IIIT-H

Web Mining
Lecture 2: Relevance Ranking

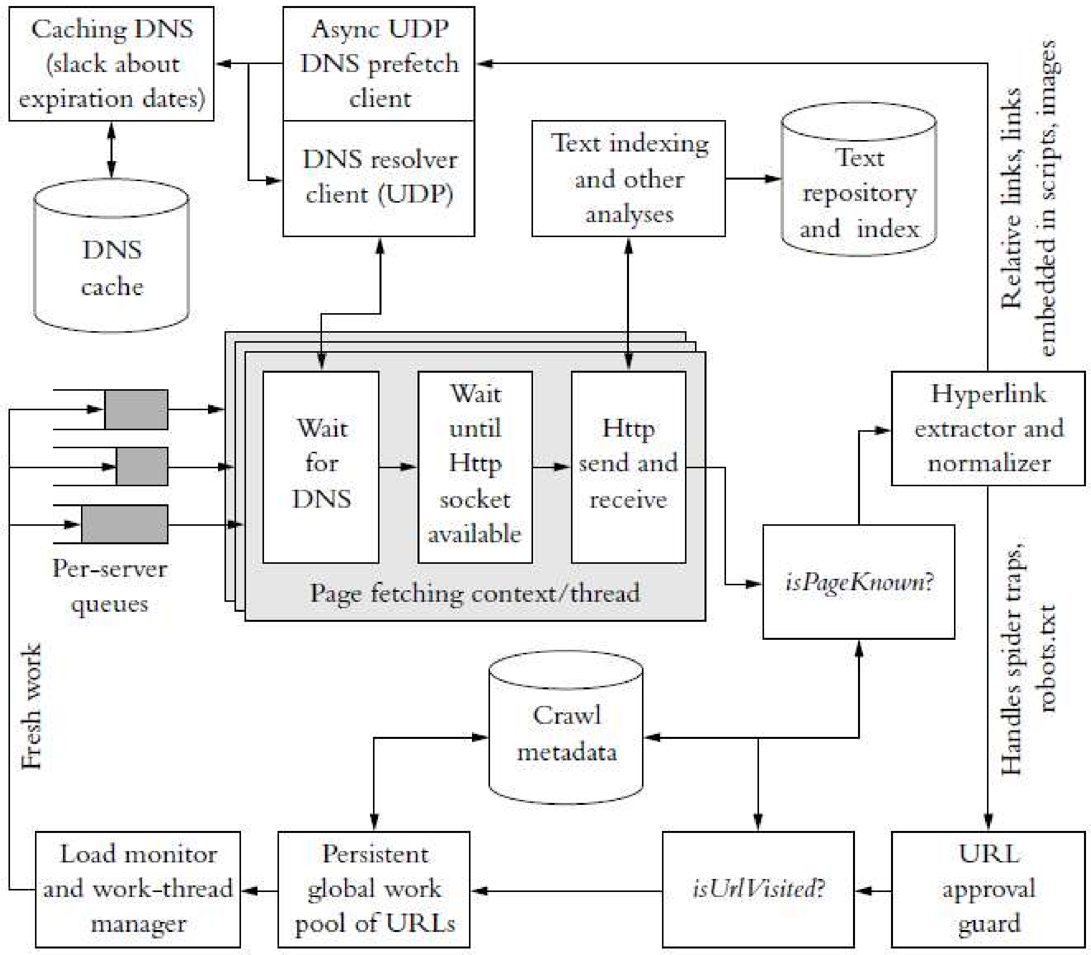
Manish Gupta

3rd Aug 2013

Slides borrowed (and modified) from
http://www.stanford.edu/class/cs276/handouts/lecture6-tfidf.ppt
http://www.stanford.edu/class/cs276/handouts/lecture7-vectorspace.ppt
http://www.stanford.edu/class/cs276/handouts/lecture8-evaluation.ppt

Recap of Lecture 1: Text Indexing and Crawling

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2


Caching DNS (slack about expiration dates)
Async UDP DNS prefetch client
DNS resolver client (UDP)
DNS cache
Text indexing and other analyses
Text repository and index
Relative links, links embedded in scripts, images
Hyperlink extractor and normalizer
Wait for DNS
Wait until Http socket available
Http send and receive
Page fetching context/thread
Per-server queues
isPageKnown?
Fresh work
Crawl metadata
Handles spider traps, robots.txt
Load monitor and work-thread manager
Persistent global work pool of URLs
isUrlVisited?
URL approval guard

Today’s Agenda

- Need for Relevance Ranking
- TF and IDF
- Vector Space Model
- Evaluation Metrics for Ranking

Today's Agenda

- **Need for Relevance Ranking**
- TF and IDF
- Vector Space Model
- Evaluation Metrics for Ranking

Ranked Retrieval

- Thus far, our queries have all been Boolean.
  - Documents either match or don’t.
- Good for expert users with precise understanding of their needs and the collection.
  - Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users.
  - Most users are incapable of writing Boolean queries.
  - Most users don’t want to wade through 1000s of results.
    - This is particularly true of web search.

Problem with Boolean search: feast or famine

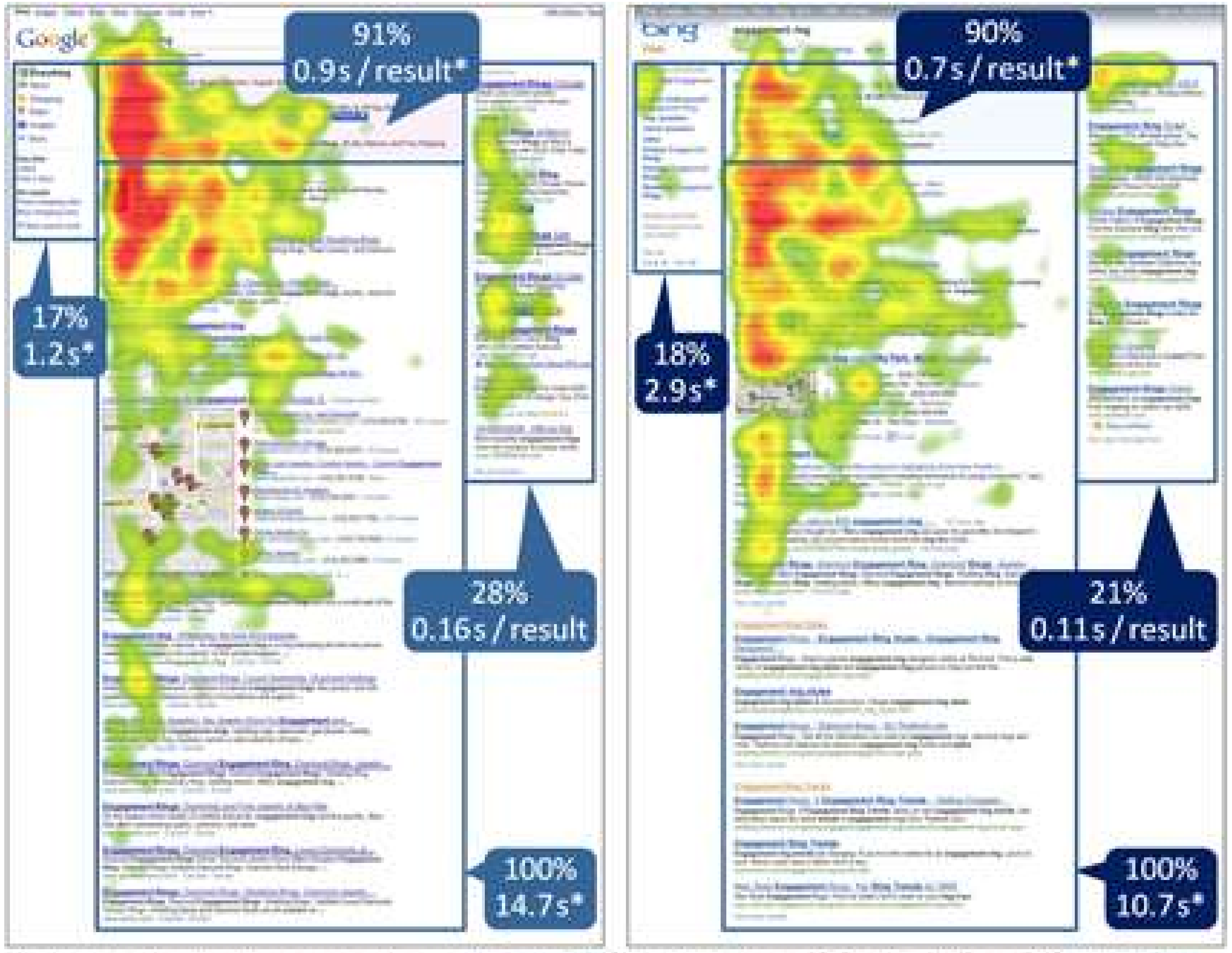
- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1: "*standard user dlink 650*" → 200,000 hits
- Query 2: "*standard user dlink 650 no card found*": 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits.
  - AND gives too few; OR gives too many

Feast or Famine: Not a Problem in Ranked Retrieval

- Rather than a set of documents satisfying a query expression, in ranked retrieval, the system returns an ordering over the (top) documents in the collection for a query
- When a system produces a ranked result set, large result sets are not an issue
  - Indeed, the size of the result set is not an issue
  - We just show the top *k* (≈ 10) results
  - We don't overwhelm the user
  - Premise: the ranking algorithm works. Is it true?

Eye Tracking Study on Search Results

Google→



←Bing

Heatmaps showing the aggregate gaze time of all 24 participants on Google (left) and Bing (right) for one of the transactional tasks. The red color indicates areas that received the most total gaze time (4.5 seconds and above). Each callout includes the percentage of participants who looked at the area and the time (in seconds) they spent looking there. The numerical data are an average across all four tasks. Asterisks indicate values that were significantly different between Google and Bing at alpha = .1.

http://blog.mediative.com/en/2011/08/31/eye-tracking-google-through-the-years/

Scoring as the Basis of Ranked Retrieval

- We wish to return in order the documents most likely to be useful to the searcher
- How can we rank-order the documents in the collection with respect to a query?
- Assign a score – say in [0, 1] – to each document
- This score measures how well document and query "match".

Query-Document Matching Scores

- We need a way of assigning a score to a query/document pair
- Let's start with a one-term query
- If the query term does not occur in the document: score should be 0
- The more frequent the query term in the document, the higher the score (should be)
- We will look at a number of alternatives for this.
- First take: Jaccard coefficient?

Issues with Jaccard for Scoring

- It doesn't consider *term frequency* (how many times a term occurs in a document)
- Rare terms in a collection are more informative than frequent terms. Jaccard doesn't consider this information
- We need a more sophisticated way of normalizing for length
- Later in this lecture, we'll use $|A \cap B| / \sqrt{|A \cup B|}$
- . . . instead of $|A \cap B|/|A \cup B|$ (Jaccard) for length normalization.

Recall (Lecture 1): Binary Term-Document Incidence Matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

Term-Document Count Matrices

- Consider the number of occurrences of a term in a document
  - Each document is a count vector in $\mathbb{N}^{v}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Today's Agenda

- Need for Relevance Ranking
- **TF and IDF**
- Vector Space Model
- Evaluation Metrics for Ranking

Term Frequency TF

- The term frequency tf$_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- Relevance does not increase proportionally with term frequency.
- So use log frequency weighting
- The log frequency weight of term t in d is $w_{td} = 1 + \log_{10} tf_{td}$ if $tf_{td} > 0$
- Score for a document-query pair: sum over terms *t* in both *q* and *d*
  - $score = \sum_{t \in q \cap d} (1 + \log_{10} tf_{td})$

Inverse Document Frequency IDF

- Frequent terms are less informative than rare terms
- df_t is the document frequency of t: the number of documents that contain t
  - $df_t \leq N$
- $idf_t = \log_{10}\left(\frac{N}{df_t}\right)$
- IDF has no effect on ranking one term queries
  - IDF affects the ranking of documents for queries with at least two terms
  - For the query capricious person, IDF weighting makes occurrences of capricious count for much more in the final document ranking than occurrences of person.

TD-IDF Weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = \log(1 + \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Score for a document given a query

$$\mathrm{Score}(q,d) = \sum_{t \in q \cap d} \mathrm{tf.idf}_{t,d}$$

- There are many variants
  – How “tf” is computed (with/without logs)
  – Whether the terms in the query are also weighted
  – …

Binary → Count → Weight Matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$

Ranking in Vector Space Model

- Represent both query and document as vectors in the |V|-dimensional space
- Use cosine similarity as the similarity measure
  - Incorporates length normalization automatically (longer vs shorter documents)

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \bullet \frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- *q_i* is the tf-idf weight of term *i* in the query
- *d_i* is the tf-idf weight of term *i* in the document

Computing Cosine Scores

```
CosineScore($q$)
float $Scores[N] = 0$
float $Length[N]$
for each query term $t$
do calculate $w_{t,q}$ and fetch postings list for $t$
    for each pair($d$, $\mathrm{tf}_{t,d}$) in postings list
    do $Scores[d] += w_{t,d} \times w_{t,q}$
Read the array $Length$
for each $d$
do $Scores[d] = Scores[d]/Length[d]$
return Top $K$ components of $Scores[]$
```

TF-IDF Weighting has many Variants

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Columns headed 'n' are acronyms for weight schemes.
SMART Notation: denotes the combination in use in an engine, with the notation *ddd.qqq*, using the acronyms from the previous table
A very standard weighting scheme is: lnc.ltc

Okapi BM25

- Given a query Q, containing keywords $q_1, \ldots, q_n$, the BM25 score of a document D is:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) . \frac{tf_{q_i,D} . (k_1 + 1)}{tf_{q_i,D} + k_1 . \left(1 - b + b . \frac{|D|}{avgdl}\right)}$$

- |D| is the length of the document D in words
- avgdl is the average document length in the text collection from which documents are drawn
- k_1 and b are free parameters, usually chosen, in absence of an advanced optimization, as $k_1 \in [1.2,2.0]$ and b = 0.75.
- $IDF(q_i) = log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$
  - N = total number of documents in the collection
  - $N(q_i)$ = documents containing q_i

Summary – Vector Space Ranking

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top *K* (e.g., *K* = 10) to the user

Today's Agenda

- Need for Relevance Ranking
- TF and IDF
- **Vector Space Model**
- Evaluation Metrics for Ranking

Efficient Cosine Ranking

- Find the K docs in the collection "nearest" to the query $\Rightarrow$ K largest query-doc cosines.
- Efficient ranking
  - Computing a single cosine efficiently.
  - Choosing the K largest cosine values efficiently.
    - Can we do this without computing all N cosines?
    - We don't need to totally order all docs in the collection
    - Let J = number of docs with nonzero cosines
      - We seek the K best of these J
    - Use heap for selecting top K
    - Exact topK is difficult, but approx-topK is feasible and acceptable

Generic Approach

- Find a set A of *contenders*, with $K < |A| << N$
  - A does not necessarily contain the top K, but has many docs from among the top K
  - Return the top K docs in A
- Think of A as <u>pruning</u> non-contenders

Index Elimination

- Basic algorithm only considers docs containing at least one query term
- Take this further
  - Only consider high-idf query terms
  - Only consider docs containing many query terms

Champion Lists

- Precompute for each dictionary term *t*, the *r* docs of highest weight in *t*'s postings
  - Call this the champion list for *t*
  - (aka fancy list or top docs for *t*)
- Note that *r* has to be chosen at index time
- At query time, only compute scores for docs in the champion list of some query term
  - Pick the *K* top-scoring docs from amongst these

Static Quality Scores

- We want top-ranking documents to be both *relevant* and *authoritative*
- *Relevance* is being modeled by cosine scores
- *Authority* is typically a query-independent property of a document
- Examples of authority signals
  - Wikipedia among websites
  - Articles in certain newspapers
  - A paper with many citations
  - Many diggs, Y!buzzes or del.icio.us marks
  - (Pagerank)
- Assign to each document a *query-independent* quality score in [0,1] to each document *d*
  - Denote this by *g(d)*

Net Score

- Consider a simple total score combining cosine relevance and authority
- net-score(q,d) = $g(d) + $ cosine(q,d)
  - Can use some other linear combination than an equal weighting
  - Indeed, any function of the two "signals" of user happiness
  - Now we seek the top *K* docs by net score

Top *K* by Net Score – Fast Methods

- First idea: Order all postings by *g(d)*
- Why? Under g(d)-ordering, top-scoring docs likely to appear early in postings traversal
- Key: this is a common ordering for all postings
- Thus, can concurrently traverse query terms' postings for
  - Postings intersection
  - Cosine score computation

Champion Lists in *g(d)*-Ordering

- Can combine champion lists with *g(d)*-ordering
- Maintain for each term a champion list of the *r* docs with highest $g(d) + \text{tf-idf}_{td}$
- Seek top-*K* results from only the docs in these champion lists

High and Low Lists

- For each term, we maintain two postings lists called *high* and *low*
  - Think of *high* as the champion list
- When traversing postings on a query, only traverse *high* lists first
  - If we get more than *K* docs, select the top *K* and stop
  - Else proceed to get docs from the *low* lists
- Can be used even for simple cosine scores, without global quality *g(d)*
- A means for segmenting index into two tiers

Impact-Ordered Postings

- We only want to compute scores for docs for which $tf_{t,d}$ is high enough
- We sort each postings list by $tf_{t,d}$
- Now: not all postings in a common order!
- How do we compute scores in order to pick off top *K*?
  - Early termination
  - idf-ordered query terms

1. Early Termination

- When traversing *t*'s postings, stop early after either
  - a fixed number of *r* docs
  - $tf_{t,d}$ drops below some threshold
- Take the union of the resulting sets of docs
  - One from the postings of each query term
- Compute only the scores for docs in this union

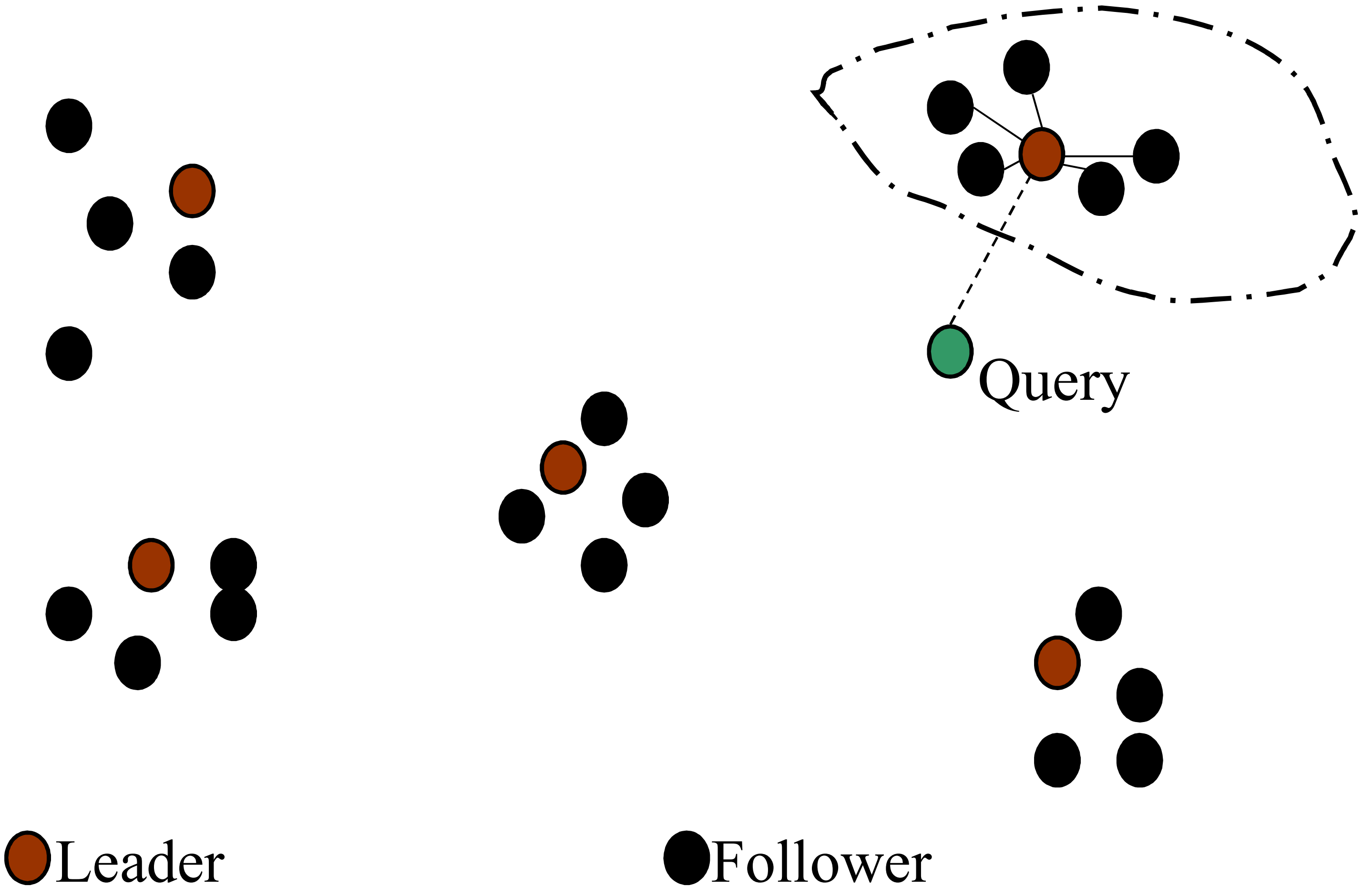
2. IDF-Ordered Query Terms

- When considering the postings of query terms (outer loop)
- Look at the query terms in order of decreasing idf
    - High idf terms likely to contribute most to score
- As we update score contribution from each query term
    - Stop if doc scores relatively unchanged
- Can apply to cosine or some other net scores

Cluster Pruning

- Pick √N *docs* at random: call these *leaders*
- For every other doc, pre-compute nearest leader
  - Docs attached to a leader: its *followers*
  - Likely: each leader has ~ √*N* followers
- Can recur on leader/follower construction
- Process a query as follows
  - Given query *Q*, find its nearest *leader L*
  - Seek *K* nearest docs from among *L*'s followers

Visualization



Parametric and Zone Indexes

- Thus far, a doc has been a sequence of terms
- In fact documents have multiple parts, some with special semantics:
  - Author
  - Title
  - Date of publication
  - Language
  - Format
  - etc.
- These constitute the metadata about a document

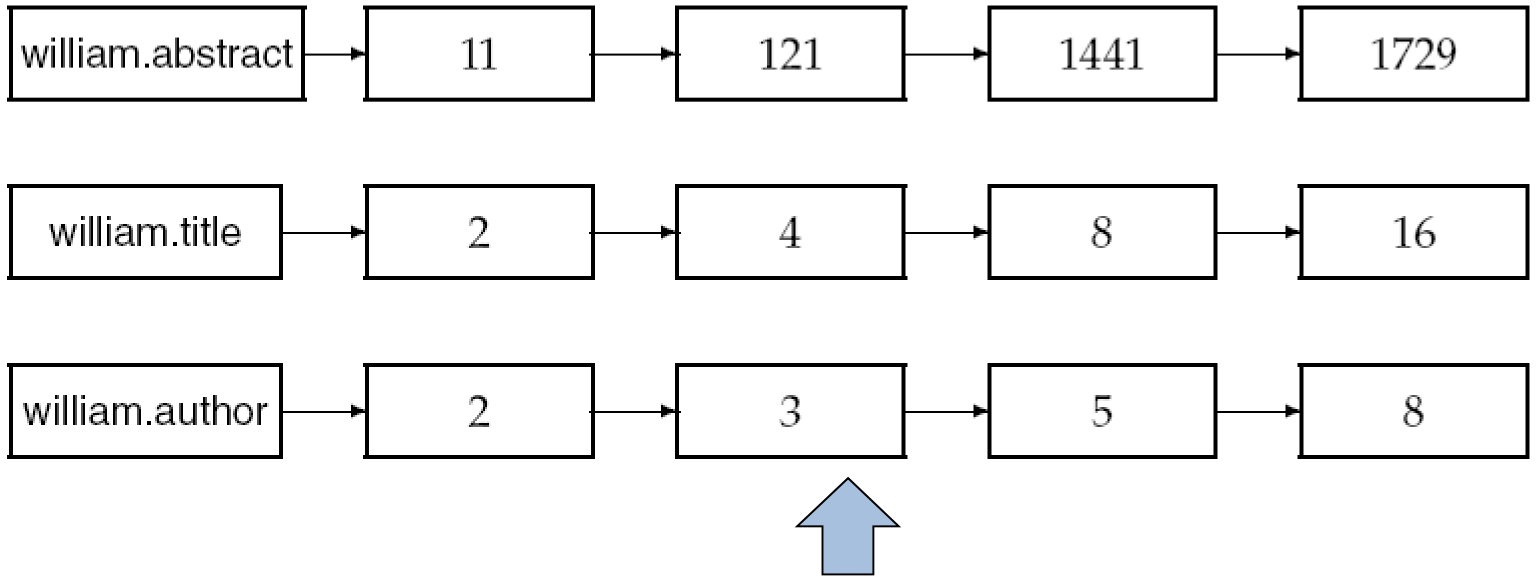
Fields

- We sometimes wish to search by these metadata
  - E.g., find docs authored by William Shakespeare in the year 1601, containing *alas poor Yorick*
- Year = 1601 is an example of a field
- Also, author last name = shakespeare, etc
- Field or parametric index: postings for each field value
  - Sometimes build range trees (e.g., for dates)
- Field query typically treated as conjunction
  - (doc *must* be authored by shakespeare)

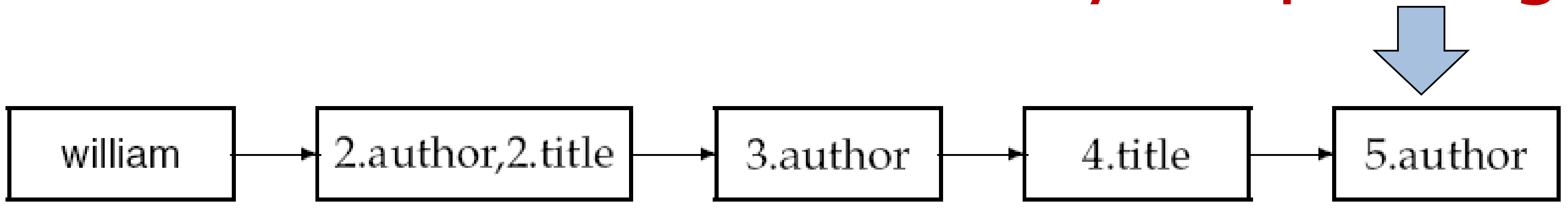
Zone

- A zone is a region of the doc that can contain an arbitrary amount of text e.g.,
  - Title
  - Abstract
  - References …
- Build inverted indexes on zones as well to permit querying
- E.g., “find docs with *merchant* in the title zone and matching the query *gentle rain*”

Example Zone Indexes



Encode zones in dictionary vs. postings.



Tiered Indexes

- Break postings up into a hierarchy of lists
  - Most important
  - …
  - Least important
- Can be done by *g(d)* or another measure
- Inverted index thus broken up into tiers of decreasing importance
- At query time use top tier unless it fails to yield *K* docs
  - If so drop to lower tiers

Query Term Proximity

- Free text queries: just a set of terms typed into the query box – common on the web
- Users prefer docs in which query terms occur within close proximity of each other
- Let *w* be the smallest window in a doc containing all query terms, e.g.,
- For the query *strained mercy* the smallest window in the doc *The quality of mercy is not strained* is 4 (words)
- Would like scoring function to take this into account – how?

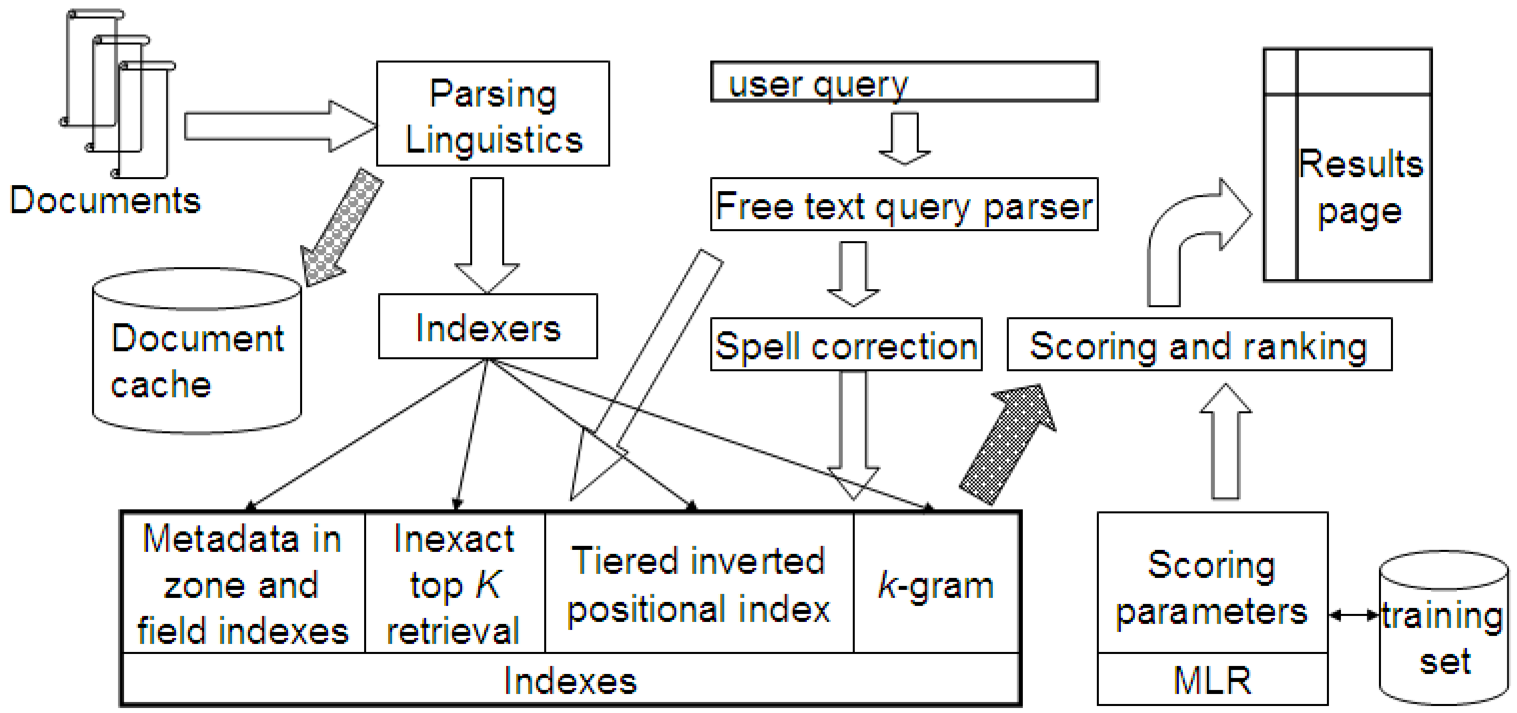
Query Parsers

- Free text query from user may in fact spawn one or more queries to the indexes, e.g. query *rising interest rates*
  - Run the query as a phrase query
  - If <*K* docs contain the phrase *rising interest rates*, run the two phrase queries *rising interest* and *interest rates*
  - If we still have <*K* docs, run the vector space query *rising interest rates*
  - Rank matching docs by vector space scoring
- This sequence is issued by a <u>query parser</u>

Aggregate Scores

- We’ve seen that score functions can combine cosine, static quality, proximity, etc.
- How do we know the best combination?
- Some applications – expert-tuned
- Increasingly common: machine-learned
  - Learning to rank framework
  - Each of the scores is fed as features
  - Weight vector is learned using available labeled relevance training data

Putting it all Together



Today's Agenda

- Need for Relevance Ranking
- TF and IDF
- Vector Space Model
- **Evaluation Metrics for Ranking**

Measures for a Search Engine

- How fast does it index
  - Number of documents/hour
- How fast does it search
  - Latency as a function of index size
- Expressiveness of query language
  - Ability to express complex information needs
  - Speed on complex queries
- Uncluttered UI
- Is it free?

Standard Relevance Benchmarks

- TREC - National Institute of Standards and Technology (NIST) has run a large IR test bed for many years
- Reuters and other benchmark doc collections used
- “Retrieval tasks” specified
  - sometimes as queries
- Human experts mark, for each query and for each doc, Relevant or Nonrelevant
  - or at least for subset of docs that some system returned for that query

50

Unranked Retrieval Evaluation: Precision and Recall

- **Precision**: fraction of retrieved docs that are relevant = P(relevant|retrieved)
- **Recall**: fraction of relevant docs that are retrieved = P(retrieved|relevant)

	Relevant	Nonrelevant
Retrieved	tp	fp
Not Retrieved	fn	tn

- Precision P = tp/(tp + fp)
- Recall R = tp/(tp + fn)

A Combined Measure: *F*

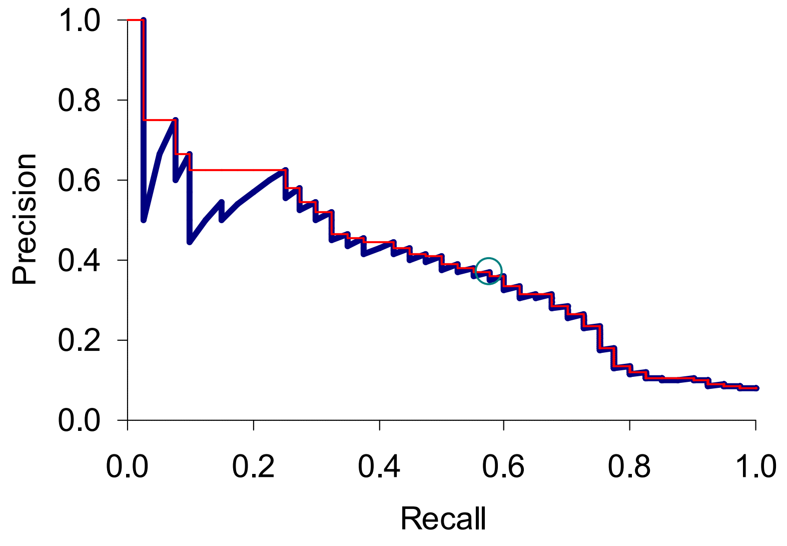
- Combined measure that assesses precision/recall tradeoff is **F measure** (weighted harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- People usually use balanced F_1 measure
  - i.e., with $\beta = 1$ or $\alpha = ½$
- Harmonic mean is always conservative
  - Less than arithmetic mean and geometric mean at all points

Evaluating Ranked Results

- Evaluation of ranked results
  - The system can return any number of results
  - By taking various numbers of the top returned documents (levels of recall), the evaluator can produce a *precision-recall curve*



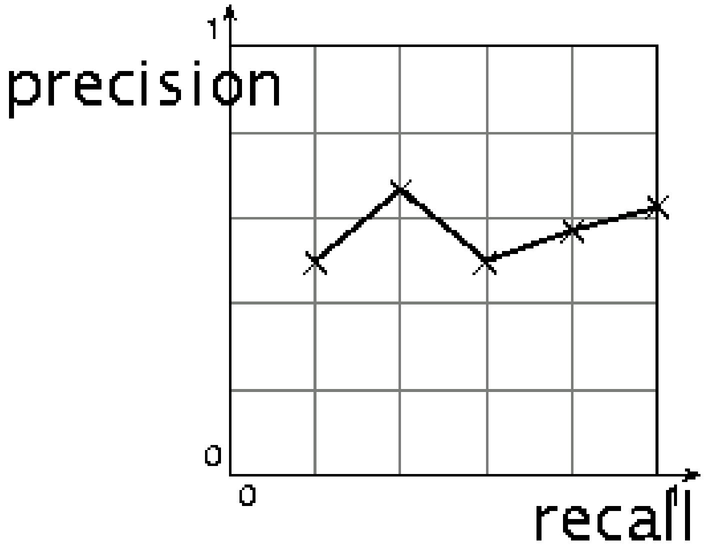
Averaging over Queries

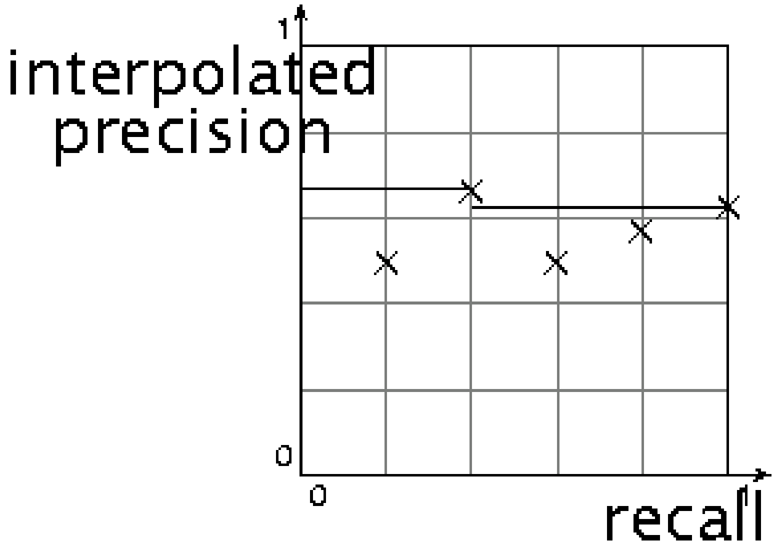
- A precision-recall graph for one query isn’t a very sensible thing to look at
- You need to average performance over a whole bunch of queries.
- But there’s a technical issue
  - Precision-recall calculations place some points on the graph
  - How do you determine a value (interpolate) between the points?

54

Interpolated Precision

- Idea: If locally precision increases with increasing recall, then you should get to count that…
- So you take the max of precisions to right of value





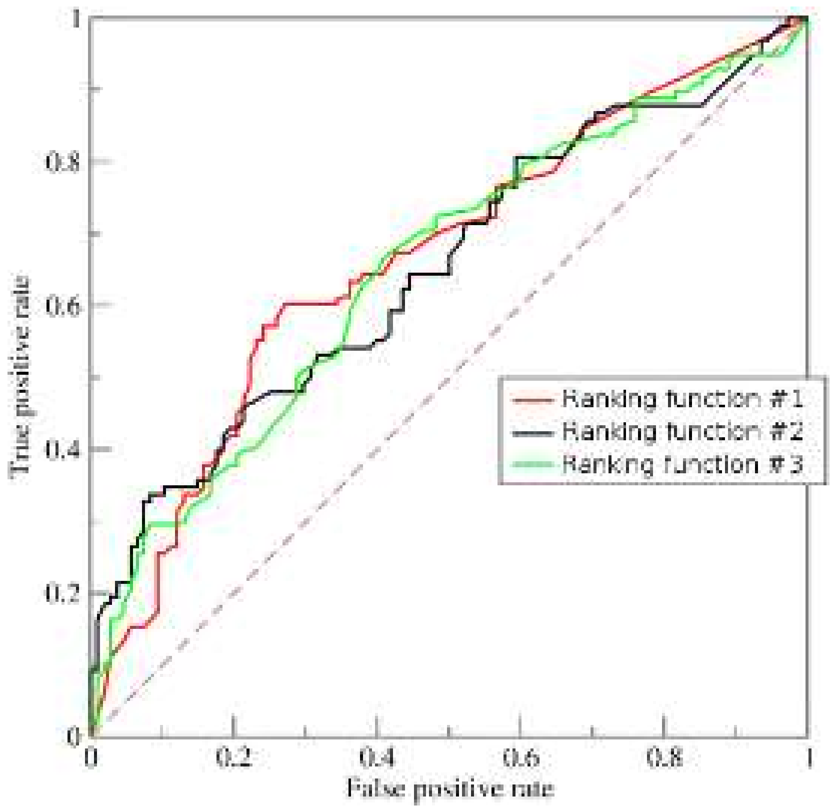
55

ROC Curve

- Shorthand: good=relevant, bad=irrelevant
- Fix query, suppose n^+ good, n^- bad docs
- Algorithm sorts all $(n^+ + n^-)$ docs
- For i = 0, 1, 2, …, $(n^+ + n^-)$
  - Suppose algorithm marks top i as relevant
  - Rest as irrelevant
  - In top i, say n_i^+ actually good, $n_i^- = i - n_i^+$ bad
- True positives@i=$\left(\frac{n_i^+}{n^+}\right)$
- False positives@i=$\left(\frac{n_i^-}{n^-}\right)$
- Plot y = true positive vs. x = false positive

Area under ROC Curve (AUC)

- ROC = Receiver Operating Characteristic curve
- Suppose ranking is random
  - Will get roughly diagonal line
- Good ranking functions
  - Will get very high true positive at very small false positive rates
  - Large lift above diagonal
- Measure area under the curve
  - ½ ⇒ effectively random
  - Close to 1 ⇒ good ranking algorithm



Other Evaluation Measures

- Precision at fixed retrieval level
  - Precision-at-*k*: Precision of top *k* results
- 11-point interpolated average precision
  - The standard measure in the early TREC competitions: take the precision at 11 levels of recall varying from 0 to 1 by tenths of the documents, using interpolation (the value for 0 is always interpolated!), and average them
- Mean average precision (MAP)
  - Precision at rank p_i is (i/p_i) and $1 \leq p_1 < p_2 < \cdots < p_R$
  - Then $MAP = \frac{1}{R}\sum_{i=1}^{R} \frac{i}{p_i}$
  - Avoids interpolation, use of fixed recall levels
  - MAP for query collection is arithmetic average
- R-precision
  - If we have a known set of relevant documents of size *Rel,* then calculate precision of the top *Rel* docs returned
  - Perfect system could score 1.0.

58

Normalized Discounted Cumulative Gain

- Fix query q
- Relevance level of document ranked j wrt q be $r_q(j)$
- $r_q(j)=0$ means totally irrelevant
- Response list is inspected up to rank L
- Discounted cumulative gain for query q is

$$\mathrm{NDCG}_q = \underbrace{Z_q}_{\text{normalized}} \overbrace{\sum_{j=1}^{L}}^{\text{cumulative}} \frac{\overbrace{2^{r_q(j)} - 1}^{\text{gain}}}{\underbrace{\log(1+j)}_{\text{rank discount}}}$$

- Z_q is a normalization factor that ensures the perfect ordering has NDCG$_q$ = 1
- Overall NDCG is average of NDCG$_q$ over all q
- No notion of recall, only precision at low ranks

Kappa Measure for Inter-Judge (Dis)agreement

- Kappa measure
  - Agreement measure among judges
  - Designed for categorical judgments
  - Corrects for chance agreement
- Kappa = [P(A) – P(E)] / [1 – P(E)]
- P(A) – proportion of time judges agree
- P(E) – what agreement would be by chance
- Kappa = 0 for chance agreement, 1 for total agreement.

P(A)? P(E)?

Kappa Measure: Example

Number of docs	Judge 1	Judge 2
300	Relevant	Relevant
70	Nonrelevant	Nonrelevant
20	Relevant	Nonrelevant
10	Nonrelevant	Relevant

61

Kappa Example

- P(A) = 370/400 = 0.925
- P(nonrelevant) = (10+20+70+70)/800 = 0.2125
- P(relevant) = (10+20+300+300)/800 = 0.7878
- P(E) = 0.2125^2 + 0.7878^2 = 0.665
- Kappa = (0.925 – 0.665)/(1-0.665) = 0.776

- Kappa > 0.8 = good agreement
- 0.67 < Kappa < 0.8 -> “tentative conclusions”
- Depends on purpose of study
- For >2 judges: average pairwise kappas

62

Critique of Pure Relevance

- Relevance vs Marginal Relevance
  - A document can be redundant even if it is highly relevant
  - Duplicates
  - The same information from different sources
  - Marginal relevance is a better measure of utility for the user.
- Using facts/entities as evaluation units more directly measures true relevance
- Diversity of search results?
- Other factors: Freshness? Trustworthiness?

Evaluation at Large Search Engines

- Search engines have test collections of queries and hand-ranked results
- Recall is difficult to measure on the web
- Search engines often use precision at top k, e.g., k = 10
- . . . or measures that reward you more for getting rank 1 right than for getting rank 10 right.
  - NDCG (Normalized Cumulative Discounted Gain)
- Search engines also use non-relevance-based measures.
  - Clickthrough on first result
    - Not very reliable if you look at a single clickthrough … but pretty reliable in the aggregate.
  - Studies of user behavior in the lab
  - A/B testing

64

A/B Testing

- Purpose: Test a single innovation
- Prerequisite: You have a large search engine up and running.
- Have most users use old system
- Divert a small proportion of traffic (e.g., 1%) to the new system that includes the innovation
- Evaluate with an “automatic” measure like clickthrough on first result
- Now we can directly see if the innovation does improve user happiness.
- Probably the evaluation methodology that large search engines trust most
- In principle less powerful than doing a multivariate regression analysis, but easier to understand

65

Summaries/Snippets

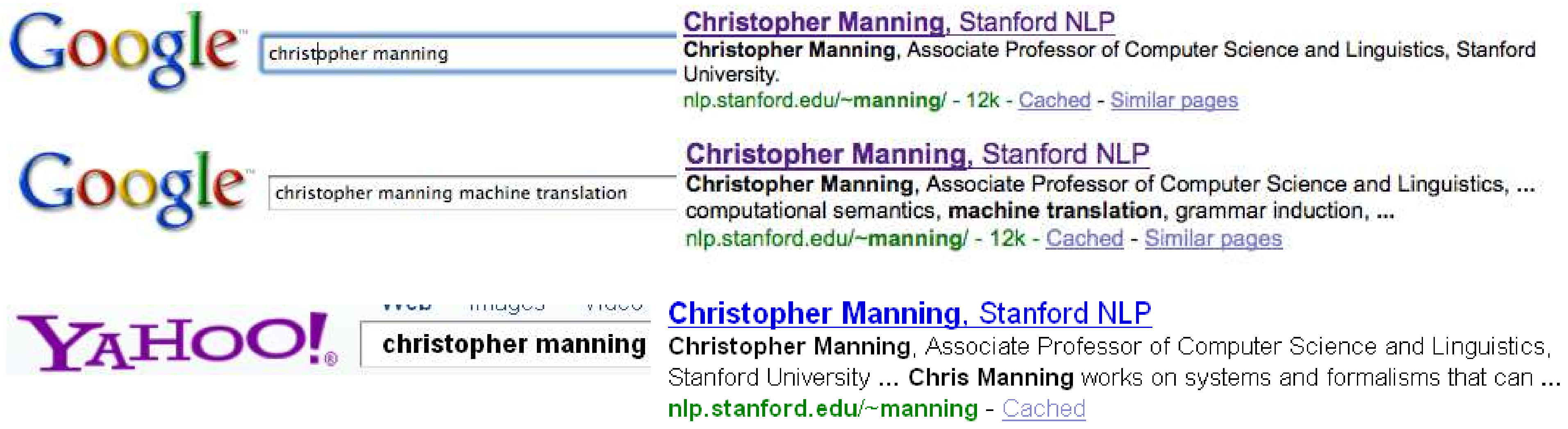
- The title is often automatically extracted from document metadata. What about the summaries?
  - This description is crucial.
  - User can identify good/relevant hits based on description.
- Two basic kinds:
  - Static
  - Dynamic
- A **static summary** of a document is always the same, regardless of the query that hit the doc
- A **dynamic summary** is a *query-dependent* attempt to explain why the document was retrieved for the query at hand

Static Summaries

- In typical systems, the static summary is a subset of the document
- Simplest heuristic: the first 50 (or so – this can be varied) words of the document
  - Summary cached at indexing time
- More sophisticated: extract from each document a set of “key” sentences
  - Simple NLP heuristics to score each sentence
  - Summary is made up of top-scoring sentences.
- Most sophisticated: NLP used to synthesize a summary
  - Seldom used in IR

Dynamic Summaries

- Present one or more "windows" within the document that contain several of the query terms
  - "KWIC" snippets: Keyword in Context presentation



christopher manning

Christopher Manning, Stanford NLP
Christopher Manning, Associate Professor of Computer Science and Linguistics, Stanford University.
nlp.stanford.edu/~**manning**/ - 12k - Cached - Similar pages

christopher manning machine translation

Christopher Manning, Stanford NLP
Christopher Manning, Associate Professor of Computer Science and Linguistics, ...
computational semantics, **machine translation**, grammar induction, ...
nlp.stanford.edu/~**manning**/ - 12k - Cached - Similar pages

christopher manning

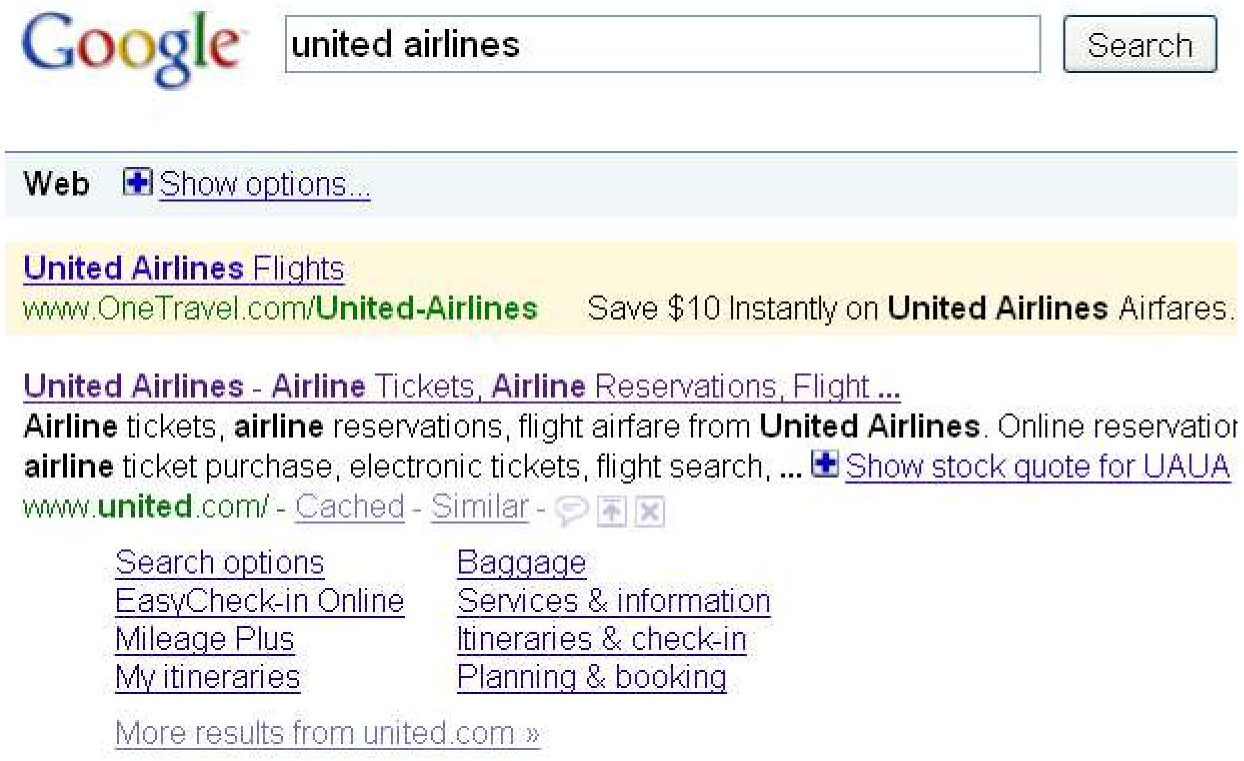
Christopher Manning, Stanford NLP
Christopher Manning, Associate Professor of Computer Science and Linguistics, Stanford University ... **Chris Manning** works on systems and formalisms that can ...
nlp.stanford.edu/~**manning** - Cached

Techniques for Dynamic Summaries

- Find small windows in doc that contain query terms
  - Requires fast window lookup in a document cache
- Score each window wrt query
  - Use various features such as window width, position in document, etc.
  - Combine features through a scoring function
- Challenges in evaluation: judging summaries
  - Easier to do pairwise comparisons rather than binary relevance assessments

Quicklinks

- For a *navigational query* such as ***united airlines*** user's need likely satisfied on www.united.com
- Quicklinks provide navigational cues on that home page



united airlines

Search

Web Show options...

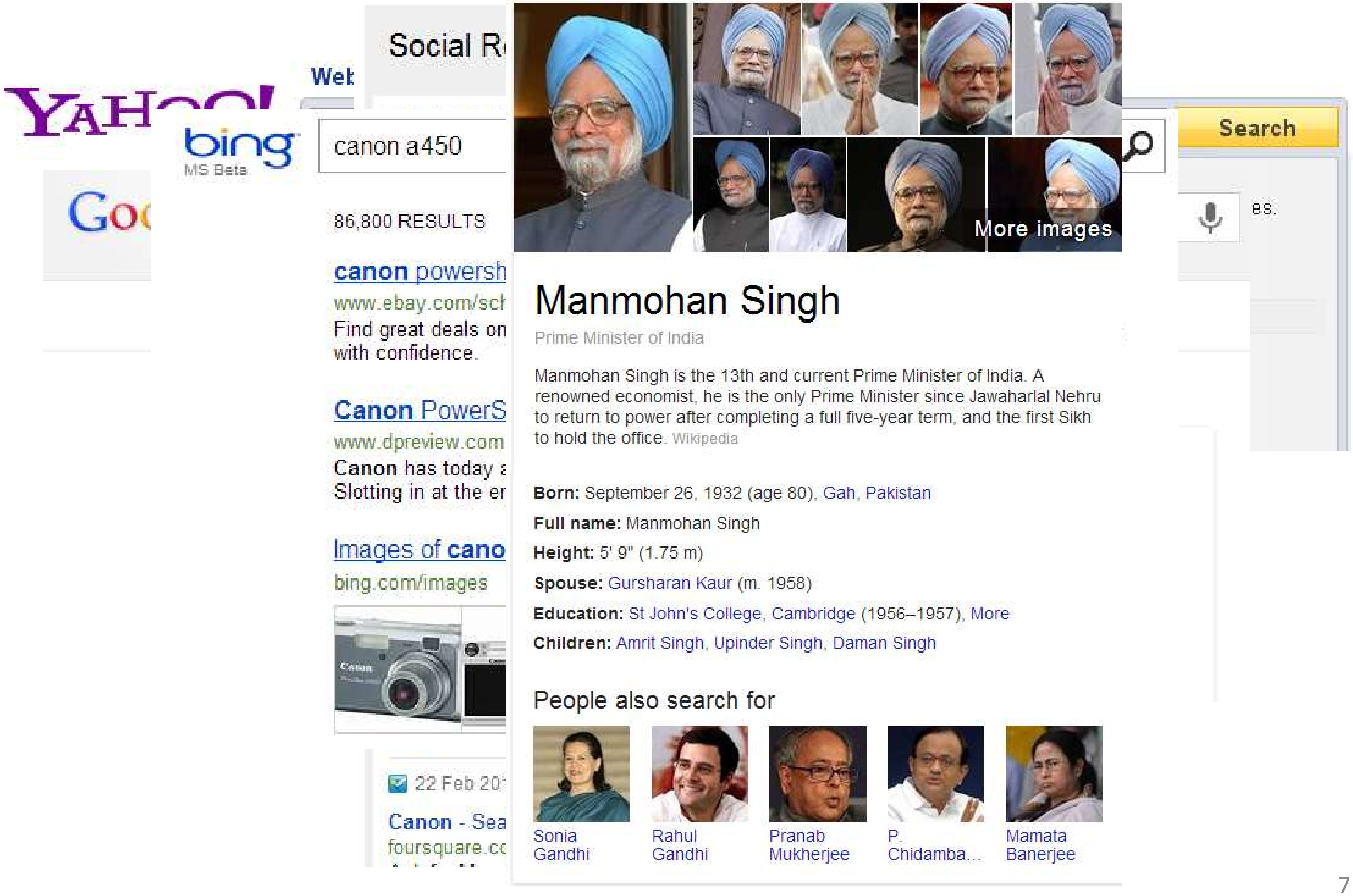
United Airlines Flights
www.OneTravel.com/United-Airlines Save $10 Instantly on United Airlines Airfares.

United Airlines - Airline Tickets, Airline Reservations, Flight ...
Airline tickets, airline reservations, flight airfare from United Airlines. Online reservatio
airline ticket purchase, electronic tickets, flight search, ... Show stock quote for UAUA
www.united.com/ - Cached - Similar -

Search options	Baggage
EasyCheck-in Online	Services & information
Mileage Plus	Itineraries & check-in
My itineraries	Planning & booking

More results from united.com »

Alternative Results Presentations?



Social R

Web

canon a450

86,800 RESULTS

canon powersh
www.ebay.com/sch
Find great deals on
with confidence.

Canon PowerS
www.dpreview.com
Canon has today a
Slotting in at the er

Images of cano
bing.com/images

22 Feb 20
Canon - Sea
foursquare.co

Search

es.

Manmohan Singh

Prime Minister of India

Manmohan Singh is the 13th and current Prime Minister of India. A renowned economist, he is the only Prime Minister since Jawaharlal Nehru to return to power after completing a full five-year term, and the first Sikh to hold the office. Wikipedia

Born: September 26, 1932 (age 80), Gah, Pakistan

Full name: Manmohan Singh

Height: 5' 9" (1.75 m)

Spouse: Gursharan Kaur (m. 1958)

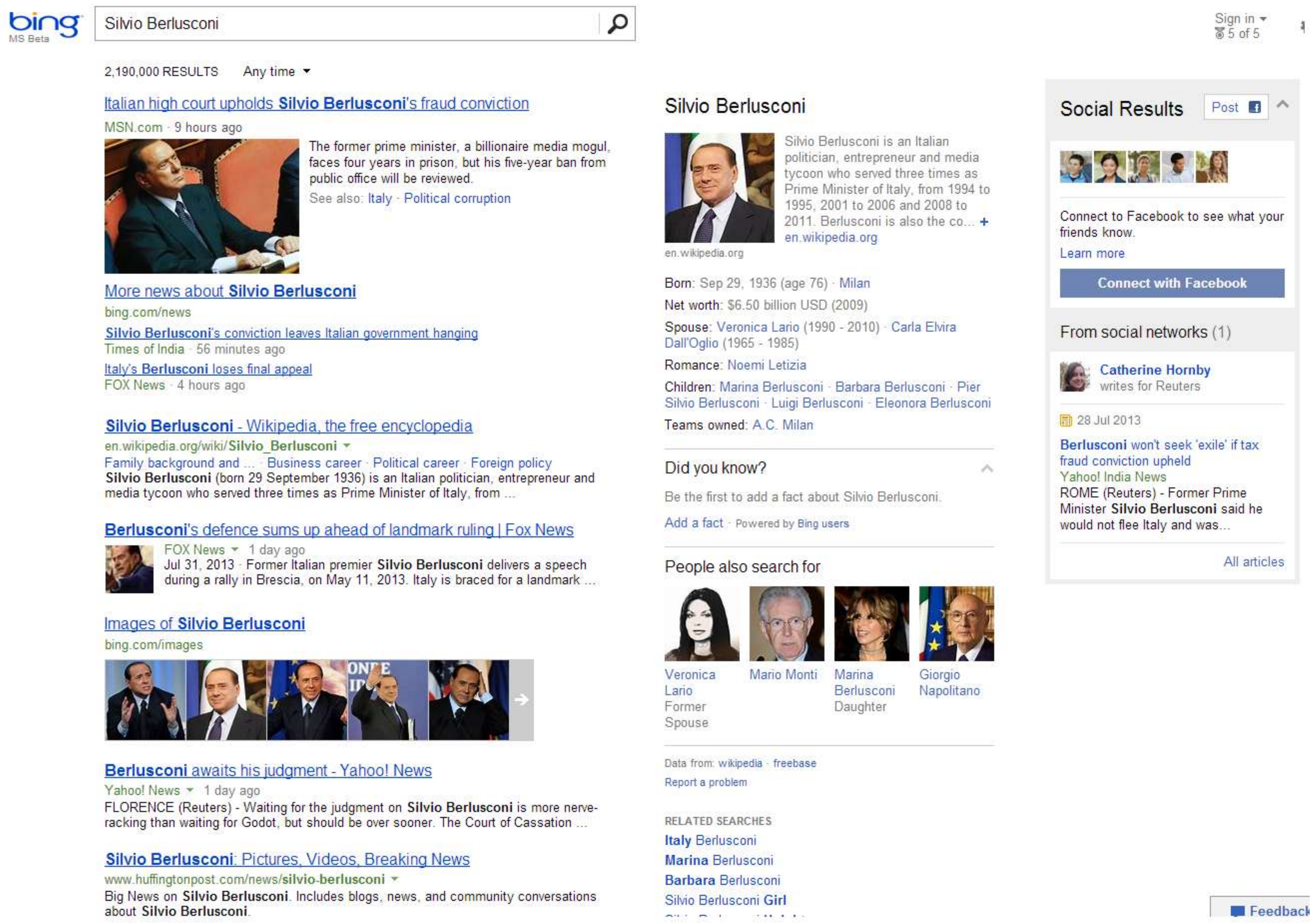
Education: St John's College, Cambridge (1956–1957), More

Children: Amrit Singh, Upinder Singh, Daman Singh

People also search for

Sonia Gandhi | Rahul Gandhi | Pranab Mukherjee | P. Chidamba... | Mamata Banerjee

Complexity of Heterogeneous Results: How to Rank?



bing MS Beta

Silvio Berlusconi

Sign in
5 of 5

2,190,000 RESULTS Any time

Italian high court upholds **Silvio Berlusconi**'s fraud conviction
MSN.com · 9 hours ago
The former prime minister, a billionaire media mogul, faces four years in prison, but his five-year ban from public office will be reviewed.
See also: Italy · Political corruption

More news about **Silvio Berlusconi**
bing.com/news
Silvio Berlusconi's conviction leaves Italian government hanging
Times of India · 56 minutes ago
Italy's **Berlusconi** loses final appeal
FOX News · 4 hours ago

Silvio Berlusconi - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Silvio_Berlusconi**
Family background and ... · Business career · Political career · Foreign policy
Silvio Berlusconi (born 29 September 1936) is an Italian politician, entrepreneur and media tycoon who served three times as Prime Minister of Italy, from ...

Berlusconi's defence sums up ahead of landmark ruling | Fox News
FOX News · 1 day ago
Jul 31, 2013 · Former Italian premier **Silvio Berlusconi** delivers a speech during a rally in Brescia, on May 11, 2013. Italy is braced for a landmark ...

Images of **Silvio Berlusconi**
bing.com/images

Berlusconi awaits his judgment - Yahoo! News
Yahoo! News · 1 day ago
FLORENCE (Reuters) - Waiting for the judgment on **Silvio Berlusconi** is more nerve-racking than waiting for Godot, but should be over sooner. The Court of Cassation ...

Silvio Berlusconi: Pictures, Videos, Breaking News
www.huffingtonpost.com/news/**silvio-berlusconi**
Big News on **Silvio Berlusconi**. Includes blogs, news, and community conversations about **Silvio Berlusconi**.

Silvio Berlusconi

Silvio Berlusconi is an Italian politician, entrepreneur and media tycoon who served three times as Prime Minister of Italy, from 1994 to 1995, 2001 to 2006 and 2008 to 2011. Berlusconi is also the co... +
en.wikipedia.org
en.wikipedia.org

Born: Sep 29, 1936 (age 76) · Milan
Net worth: $6.50 billion USD (2009)
Spouse: Veronica Lario (1990 - 2010) · Carla Elvira Dall'Oglio (1965 - 1985)
Romance: Noemi Letizia
Children: Marina Berlusconi · Barbara Berlusconi · Pier Silvio Berlusconi · Luigi Berlusconi · Eleonora Berlusconi
Teams owned: A.C. Milan

Did you know?
Be the first to add a fact about Silvio Berlusconi.
Add a fact · Powered by Bing users

People also search for
Veronica Lario Former Spouse
Mario Monti
Marina Berlusconi Daughter
Giorgio Napolitano

Data from: wikipedia · freebase
Report a problem

RELATED SEARCHES
Italy Berlusconi
Marina Berlusconi
Barbara Berlusconi
Silvio Berlusconi **Girl**

Social Results Post

Connect to Facebook to see what your friends know.
Learn more
Connect with Facebook

From social networks (1)
Catherine Hornby
writes for Reuters
28 Jul 2013
Berlusconi won't seek 'exile' if tax fraud conviction upheld
Yahoo! India News
ROME (Reuters) - Former Prime Minister **Silvio Berlusconi** said he would not flee Italy and was...
All articles

Feedback

Take-away Messages

- Today we learned
  - How the IR world moved from binary ranking to relevance ranking
  - TF and IDF are the basis of all ranking schemes
  - Vector Space Model and how to make ranking efficient
  - Evaluation metrics for ranking

Preview of Lecture 3: Similarity Search

- Shingling
- Min-hash
- Divide-Compute-Merge Solution
- Locally Sensitive Hashing (LSH)
- Applications of LSH
- Theory of LSH
- Index-based Methods

Disclaimers

- This course represents opinions of the instructor only. It does not reflect views of Microsoft or any other entity (except of authors from whom the slides have been borrowed).
- Algorithms, techniques, features, etc mentioned here might or might not be in use by Microsoft or any other company.
- Lot of material covered in this course is borrowed from slides across many universities and conference tutorials. These are gratefully acknowledged.

Thanks!