CSE 5243 INTRO. TO DATA MINING

Graph Data & Introduction to Information Retrieval

Huan Sun, CSE@The Ohio State University

Slides adapted from Prof. Srinivasan Parthasarathy @OSU

Chapter 4 Graph Data:
http://www.dataminingbook.info/pmwiki.php/Main/BookPathUploads?action=downloadman&upname=book-20160121.pdf ,
http://www.dataminingbook.info/pmwiki.php

GRAPH BASICS AND A GENTLE INTRODUCTION TO PAGERANK

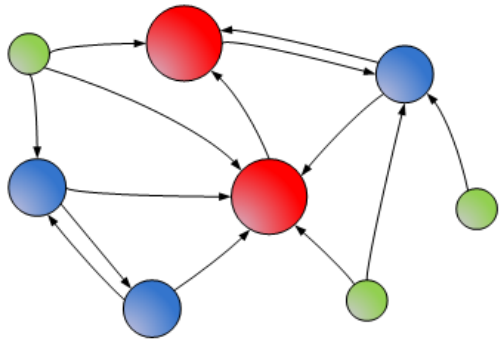
Slides adapted from Prof. Srinivasan Parthasarathy @OSU

Background

- Besides the keywords, what other evidence can one use to rate the importance of a webpage?
- Solution: Use the hyperlink structure
- E.g. a webpage linked by many webpages is probably important.
  - but this method is not global (comprehensive).
- PageRank is published by Larry Page and Sergey Brin in 1998.

Idea

- A graph representing WWW
  - Node: webpage
  - Directed edge: hyperlink



- A user randomly clicks the hyperlink to surf WWW.
  - The probability a user stop in a particular webpage is the PageRank value.

- A node that is linked by many nodes with a high importance value receives a high value itself;
  If there are no links to a node, then there is no support for that page.

Formal Formulation

Let $G = (V, E)$ be a directed graph, with $|V| = n$. The adjacency matrix of G is an $n \times n$ asymmetric matrix $\mathbf{A}$ given as

$$\mathbf{A}(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 0 & \text{if } (u, v) \notin E \end{cases}$$

Let $p(u)$ be a positive real number, called the *prestige* score for node u.

$$p(v) = \sum_u \mathbf{A}(u, v) \cdot p(u)$$

$$= \sum_u \mathbf{A}^T(v, u) \cdot p(u)$$

the prestige of a node depends on the prestige of other nodes pointing to it

Formal Formulation

Let $p(u)$ be a positive real number, called the *prestige* score for node u.

$$p(v) = \sum_u \mathbf{A}(u, v) \cdot p(u)$$

$$= \sum_u \mathbf{A}^T(v, u) \cdot p(u)$$

the prestige of a node depends on the prestige of other nodes pointing to it

Across all the nodes, we can recursively express the prestige scores as

$$\mathbf{p}' = \mathbf{A}^T\mathbf{p}$$

where $\mathbf{p}$ is an n-dimensional column vector corresponding to the prestige scores for each vertex.

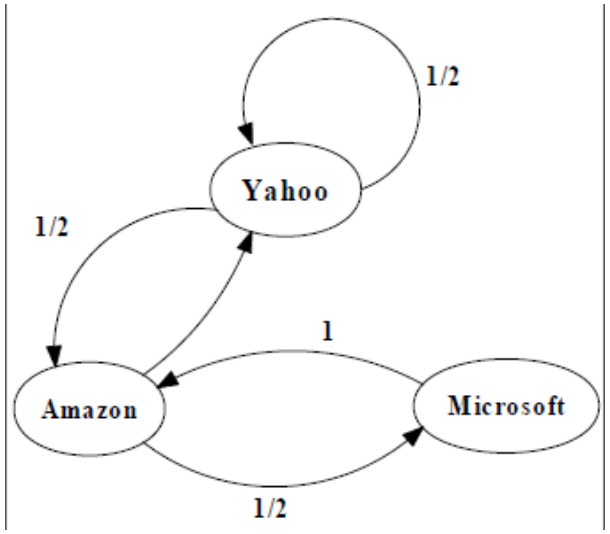
Iterative Computation

$$
\begin{aligned}
\mathbf{p}_k &= \mathbf{A}^T \mathbf{p}_{k-1} \\
&= \mathbf{A}^T (\mathbf{A}^T \mathbf{p}_{k-2}) = (\mathbf{A}^T)^2 \mathbf{p}_{k-2} \\
&= (\mathbf{A}^T)^2 (\mathbf{A}^T \mathbf{p}_{k-3}) = (\mathbf{A}^T)^3 \mathbf{p}_{k-3} \\
&= \vdots \\
&= (\mathbf{A}^T)^k \mathbf{p}_0
\end{aligned}
$$

where $\mathbf{p}_0$ is the initial prestige vector. It is well known that the vector $\mathbf{p}_k$ converges to the dominant eigenvector of $\mathbf{A}^T$ with increasing k.

For proof, read Theorem 11.18 http://mathfaculty.fullerton.edu/mathews/n2003/powermethod/PowerMethodProof.pdf

Example 1



$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

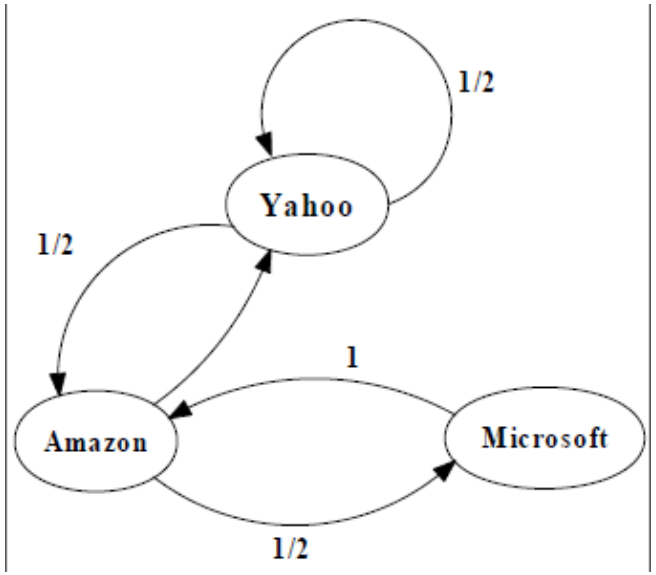
=the transpose of A (adjacency matrix)

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

Prestige vector calculation: first iteration

Example 1



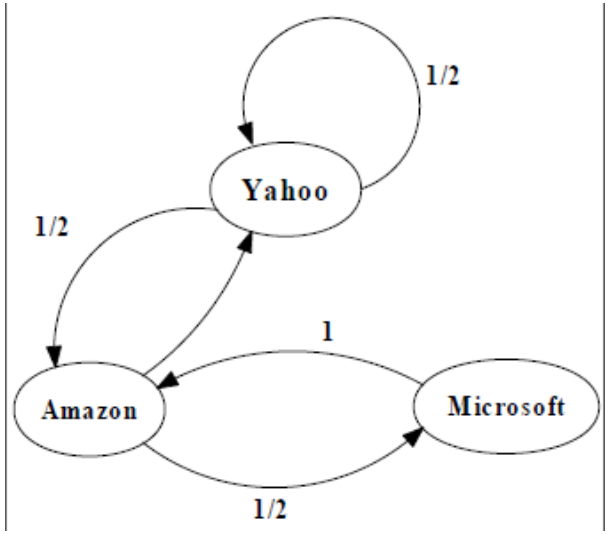
$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 5/12 \\ 1/3 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix}$$

Prestige vector calculation: second iteration

Example 1



$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 3/8 \\ 11/24 \\ 1/6 \end{bmatrix} \begin{bmatrix} 5/12 \\ 17/48 \\ 11/48 \end{bmatrix} \ldots \begin{bmatrix} 2/5 \\ 2/5 \\ 1/5 \end{bmatrix}$$

Convergence after some iterations

More Explanation & Examples

Section 4.3.2
in http://www.dataminingbook.info/pmwiki.php/Main/BookPathUploads?action=downloadman&upname=book-20160121.pdf

Algorithm 4.1

Example 4.6 (pay attention to scaling)

A simple version

$$R(u) = \sum_{v \in B_u} \frac{R(v)}{N_v}$$

- u: a webpage
- B_u: the set of u's backlinks
- N_v: the number of forward links of page v

- Initially, R(u) is 1 /N for every webpage
- Iteratively update each webpage's PR value until convergence.

A little more advanced version

- Adding a damping factor *d*
- Imagine that a surfer would stop clicking a hyperlink with probability 1-*d*

$$R(u) = \frac{(1-d)}{N} + d\sum_{v \in B_u} \frac{R(v)}{N_v}$$

- R(u) is at least (1-d), onsider the prob of a surfer randomly stops at u)
  - N is the total number of nodes.

Other applications

- Social network (Facebook, Twitter, etc)
  - Node: Person; Edge: Follower / Followee / Friend
  - Higher PR value: Celebrity
- Citation network
  - Node: Paper; Edge: Citation
  - Higher PR values: Important Papers.
- Protein-protein interaction network
  - Node: Protein; Edge: Two proteins bind together
  - Higher PR values: Essential proteins.

SEARCH ENGINES

INFORMATION RETRIEVAL IN PRACTICE

BOOK:
HTTP://CIIR.CS.UMASS.EDU/DOWNLOADS/SEIRIP.PDF

SLIDES:
HTTP://WWW.SEARCH-ENGINES-BOOK.COM/SLIDES/

Slides adapted from Prof. W. Bruce Crof @UMASS

Search Engines and Information Retrieval

Search and Information Retrieval

- Search on the Web is a daily activity for many people throughout the world
- Search and communication are most popular uses of the computer
- Applications involving search are everywhere
- The field of computer science that is most involved with R&D for search is *information retrieval (IR)*

Information Retrieval

- "*Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information.*" (Salton, 1968)

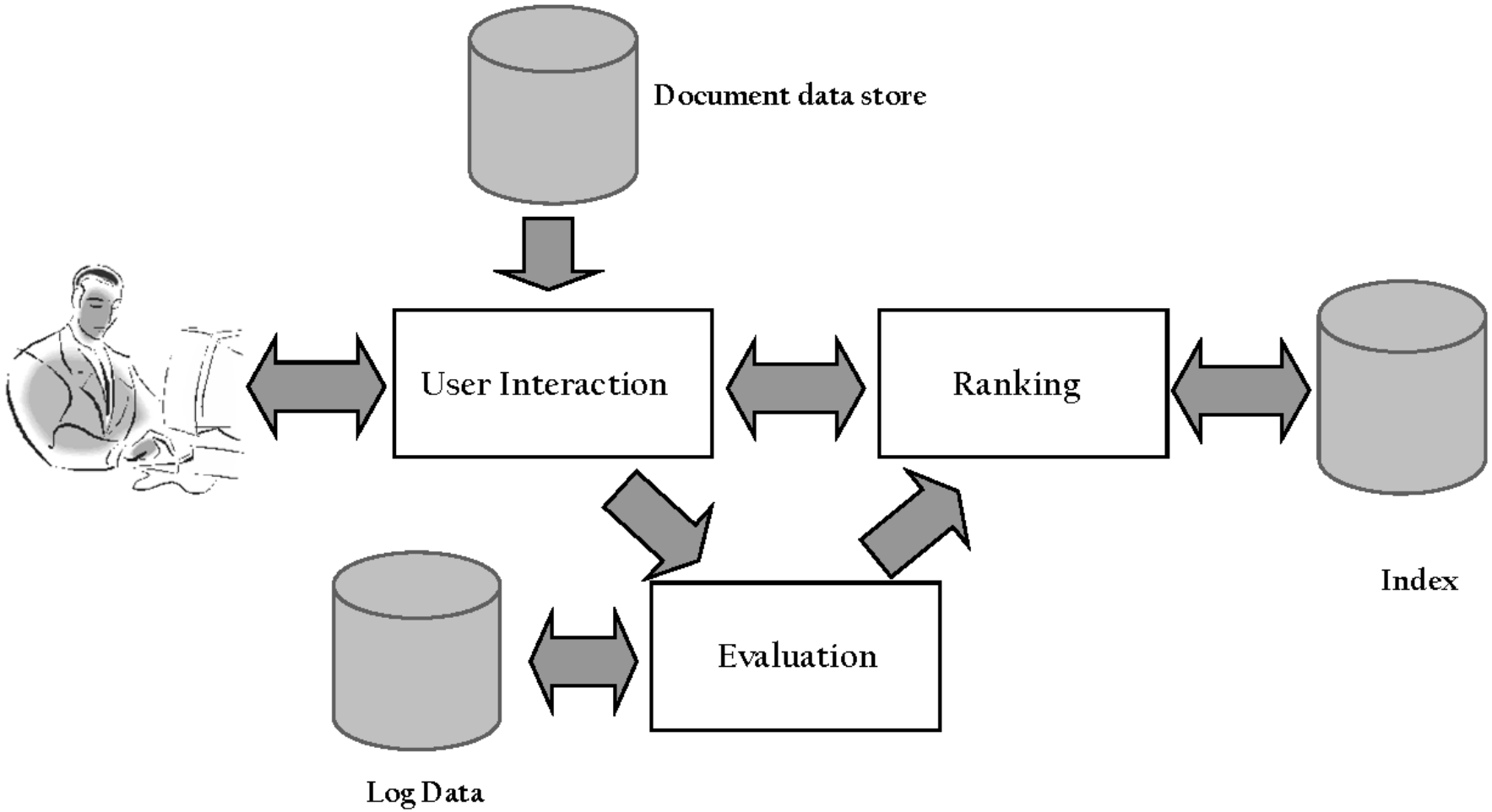
- General definition that can be applied to many types of information and search applications

- Primary focus of IR since the 50s has been on *text* and *documents*

Query Process & Search Engine Components

?

Query Process



RETRIEVAL MODELS

Language Model

- *Unigram language model*
  - probability distribution over the words in a language
  - generation of text consists of pulling words out of a "bucket" according to the probability distribution and replacing them
- N-gram language model
  - some applications use bigram and trigram language models where probabilities depend on previous words

Language Model

- A *topic* in a document or query can be represented as a language model
  - i.e., words that tend to occur often when discussing a topic will have high probabilities in the corresponding language model
- *Multinomial* distribution over words
  - text is modeled as a finite sequence of words, where there are *t* possible words at each point in the sequence
  - commonly used, but not only possibility
  - doesn't model *burstiness*

LMs for Retrieval

- 3 possibilities:
  - probability of generating the query text from a document language model
  - probability of generating the document text from a query language model
  - comparing the language models representing the query and document topics
- Models of topical relevance

Query-Likelihood Model

- Rank documents by the probability that the query could be generated by the document model (i.e. same topic)
- Given query, start with P(D|Q)
- Using Bayes' Rule
- Assuming prior is uniform, unigram model

$$p(D|Q) \overset{rank}{=} P(Q|D)P(D)$$

$$P(Q|D) = \prod_{i=1}^{n} P(q_i|D)$$

Estimating Probabilities

- Obvious estimate for unigram probabilities is

$$P(q_i|D) = \frac{f_{q_i,D}}{|D|}$$

- *Maximum likelihood estimate*
  - makes the observed value of $f_{q_i,D}$ most likely
- If query words are missing from document, score will be zero
  - Missing 1 out of 4 query words same as missing 3 out of 4

Smoothing

- Document texts are a *sample* from the language model
  - Missing words should not have zero probability of occurring

- *Smoothing* is a technique for estimating probabilities for missing (or unseen) words
  - lower (or *discount*) the probability estimates for words that are seen in the document text
  - assign that "left-over" probability to the estimates for the words that are not seen in the text

Estimating Probabilities

- Estimate for unseen words is $\alpha_D P(q_i|C)$
  - $P(q_i|C)$ is the probability for query word i in the *collection* language model for collection C (background probability)
  - α_D is a parameter

- Estimate for words that occur is

$$(1 - \alpha_D)\, P(q_i|D) + \alpha_D\, P(q_i|C)$$

- Different forms of estimation come from different α_D

Jelinek-Mercer Smoothing

- α_D is a constant, λ
- Gives estimate of

$$p(q_i|D) = (1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}$$

- Ranking score

$$P(Q|D) = \prod_{i=1}^{n}((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

- Use logs for convenience
  - accuracy problems multiplying small numbers

$$\log P(Q|D) = \sum_{i=1}^{n}\log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

Where is *tf.idf* Weight?

$$\log P(Q|D) = \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

$$= \sum_{i:f_{q_i,D}>0} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}) + \sum_{i:f_{q_i,D}=0} \log(\lambda\frac{c_{q_i}}{|C|})$$

$$= \sum_{i:f_{q_i,D}>0} \log \frac{((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})}{\lambda\frac{c_{q_i}}{|C|}} + \sum_{i=1}^{n} \log(\lambda\frac{c_{q_i}}{|C|})$$

$$\overset{rank}{=} \sum_{i:f_{q_i,D}>0} \log\left(\frac{((1-\lambda)\frac{f_{q_i,D}}{|D|}}{\lambda\frac{c_{q_i}}{|C|}} + 1\right)$$

- proportional to the term frequency, inversely proportional to the collection frequency

Dirichlet Smoothing

- α_D depends on document length

$$\alpha_D = \frac{\mu}{|D|+\mu}$$

- Gives probability estimation of

$$p(q_i|D) = \frac{f_{q_i,D}+\mu\frac{c_{q_i}}{|C|}}{|D|+\mu}$$

- and document score

$$\log P(Q|D) = \sum_{i=1}^{n} \log \frac{f_{q_i,D}+\mu\frac{c_{q_i}}{|C|}}{|D|+\mu}$$

Query Likelihood Example

- For the term "president"
  - $f_{qi,D} = 15$, $c_{qi} = 160{,}000$
- For the term "lincoln"
  - $f_{qi,D} = 25$, $c_{qi} = 2{,}400$
- number of word occurrences in the document $|d|$ is assumed to be 1,800
- number of word occurrences in the collection is 10^9
  - 500,000 documents times an average of 2,000 words
- $\mu = 2{,}000$

Query Likelihood Example

$$
\begin{aligned}
QL(Q,D) &= \log \frac{15 + 2000 \times (1.6 \times 10^5 / 10^9)}{1800 + 2000} \\
&\quad + \log \frac{25 + 2000 \times (2400/10^9)}{1800 + 2000} \\
&= \log(15.32/3800) + \log(25.005/3800) \\
&= -5.51 + -5.02 = -10.53
\end{aligned}
$$

- Negative number because summing logs of small numbers

Query Likelihood Example

Frequency of "president"	Frequency of "lincoln"	QL score
15	25	-10.53
15	1	-13.75
15	0	-19.05
1	25	-12.99
0	25	-14.40

Relevance Models

- *Relevance model* – language model representing information need
  - query and relevant documents are samples from this model
- $P(D|R)$ - probability of generating the text in a document given a relevance model
  - *document likelihood* model
  - less effective than query likelihood due to difficulties comparing across documents of different lengths

Backup slides

Galago Query Language

- A document is viewed as a sequence of text that may contain arbitrary tags
- A single *context* is generated for each unique tag name
- An *extent* is a sequence of text that appears within a single begin/end tag pair of the same type as the context

Galago Query Language

```
<html>
<head>
<title>Department Descriptions</title>
</head>
<body>
The following list describes ...
<h1>Agriculture</h1> ...
<h1>Chemistry</h1> ...
<h1>Computer Science</h1> ...
<h1>Electrical Engineering</h1> ...
</body>
</html>
```

title context:

```
<title>Department Descriptions</title>
```

h1 context:

```
<h1>Agriculture</h1>
<h1>Chemistry</h1> ...
<h1>Computer Science</h1> ...
<h1>Electrical Engineering</h1> ...
```

body context:

```
<body> The following list describes ...
<h1>Agriculture</h1> ...
<h1>Chemistry</h1> ...
<h1>Computer Science</h1> ...
<h1>Electrical Engineering</h1> ...
</body>
```

Galago Query Language

Simple terms:

term – term that will be normalized and stemmed.
"term" – term is not normalized or stemmed.
Examples:
presidents
"NASA"

Galago Query Language

Proximity terms:

#od:N(...) – ordered window – terms must appear ordered, with at most N-1 terms between each.
#od(...) – unlimited ordered window – all terms must appear ordered anywhere within current context.
#uw:N(...) – unordered window – all terms must appear within a window of length N in any order.
#uw(...) – unlimited unordered window – all terms must appear within current context in any order.
Examples:
#od:1(white house) – matches "white house" as an exact phrase.
#od:2(white house) – matches "white * house" (where * is any word or null).
#uw:2(white house) – matches "white house" and "house white".

Galago Query Language

Synonyms:

#syn(...)
#wsyn(...)
Examples:
#syn(dog canine) – simple synonym based on two terms.
#syn(#od:1(united states) #od:1(united states of america)) – creates a synonym from two proximity terms.
#wsyn(1.0 donald 0.8 don 0.5 donnie) – weighted synonym indicating relative importance of terms.

Galago Query Language

Anonymous terms:

#any:.() – used to match extent types
Examples:
#any:person() – matches any occurrence of a person extent.
#od:1(lincoln died in #any:date()) – matches exact phrases of the form:"lincoln died in <date>...</date>".

Galago Query Language

Context restriction and evaluation:

expression.C1,,...,CN – matches when the expression appears in all contexts C1 through CN.
expression.(C1,...,CN) – evaluates the expression using the language model defined by the concatenation of contexts C1...CN within the document.
Examples:
dog.title – matches the term "dog" appearing in a title extent.
#uw(smith jones).author – matches when the two names "smith" and "jones" appear in an author extent.
dog.(title) – evaluates the term based on the title language model for the document.
#od:1(abraham lincoln).person.(header) – builds a language model from all of the "header" text in the document and evaluates #od:1(abraham lincoln).person in that context (i.e., matches only the exact phrase appearing within a person extent within the header context).

Galago Query Language

Belief operators:

#combine(...) – this operator is a normalized version of the $bel_{and}(q)$ operator in the inference network model. See the discussion below for more details.
#weight(...) – this is a normalized version of the $bel_{wand}(q)$ operator.
#filter(...) – this operator is similar to #combine, but with the difference that the document must contain at least one instance of all terms (simple, proximity, synonym, etc.). The evaluation of nested belief operators is not changed.

Galago Query Language

Examples:

#combine(#syn(dog canine) training) – rank by two terms, one of which is a synonym.

#combine(biography #syn(#od:1(president lincoln) #od:1(abraham lincoln))) – rank using two terms, one of which is a synonym of "president lincoln" and "abraham lincoln".

#weight(1.0 #od:1(civil war) 3.0 lincoln 2.0 speech) – rank using three terms, and weight the term "lincoln" as most important, followed by "speech", then "civil war".

#filter(aquarium #combine(tropical fish)) – consider only those documents containing the word "aquarium" and "tropical" or "fish", and rank them according to the query #combine(aquarium #combine(tropical fish)).

#filter(#od:1(john smith).author) #weight(2.0 europe 1.0 travel) – rank documents about "europe" or "travel" that have "John Smith" in the author context.

Web Search

- Most important, but not only, search application
- Major differences to TREC news
  - Size of collection
  - Connections between documents
  - Range of document types
  - Importance of spam
  - Volume of queries
  - Range of query types

Search Taxonomy

- *Informational*
  - Finding information about some topic which may be on one or more web pages
  - Topical search
- *Navigational*
  - finding a particular web page that the user has either seen before or is assumed to exist
- *Transactional*
  - finding a site where a task such as shopping or downloading music can be performed

Web Search

- For effective navigational and transactional search, need to combine features that reflect *user relevance*
- Commercial web search engines combine evidence from *hundreds* of features to generate a ranking score for a web page
  - page content, page metadata, anchor text, links (e.g., PageRank), and user behavior (click logs)
  - page metadata – e.g., "age", how often it is updated, the URL of the page, the domain name of its site, and the amount of text content

Search Engine Optimization

- *SEO*: understanding the relative importance of features used in search and how they can be manipulated to obtain better search rankings for a web page
  - e.g., improve the text used in the title tag, improve the text in heading tags, make sure that the domain name and URL contain important keywords, and try to improve the anchor text and link structure
  - Some of these techniques are regarded as not appropriate by search engine companies

Web Search

- In TREC evaluations, most effective features for navigational search are:
  - text in the title, body, and heading (h1, h2, h3, and h4) parts of the document, the anchor text of all links pointing to the document, the PageRank number, and the inlink count
- Given size of Web, many pages will contain all query terms
  - Ranking algorithm focuses on discriminating between these pages
  - Word proximity is important

Term Proximity

- Many models have been developed
- N-grams are commonly used in commercial web search
- *Dependence model* based on inference net has been effective in TREC - e.g.

```
#weight(
        0.8 #combine(embryonic stem cells)
        0.1 #combine( #od:1(stem cells) #od:1(embryonic stem)
                      #od:1(embryonic stem cells))
        0.1 #combine( #uw:8(stem cells) #uw:8(embryonic cells)
                      #uw:8(embryonic stem) #uw:12(embryonic stem cells)))
```

Example Web Query

```
#weight(
        0.1 #weight( 0.6 #prior(pagerank) 0.4 #prior(inlinks))
        1.0 #weight(
            0.9 #combine(
                #weight( 1.0 pet.(anchor) 1.0 pet.(title)
                         3.0 pet.(body) 1.0 pet.(heading))
                #weight( 1.0 therapy.(anchor) 1.0 therapy.(title)
                         3.0 therapy.(body) 1.0 therapy.(heading)))
            0.1 #weight(
                1.0 #od:1(pet therapy).(anchor) 1.0 #od:1(pet therapy).(title)
                3.0 #od:1(pet therapy).(body) 1.0 #od:1(pet therapy).(heading))
            0.1 #weight(
                1.0 #uw:8(pet therapy).(anchor) 1.0 #uw:8(pet therapy).(title)
                3.0 #uw:8(pet therapy).(body) 1.0 #uw:8(pet therapy).(heading)))
        )
```

Machine Learning and IR

- Considerable interaction between these fields
  - Rocchio algorithm (60s) is a simple learning approach
  - 80s, 90s: learning ranking algorithms based on user feedback
  - 2000s: text categorization
- Limited by amount of training data
- Web query logs have generated new wave of research
  - e.g., “Learning to Rank”

Generative vs. Discriminative

- All of the probabilistic retrieval models presented so far fall into the category of *generative models*
  - A generative model assumes that documents were generated from some underlying model (in this case, usually a multinomial distribution) and uses training data to estimate the parameters of the model
  - probability of belonging to a class (i.e. the relevant documents for a query) is then estimated using Bayes' Rule and the document model

Generative vs. Discriminative

- A *discriminative* model estimates the probability of belonging to a class directly from the observed features of the document based on the training data
- Generative models perform well with low numbers of training examples
- Discriminative models usually have the advantage given enough training data
  - Can also easily incorporate many features

Discriminative Models for IR

- Discriminative models can be trained using explicit relevance judgments or click data in query logs
  - Click data is much cheaper, more noisy
  - e.g. Ranking Support Vector Machine (SVM) takes as input *partial rank* information for queries
    - partial information about which documents should be ranked higher than others

Ranking SVM

- Training data is

$$(q_1, r_1), (q_2, r_2), \ldots, (q_n, r_n)$$

  - r is partial rank information
    - if document d_a should be ranked higher than d_b, then (d_a, d_b) $\in r_i$
  - partial rank information comes from relevance judgments (allows multiple levels of relevance) or click data
    - e.g., d_1, d_2 and d_3 are the documents in the first, second and third rank of the search output, only d_3 clicked on $\rightarrow$ (d_3, d_1) and (d_3, d_2) will be in desired ranking for this query

Ranking SVM

- Learning a linear ranking function $\vec{w}.\vec{d}_a$
  - where *w* is a weight vector that is adjusted by learning
  - d_a is the vector representation of the features of document
  - *non-linear* functions also possible
- Weights represent importance of features
  - learned using training data
  - e.g.,

$$\vec{w}.\vec{d} = (2,1,2).(2,4,1) = 2.2 + 1.4 + 2.1 = 10$$

Ranking SVM

- Learn *w* that satisfies as many of the following conditions as possible:

$$\forall (d_i, d_j) \in r_1 \quad : \quad \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j}$$

$$\ldots$$

$$\forall (d_i, d_j) \in r_n \quad : \quad \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j}$$

- Can be formulated as an optimization problem

Ranking SVM

$$
\begin{aligned}
minimize: \quad & \frac{1}{2}\vec{w}.\vec{w} + C\sum \xi_{i,j,k} \\
subject\ to: \quad & \\
\forall (d_i, d_j) \in r_1 \quad : \quad & \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} + 1 - \xi_{i,j,1} \\
& \dots \\
\forall (d_i, d_j) \in r_n \quad : \quad & \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} + 1 - \xi_{i,j,n} \\
& \forall i \forall j \forall k : \xi_i, j, k \geq 0
\end{aligned}
$$

- ξ, known as a slack variable, allows for misclassification of difficult or noisy training examples, and C is a parameter that is used to prevent overfitting

Ranking SVM

- Software available to do optimization
- Each pair of documents in our training data can be represented by the vector:

$$(\vec{d}_i - \vec{d}_j)$$

- Score for this pair is:

$$\vec{w}.(\vec{d}_i - \vec{d}_j)$$

- SVM classifier will find a *w* that makes the smallest score as large as possible
  - make the differences in scores as large as possible for the pairs of documents that are hardest to rank

Topic Models

- Improved representations of documents
  - can also be viewed as improved smoothing techniques
  - improve estimates for words that are related to the topic(s) of the document
    - instead of just using background probabilities
- Approaches
  - *Latent* Semantic Indexing (LSI)
  - Probabilistic *Latent* Semantic Indexing (pLSI)
  - *Latent* Dirichlet Allocation (LDA)

LDA

- Model document as being generated from a *mixture* of topics

1. For each document D, pick a multinomial distribution θ_D from a Dirichlet distribution with parameter α ,
2. For each word position in document D,
   (a) pick a topic z from the multinomial distribution θ_D ,
   (b) Choose a word w from $P(w|z, \beta)$, a multinomial probability conditioned on the topic z with parameter β.

LDA

- Gives language model probabilities

$$P_{lda}(w|D) = P(w|\theta_D, \beta) = \sum_z P(w|z, \beta)P(z|\theta_D)$$

- Used to smooth the document representation by mixing them with the query likelihood probability as follows:

$$P(w|D) = \lambda \left(\frac{f_{w,D} + \mu \frac{c_w}{|C|}}{|D| + \mu} \right) + (1 - \lambda) P_{lda}(w|D)$$

LDA

- If the LDA probabilities are used directly as the document representation, the effectiveness will be significantly reduced because the features are *too smoothed*
  - e.g., in typical TREC experiment, only 400 topics used for the *entire* collection
  - generating LDA topics is expensive
- When used for smoothing, effectiveness is improved

LDA Example

- Top words from 4 LDA topics from TREC news

Arts	*Budgets*	*Children*	*Education*
new	million	children	school
film	tax	women	students
show	program	people	schools
music	budget	child	education
movie	billion	years	teachers
play	federal	families	high
musical	year	work	public
best	spending	parents	teacher
actor	new	says	bennett
first	state	family	manigat
york	plan	welfare	namphy
opera	money	men	state
theater	programs	percent	president
actress	government	care	elementary
love	congress	life	haiti

Summary

- Best retrieval model depends on application and data available
- Evaluation corpus (or test collection), training data, and user data are all critical resources
- Open source search engines can be used to find effective ranking algorithms
  - Galago query language makes this particularly easy
- Language resources (e.g., thesaurus) can make a big difference