

Finding Similar Pages Using Link Structure

- Given a page, *P*, let *R* (the root set) be *t* (e.g., 200) pages that point to *P*.
- Grow a base set *S* from *R*.
- Run HITS on *S*.
- Return the best authorities in *S* as the best similar-pages for *P*.
- Finds authorities in the "link neighbor-hood" of *P*.

36

Similar Page Results

- Given "honda.com"
  - toyota.com
  - ford.com
  - bmwusa.com
  - saturncars.com
  - nissanmotors.com
  - audi.com
  - volvocars.com

37

HITS for Clustering

- An ambiguous query can result in the principal eigenvector only covering one of the possible meanings.
- Non-principal eigenvectors may contain hubs & authorities for other meanings.
- Example: "jaguar":
  - Atari video game (principal eigenvector)
  - NFL Football team (2nd non-princ. eigenvector)
  - Automobile (3rd non-princ. eigenvector)

38

In-Class Work

- Working with your neighbor(s), compute the first round of result from the HITS algorithm for the graph represented by the following adjacency matrix

0	1	0	0
0	0	0	1
1	1	0	0
1	0	1	0

39

Work in Progress

- a1=a2=a3=a4=1, h1=h2=h3=h4=1
- a1 = h3 + h4 = 2
- a2 = h1 + h3 = 2
- a3 = h4 = 1
- a4 = h2 = 1
- $c^2 = \text{sum}(a_i^2) = 4 + 4 + 1 + 1 = 10$
- a1/c = a2/c = 2/3.16 = 0.632
- a3/c = a4/c = 1/3.16 = 0.316

40

PageRank

- Billions of web pages, some more informative than others
- Links can be viewed as information about the *popularity* (*authority*?) of a web page
  - can be used by ranking algorithm
- *Inlink* count could be used as simple measure
- Link analysis algorithms like PageRank provide more reliable ratings
  - less susceptible to link spam

Random Surfer Model

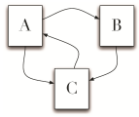
- Browse the Web using the following algorithm:
  - Choose a random number r between 0 and 1
  - If $r < \lambda$:
    - Go to a random page
  - If $r \geq \lambda$:
    - Click a link at random on the current page
  - Start again
- PageRank of a page is the probability that the "random surfer" will be looking at that page
  - links from popular pages will increase PageRank of pages they point to

Dangling Links

- Random jump prevents getting stuck on pages that
  - do not have links
  - contains only links that no longer point to other pages
  - have links forming a loop
- Links that point to the first two types of pages are called *dangling links*
  - may also be links to pages that have not yet been crawled

PageRank

- PageRank (*PR*) of page C = *PR*(A)/2 + *PR*(B)/1
- More generally,



$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

  - where B_u is the set of pages that point to u, and L_v is the number of outgoing links from page v (not counting duplicate links)

PageRank

- Don't know PageRank values at start
- Assume equal values (1/3 in this case), then iterate:
  - first iteration: *PR*(C) = 0.33/2 + 0.33 = 0.5, *PR*(A) = 0.33, and *PR*(B) = 0.17
  - second: *PR*(C) = 0.33/2 + 0.17 = 0.33, *PR*(A) = 0.5, *PR*(B) = 0.17
  - third: *PR*(C) = 0.42, *PR*(A) = 0.33, *PR*(B) = 0.25
- Converges to *PR*(C) = 0.4, *PR*(A) = 0.4, and *PR*(B) = 0.2

PageRank

- Taking random page jump into account, 1/3 chance of going to any page when $r < \lambda$
- $PR(C) = \lambda/3 + (1 - \lambda) \cdot (PR(A)/2 + PR(B)/1)$
- More generally,

$$PR(u) = \frac{\lambda}{N} + (1 - \lambda). \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

  - where N is the number of pages, λ typically 0.15

PageRank Algorithm

Let S be the total set of pages.

Let $\forall p \in S$: $E(p) = \alpha/|S|$ (for some $0<\alpha<1$, e.g. 0.15)

Initialize $\forall p \in S$: $R(p) = 1/|S|$

Until ranks do not change (much) (*convergence*)

For each $p \in S$:

$$R'(p) = \sum_{q:q \to p} \frac{R(q)}{N_q} + E(p)$$

$$c = 1/\sum_{p \in S} R'(p)$$

For each $p \in S$: $R(p) = cR'(p)$ (*normalize*)

In-Class Work

- Working with your neighbor(s), compute the first round of result from the PageRank algorithm for the graph specified by the following adjacency matrix

0	0	1	1
1	0	1	0
0	0	0	1
0	1	0	0

Work in Progress

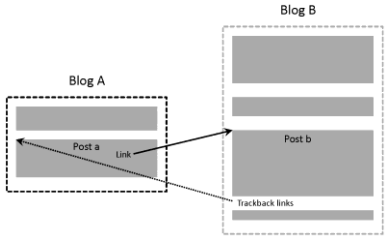
- $\alpha = 0.15$, S = 4, $E(p) = \alpha/S = 0.0375$
- $N_q = (2, 2, 1, 1)$
- Initially R' = 1/S = 0.25
- $R'(1) = \text{sum}(R_q/N_q) + E(p) = R_2/N_2 + E(p) = 0.25/2 + 0.0375 = 0.163$
- $R'(2) = \text{sum}(R_q/N_q) + E(p) = R_4/N_4 + E(p) = 0.25/1 + 0.0375 = 0.288$
- $R'(3) = \text{sum}(R_q/N_q) + E(p) = R_1/N_1 + R_2/N_2 + E(p) = 0.25/2 + 0.25/2 + 0.0375 = 0.288$
- $R'(4) = \text{sum}(R_q/N_q) + E(p) = R_1/N_1 + R_3/N_3 + E(p) = 0.25/2 + 0.25/1 + 0.0375 = 0.413$

49

Link Quality

- Link quality is affected by spam and other factors
  - e.g., *link farms* to increase PageRank
  - *trackback links* in blogs can create loops
  - links from comments section of popular blogs
    - Blog services modify comment links to contain `rel=nofollow` attribute
    - e.g., "Come visit my <a `rel=nofollow` href="http://www.page.com">web page."

Trackback Links



Information Extraction

- Automatically extract structure from text
  - annotate document using tags to identify extracted structure
- *Named entity recognition*
  - identify words that refer to something of interest in a particular application
  - e.g., people, companies, locations, dates, product names, prices, etc.

Named Entity Recognition

- Example showing semantic annotation of text using XML tags
- Information extraction also includes document structure and more complex features such as *relationships* and *events*

Fred Smith, who lives at 10 Water Street, Springfield, MA, is a long-time collector of **tropical fish.**

<PersonName><GivenName>Fred</GivenName> <Sn>Smith</Sn> </PersonName>, who lives at <address><Street >10 Water Street</Street>, <City>Springfield</City>, <State>MA</State></address>, is a long-time collector of tropical fish.

Named Entity Recognition

- *Rule-based*
  - Uses *lexicons* (lists of words and phrases) that categorize names
    - e.g., locations, peoples' names, organizations, etc.
  - Rules also used to verify or find new entity names
    - e.g., "<number> <word> street" for addresses
    - "<street address>, <city>" or "in <city>" to verify city names
    - "<street address>, <city>, <state>" to find new cities
    - "<title> <name>" to find new names

Named Entity Recognition

- Rules either developed manually by trial and error or using machine learning techniques
- *Statistical*
  - uses a probabilistic model of the words in and around an entity
  - probabilities estimated using *training data* (manually annotated text)
  - Hidden Markov Model (HMM) is one approach

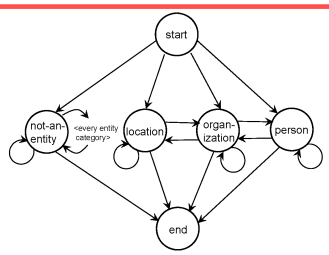
HMM for Extraction

- Resolve ambiguity in a word using *context*
  - e.g., "marathon" is a location or a sporting event, "boston marathon" is a specific sporting event
- Model context using a *generative* model of the sequence of words
  - *Markov property*: the next word in a sequence depends only on a small number of the previous words

HMM for Extraction

- *Markov Model* describes a process as a collection of states with transitions between them
  - each transition has a probability associated with it
  - next state depends only on current state and transition probabilities
- *Hidden Markov Model*
  - each state has a set of possible outputs
  - outputs have probabilities

HMM Sentence Model



- Each state is associated with a probability distribution over words (the output)

HMM for Extraction

- Could generate sentences with this model
- To recognize named entities, find sequence of "labels" that give highest probability for the sentence
  - only the outputs (words) are visible or observed
  - states are "hidden"
  - e.g., <start><name><not-an-entity><location><not-an-entity><end>
- *Viterbi* algorithm used for recognition

Named Entity Recognition

- Accurate recognition requires about 1M words of training data (1,500 news stories)
  - may be more expensive than developing rules for some applications
- Both rule-based and statistical can achieve about 90% effectiveness for categories such as names, locations, organizations
  - others, such as product name, can be much worse

Internationalization

- 2/3 of the Web is in English
- About 50% of Web users do not use English as their primary language
- Many (maybe most) search applications have to deal with multiple languages
  - *monolingual search*: search in one language, but with many possible languages
  - *cross-language search*: search in multiple languages at the same time

Internationalization

- Many aspects of search engines are language-neutral
- Major differences:
  - Text encoding (converting to Unicode)
  - Tokenizing (many languages have no word separators)
  - Stemming
- Cultural differences may also impact interface design and features provided

Chinese "Tokenizing"

1. Original text

旱灾在中国造成的影响

(the impact of droughts in China)

2. Word segmentation

旱灾 在 中国 造成 的 影响

drought at china make impact

3. Bigrams

旱灾 灾在 在中 中国 国造

造成 成的 的影 影响