

ΕΠΛ660

Ανάκτηση Πληροφοριών και Μηχανές Αναζήτησης

- Computing Scores in a complete search system & Evaluation in IR

Recap: tf-idf weighting

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

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

- Best known weighting scheme in information retrieval
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Recap: Queries as vectors

- [Key idea 1](#): Do the same for queries: represent them as vectors in the space
- [Key idea 2](#): Rank documents according to their proximity to the query in this space
- proximity = similarity of vectors

Recap: cosine(query,document)

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

Dot product
Unit vectors

$\cos(\vec{q}, \vec{d})$ is the cosine similarity of \vec{q} and \vec{d} ... or, equivalently, the cosine of the angle between \vec{q} and \vec{d} .

This lecture

- Speeding up vector space ranking
- Putting together a complete search system
 - Will require learning about a number of miscellaneous topics and heuristics

This lecture

- How do we know if our results are any good?
 - Evaluating a search engine
 - Benchmarks
 - Precision and recall
- Results summaries:
 - Making our good results usable to a user

Computing cosine scores

COSINESCORE(q)

- 1 *float* $Scores[N] = 0$
- 2 *float* $Length[N]$
- 3 **for each** query term t
- 4 **do** calculate $w_{t,q}$ and fetch postings list for t
- 5 **for each** pair($d, tf_{t,d}$) in postings list
- 6 **do** $Scores[d] += w_{t,d} \times w_{t,q}$
- 7 Read the array $Length$
- 8 **for each** d
- 9 **do** $Scores[d] = Scores[d] / Length[d]$
- 10 **return** Top K components of $Scores[]$

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?

Efficient cosine ranking

- What we're doing in effect: solving the K -nearest neighbor problem for a query vector
- In general, we do not know how to do this efficiently for high-dimensional spaces
- But it is solvable for short queries, and standard indexes support this well

Special case – unweighted queries

- No weighting on query terms
 - Assume each query term occurs only once
- Then for ranking, don't need to normalize query vector
 - Slight simplification of algorithm from Lecture 6

Faster cosine: unweighted query

```
FASTCOSINESCORE( $q$ )
1  float  $Scores[N] = 0$ 
2  for each  $d$ 
3  do Initialize  $Length[d]$  to the length of doc  $d$ 
4  for each query term  $t$ 
5  do calculate  $w_{t,q}$  and fetch postings list for  $t$ 
6     for each pair( $d, tf_{t,d}$ ) in postings list
7     do add  $wf_{t,d}$  to  $Scores[d]$ 
8  Read the array  $Length[d]$ 
9  for each  $d$ 
10 do Divide  $Scores[d]$  by  $Length[d]$ 
11 return Top  $K$  components of  $Scores[]$ 
```

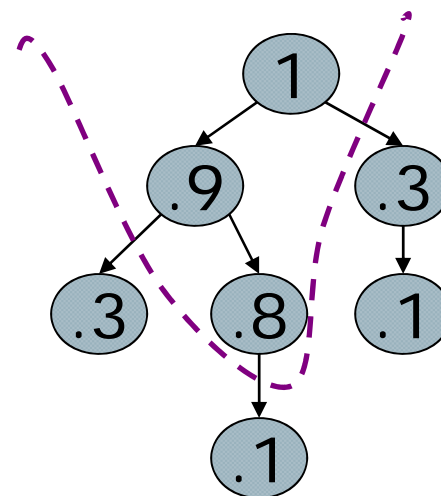
Slides by 1 **Figure 7.1** A faster algorithm for vector space scores.

Computing the K largest cosines: selection vs. sorting

- Typically we want to retrieve the top K docs (in the cosine ranking for the query)
 - not to totally order all docs in the collection
- Can we pick off docs with K highest cosines?
- Let J = number of docs with nonzero cosines
 - We seek the K best of these J

Use heap for selecting top K

- Binary tree in which each node's value $>$ the values of children
- Takes $2J$ operations to construct, then each of K “winners” read off in $2\log J$ steps.
- For $J=1\text{M}$, $K=100$, this is about 10% of the cost of sorting.



Bottlenecks

- Primary computational bottleneck in scoring: cosine computation
- **Can we avoid all this computation?**
- Yes, but may sometimes get it wrong
 - a doc *not* in the top K may creep into the list of K output docs
 - Is this such a bad thing?

Cosine similarity is only a proxy

- User has a task and a query formulation
- Cosine matches docs to query
- Thus cosine is anyway a proxy for user happiness
- If we get a list of K docs “close” to the top K by cosine measure, should be ok

Generic approach

- Find a set A of *contenders*, with $K < |A| \ll N$
 - 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 pruning non-contenders
- The same approach is also used for other (non-cosine) scoring functions
- Will look at several schemes following this approach

Index elimination

- Basic algorithm FastCosineScore of Fig 7.1 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

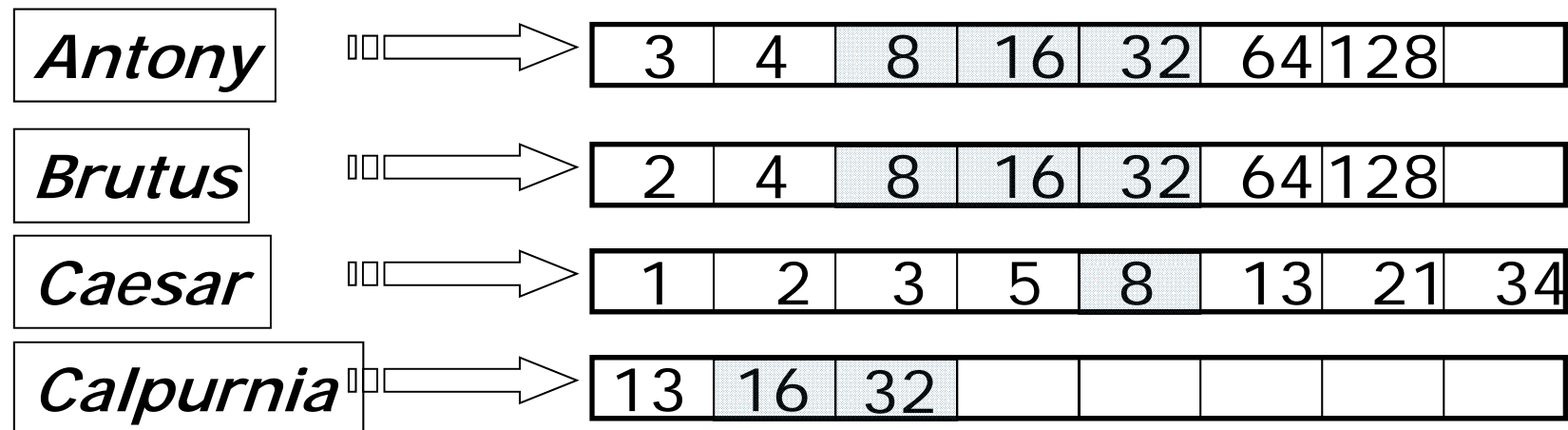
High-idf query terms only

- For a query such as *catcher in the rye*
- **Only accumulate scores from *catcher* and *rye***
- Intuition: *in* and *the* contribute little to the scores and so don't alter rank-ordering much
- Benefit:
 - **Postings of low-idf terms have many docs → these (many) docs get eliminated from set A of contenders**

Docs containing many query terms

- Any doc with at least one query term is a candidate for the top K output list
- For multi-term queries, only compute scores for docs containing several of the query terms
 - Say, at least 3 out of 4
 - Imposes a “soft conjunction” on queries seen on web search engines (early Google)
- Easy to implement in postings traversal

3 of 4 query terms



Scores only computed for docs 8, 16 and 32.

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 build time
 - Thus, it's possible that $r < K$
- 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

Exercises

- How do Champion Lists relate to Index Elimination?
Can they be used together?
- How can Champion Lists be implemented in an inverted index?
 - Note that the champion list has nothing to do with small docIDs

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)**

Quantitative



Modeling authority

- Assign to each document a *query-independent quality score* in $[0,1]$ to each document d
 - Denote this by $g(d)$
- Thus, a quantity like the number of citations is scaled into $[0,1]$
 - Exercise: suggest a formula for this.

Net score

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

Top K by net score – fast methods

- First idea: Order all postings by $g(d)$
- **Key: this is a common ordering for all postings**
- Thus, can concurrently traverse query terms' postings for
 - Postings intersection
 - Cosine score computation
- **Exercise: write pseudocode for cosine score computation if postings are ordered by $g(d)$**

Why order postings by $g(d)$?

- Under $g(d)$ -ordering, top-scoring docs likely to appear early in postings traversal
- In time-bound applications (say, we have to return whatever search results we can in 50 ms), this allows us to stop postings traversal early
 - Short of computing scores for all docs in postings

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 $wf_{t,d}$ is high enough
- We sort each postings list by $wf_{t,d}$
- Now: not all postings in a common order!
- How do we compute scores in order to pick off top K ?
 - Two ideas follow

1. Early termination

- When traversing t 's postings, stop early after either
 - a fixed number of r docs
 - $wf_{t,d}$ 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 terms

- When considering the postings of query terms
- Look at them 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

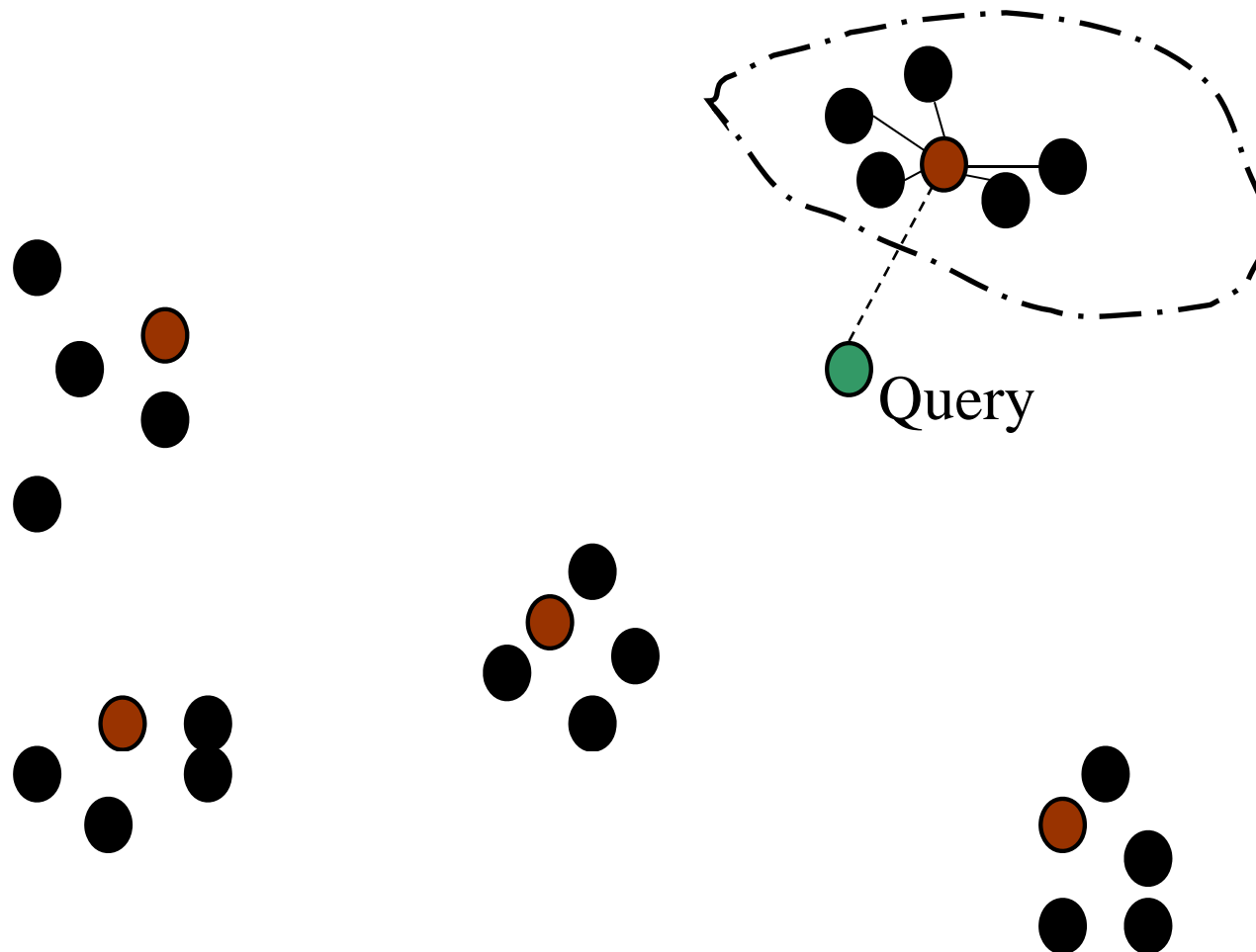
Cluster pruning: preprocessing

- Pick \sqrt{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 $\sim \sqrt{N}$ followers.

Cluster pruning: query processing

- Process a query as follows:
 - Given query Q , find its nearest *leader* L .
 - Seek K nearest docs from among L 's followers.

Visualization



● Leader

● Follower

Why use random sampling

- Fast
- Leaders reflect data distribution

General variants

- Have each follower attached to $b_1=3$ (say) nearest leaders.
- From query, find $b_2=4$ (say) nearest leaders and their followers.
- Can recur on leader/follower construction.

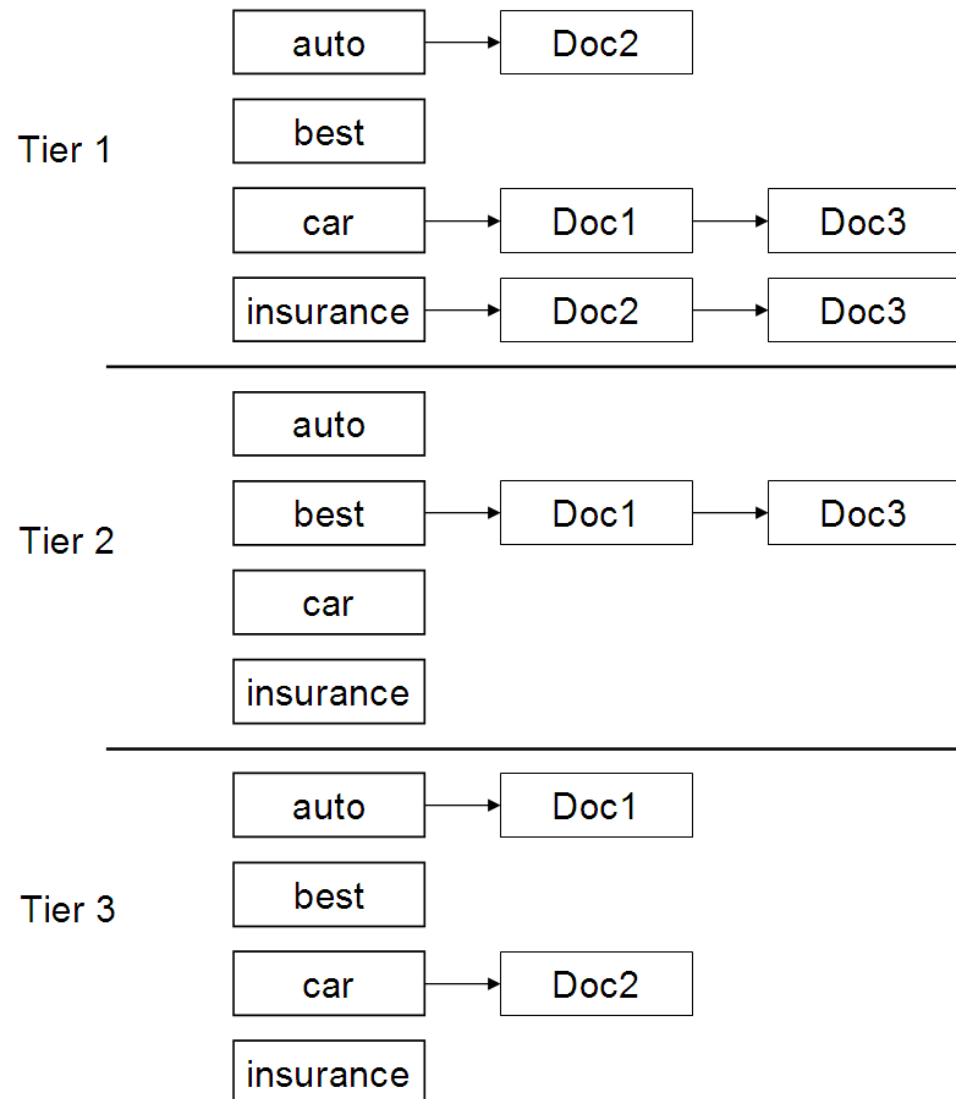
Exercises

- To find the nearest leader in step 1, how many cosine computations do we do?
 - Why did we have \sqrt{N} in the first place?
- What is the effect of the constants $b1$, $b2$ on the previous slide?
- Devise an example where this is *likely to fail* – i.e., we miss one of the K nearest docs.
 - *Likely* under random sampling.

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

Example tiered index



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?

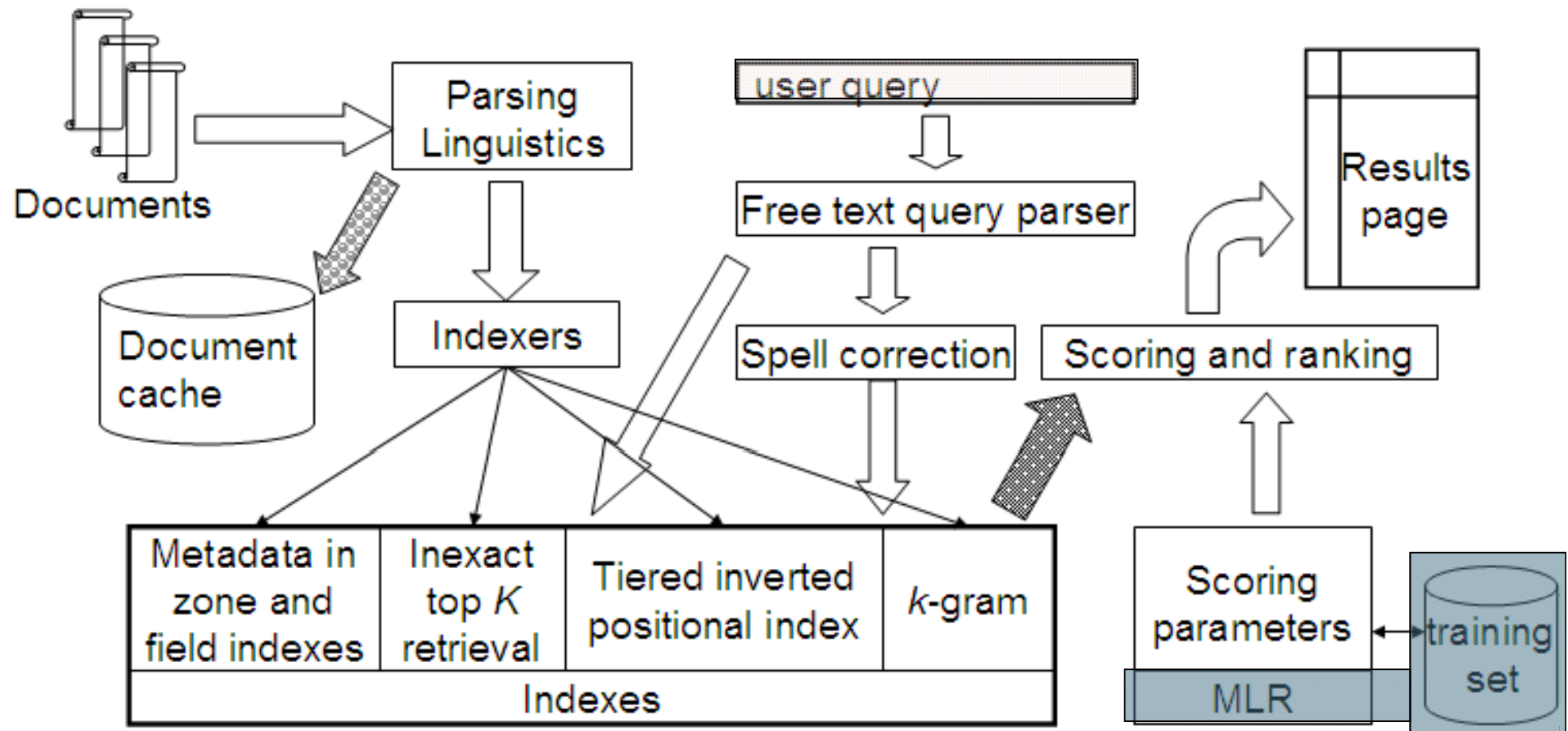
Query parsers

- 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 query parser

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**

Putting it all together



EVALUATING SEARCH ENGINES

Measures for a search engine

- How fast does it index
 - Number of documents/hour
 - (Average document size)
- 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?

Measures for a search engine

- All of the preceding criteria are *measurable*: we can quantify speed/size
 - we can make expressiveness precise
- The key measure: user happiness
 - What is this?
 - Speed of response/size of index are factors
 - But blindingly fast, useless answers won't make a user happy
- Need a way of quantifying user happiness

Measuring user happiness

- Issue: who is the user we are trying to make happy?
 - Depends on the setting
- Web engine:
 - User finds what they want and return to the engine
 - Can measure rate of return users
 - User completes their task – search as a means, not end
 - See Russell <http://dmrussell.googlepages.com/JCDL-talk-June-2007-short.pdf>
- eCommerce site: user finds what they want and buy
 - Is it the end-user, or the eCommerce site, whose happiness we measure?
 - Measure time to purchase, or fraction of searchers who become buyers?

Measuring user happiness

- Enterprise (company/govt/academic): Care about “user productivity”
 - How much time do my users save when looking for information?
 - Many other criteria having to do with breadth of access, secure access, etc.

Happiness: elusive to measure

- Most common proxy: *relevance* of search results
- But how do you measure relevance?
- We will detail a methodology here, then examine its issues
- Relevance measurement requires 3 elements:
 1. A benchmark document collection
 2. A benchmark suite of queries
 3. A usually binary assessment of either Relevant or Nonrelevant for each query and each document
 - Some work on more-than-binary, but not the standard

Evaluating an IR system

- Note: the **information need** is translated into a **query**
- Relevance is assessed relative to the **information need** *not* the **query**
- E.g., Information need: *I'm looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine.*
- Query: **wine red white heart attack effective**
- You evaluate whether the doc addresses the information need, not whether it has these words

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

Unranked retrieval evaluation: Precision and Recall

- **Precision:** fraction of retrieved docs that are relevant = $P(\text{relevant} | \text{retrieved})$
- **Recall:** fraction of relevant docs that are retrieved = $P(\text{retrieved} | \text{relevant})$

	Relevant	Nonrelevant
Retrieved	tp	fp
Not Retrieved	fn	tn

- Precision $P = \text{tp} / (\text{tp} + \text{fp})$
- Recall $R = \text{tp} / (\text{tp} + \text{fn})$

Should we instead use the accuracy measure for evaluation?

- Given a query, an engine classifies each doc as “Relevant” or “Nonrelevant”
- The **accuracy** of an engine: the fraction of these classifications that are correct
 - $(tp + tn) / (tp + fp + fn + tn)$
- **Accuracy** is a commonly used evaluation measure in machine learning classification work
- Why is this not a very useful evaluation measure in IR?

Why not just use accuracy?

- How to build a 99.9999% accurate search engine on a low budget....



- People doing information retrieval *want to find something* and have a certain tolerance for junk.

Precision/Recall

- You can get high recall (but low precision) by retrieving all docs for all queries!
- Recall is a non-decreasing function of the number of docs retrieved
- In a good system, precision decreases as either the number of docs retrieved or recall increases
 - This is not a theorem, but a result with strong empirical confirmation

Difficulties in using precision/recall

- Should average over large document collection/query ensembles
- Need human relevance assessments
 - People aren't reliable assessors
- Assessments have to be binary
 - Nuanced assessments?
- Heavily skewed by collection/authorship
 - Results may not translate from one domain to another

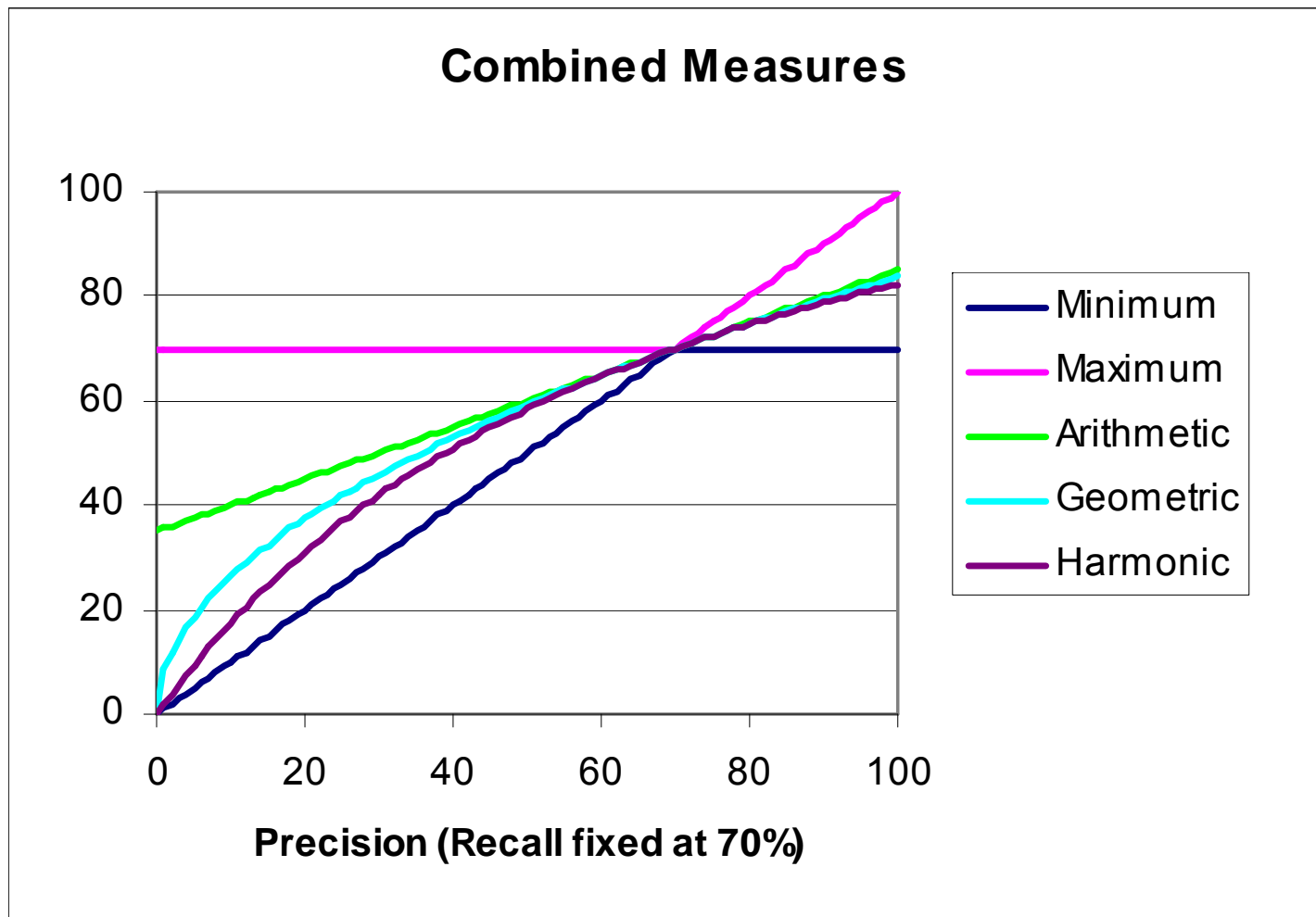
A combined measure: F

- 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 $\alpha = \frac{1}{2}$
- Harmonic mean is a conservative average
 - See CJ van Rijsbergen, *Information Retrieval*

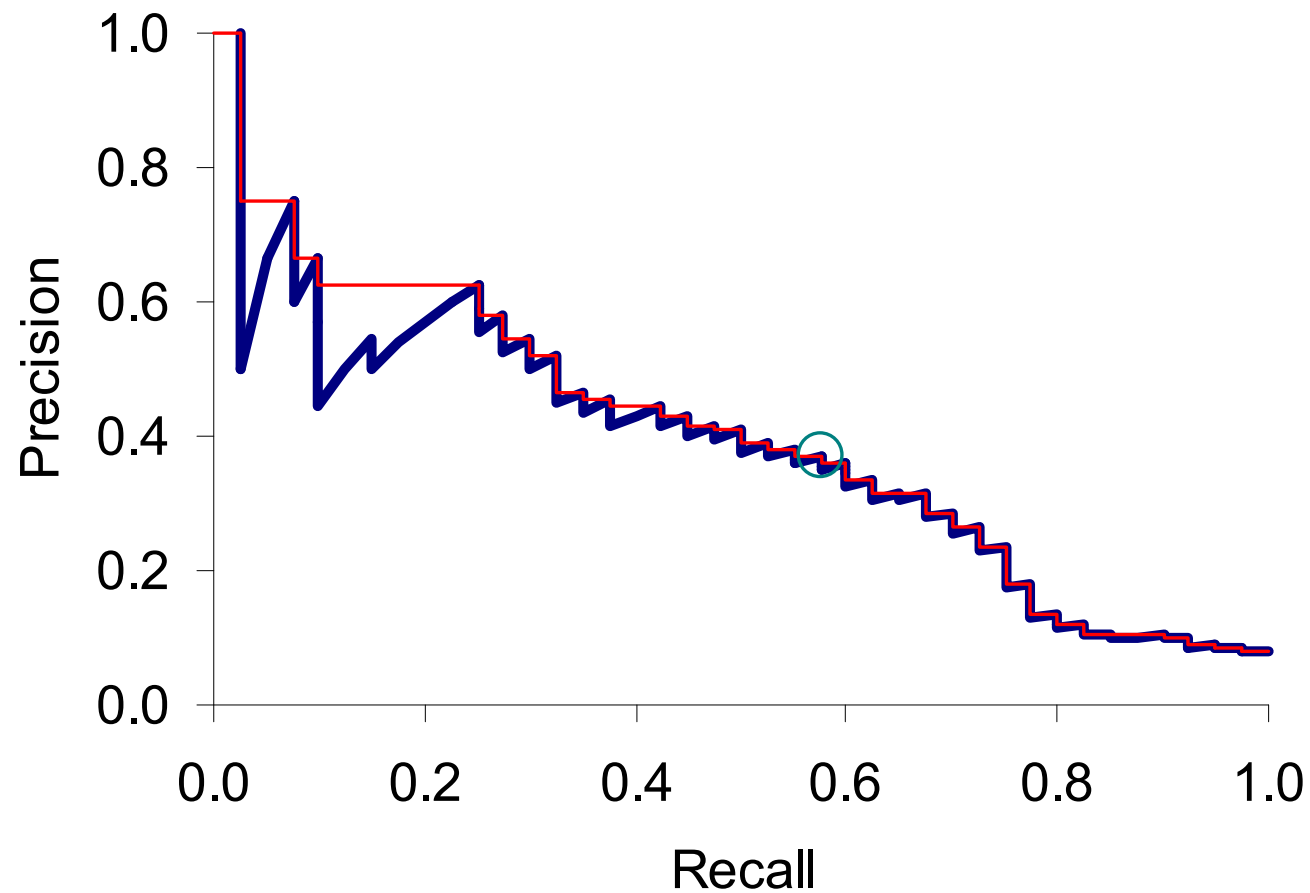
F_1 and other averages



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*

A precision-recall curve

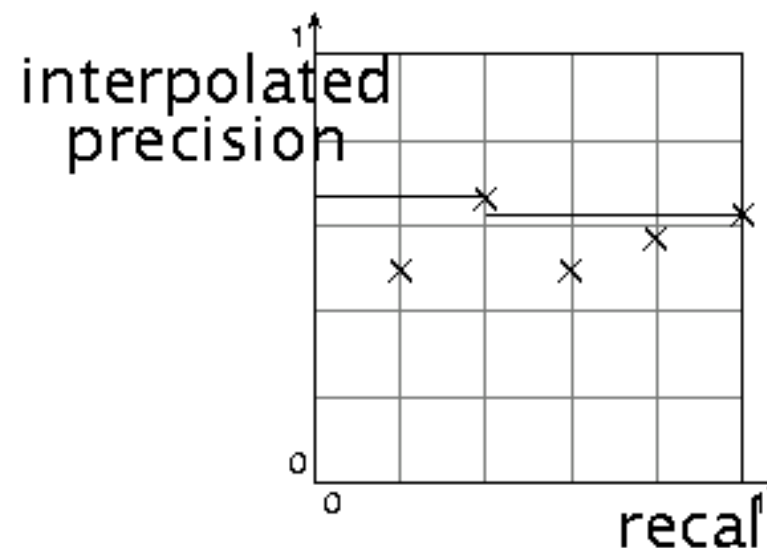
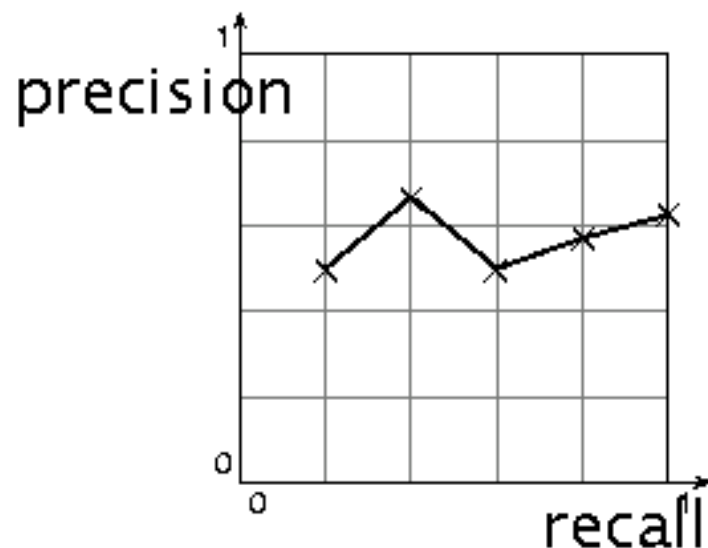


Averaging over queries

- 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?

Interpolated precision

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

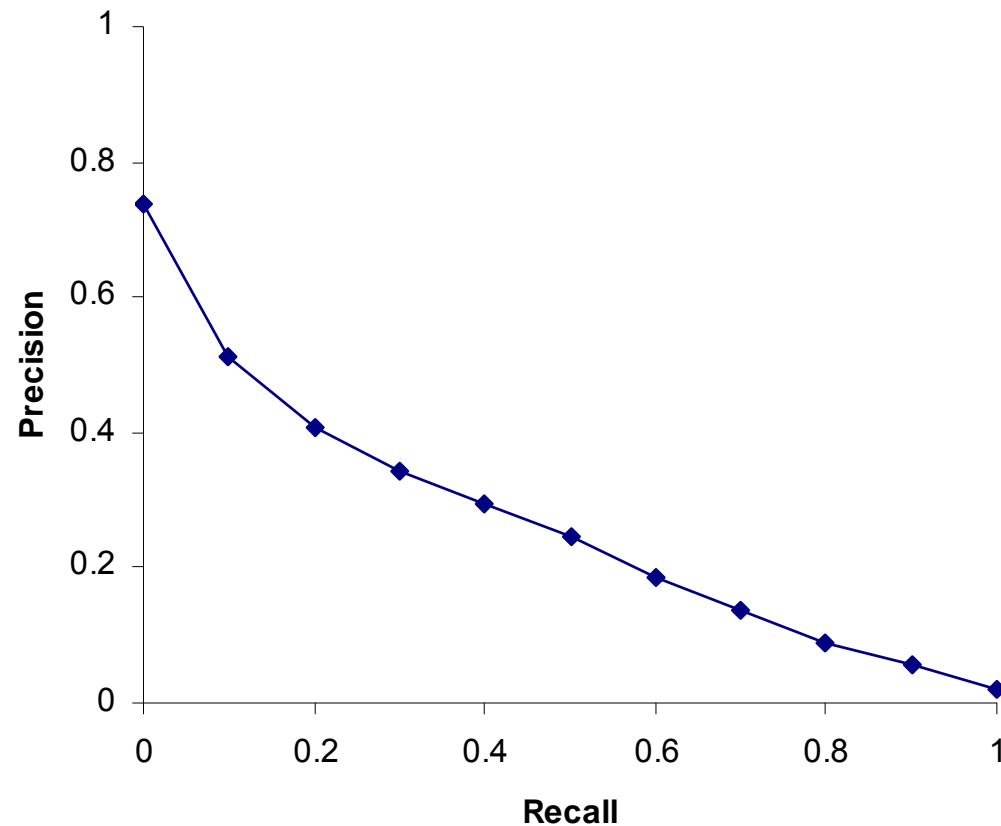


Evaluation

- Graphs are good, but people want summary measures!
 - Precision at fixed retrieval level
 - Precision-at- k : Precision of top k results
 - Perhaps appropriate for most of web search: all people want are good matches on the first one or two results pages
 - But: averages badly and has an arbitrary parameter of k
 - 11-point interpolated average precision
 - The standard measure in the early TREC competitions: you 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
 - Evaluates performance at all recall levels

Typical (good) 11 point precisions

- SabIR/Cornell 8A1 11pt precision from TREC 8 (1999)



Yet more evaluation measures...

- Mean average precision (MAP)
 - Average of the precision value obtained for the top k documents, each time a relevant doc is retrieved
 - Avoids interpolation, use of fixed recall levels
 - MAP for query collection is arithmetic ave.
 - Macro-averaging: each query counts equally
- R-precision
 - If have known (though perhaps incomplete) set of relevant documents of size Rel , then calculate precision of top Rel docs returned
 - Perfect system could score 1.0.

Variance

- For a test collection, it is usual that a system does crummily on some information needs (e.g., MAP = 0.1) and excellently on others (e.g., MAP = 0.7)
- Indeed, it is usually the case that the variance in performance of the same system across queries is much greater than the variance of different systems on the same query.
- That is, there are easy information needs and hard ones!

CREATING TEST COLLECTIONS FOR IR EVALUATION

Test Collections

TABLE 4.3 Common Test Corpora

<i>Collection</i>	<i>NDocs</i>	<i>NQrys</i>	<i>Size (MB)</i>	<i>Term/Doc</i>	<i>Q-D RelAss</i>
ADI	82	35			
AIT	2109	14	2	400	>10,000
CACM	3204	64	2	24.5	
CISI	1460	112	2	46.5	
Cranfield	1400	225	2	53.1	
LISA	5872	35	3		
Medline	1033	30	1		
NPL	11,429	93	3		
OSHMED	34,8566	106	400	250	16,140
Reuters	21,578	672	28	131	
TREC	740,000	200	2000	89-3543	» 100,000

From document collections to test collections

- Still need
 - Test queries
 - Relevance assessments
- Test queries
 - Must be germane to docs available
 - Best designed by domain experts
 - Random query terms generally not a good idea
- Relevance assessments
 - Human judges, time-consuming
 - Are human panels perfect?

Unit of Evaluation

- We can compute precision, recall, F, and ROC curve for different units.
- Possible units
 - Documents (most common)
 - Facts (used in some TREC evaluations)
 - Entities (e.g., car companies)
- May produce different results. Why?

Kappa measure for inter-judge (dis)agreement

- Kappa measure
 - Agreement measure among judges
 - Designed for categorical judgments
 - Corrects for chance agreement
- $\text{Kappa} = [P(A) - P(E)] / [1 - P(E)]$
- $P(A)$ – 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

Kappa Example

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

- $\text{Kappa} > 0.8$ = good agreement
- $0.67 < \text{Kappa} < 0.8$ -> “tentative conclusions” (Carletta '96)
- Depends on purpose of study
- For >2 judges: average pairwise kappas

TREC

- TREC Ad Hoc task from first 8 TRECs is standard IR task
 - 50 detailed information needs a year
 - Human evaluation of pooled results returned
 - More recently other related things: Web track, HARD

- A TREC query (TREC 5)

<top>

<num> Number: 225

<desc> Description:

What is the main function of the Federal Emergency Management Agency (FEMA) and the funding level provided to meet emergencies?
Also, what resources are available to FEMA such as people, equipment, facilities?

</top>

Standard relevance benchmarks:

Others

- GOV2
 - Another TREC/NIST collection
 - 25 million web pages
 - Largest collection that is easily available
 - But still 3 orders of magnitude smaller than what Google/Yahoo/MSN index
- NTCIR
 - East Asian language and cross-language information retrieval
- Cross Language Evaluation Forum (CLEF)
 - This evaluation series has concentrated on European languages and cross-language information retrieval.
- Many others

Interjudge Agreement: TREC 3

information need	number of docs judged	disagreements	NR	R
51	211	6	4	2
62	400	157	149	8
67	400	68	37	31
95	400	110	108	2
127	400	106	12	94

Impact of Inter-judge Agreement

- Impact on **absolute** performance measure can be significant (0.32 vs 0.39)
- Little impact on ranking of different systems or **relative** performance
- Suppose we want to know if algorithm A is better than algorithm B
- A standard information retrieval experiment will give us a reliable answer to this question.

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.
- But harder to create evaluation set
- See Carbonell reference

Can we avoid human judgment?

- No
- Makes experimental work hard
 - Especially on a large scale
- In some very specific settings, can use proxies
 - E.g.: for approximate vector space retrieval, we can compare the cosine distance closeness of the closest docs to those found by an approximate retrieval algorithm
- But once we have test collections, we can reuse them (so long as we don't overtrain too badly)

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

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

RESULTS PRESENTATION

Result Summaries

- Having ranked the documents matching a query, we wish to present a results list
- Most commonly, a list of the document titles plus a short summary, aka “10 blue links”

[John McCain](#)

John McCain 2008 - The Official Website of **John McCain's** 2008 Campaign for President ... African American Coalition; Americans of Faith; American Indians for **McCain**; Americans with ...
www.johnmccain.com · [Cached page](#)

[JohnMcCain.com - McCain-Palin 2008](#)

John McCain 2008 - The Official Website of **John McCain's** 2008 Campaign for President ... African American Coalition; Americans of Faith; American Indians for **McCain**; Americans with ...
www.johnmccain.com/Informing/Issues · [Cached page](#)

[John McCain News- msnbc.com](#)

Complete political coverage of **John McCain**. ... Republican leaders said Saturday that they were worried that Sen. **John McCain** was heading for defeat unless he brought stability to ...
www.msnbc.msn.com/id/16438320 · [Cached page](#)

[John McCain | Facebook](#)

Welcome to the official Facebook Page of **John McCain**. Get exclusive content and interact with **John McCain** right from Facebook. Join Facebook to create your own Page or to start ...

www.facebook.com/johnmccain · [Cached page](#)

Summaries


- 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; cf. text summarization work

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

 [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, 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



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 reservation **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 »](#)

YAHOO! [images](#) [video](#) [Local](#) [Shopping](#) [more](#)

united airlines

Also try: [united airlines reservations](#), [united airlines flight](#), [More...](#)

United Airlines - Airline Tickets, Airline Reservations ... (Nasdaq: [UAUA](#))
 Official site for **United Airlines**, commercial air carrier transporting people, property, and mail across the U.S. and worldwide.
[www.united.com](#) - 65k - [Cached](#)

[Planning & Booking](#) [Shop for Flights](#)
[Itineraries & Check-in](#) [Special Deals](#)
[Mileage Plus](#) [Flight Status](#)
[Services & Information](#) [Customer Service](#)

[more results from united.com »](#)

Search Pad
 SearchScan - On

102,000,000 results for **united airlines**:

[Show All](#)

[United Air Lines](#)

[Wikipedia](#)

bing united airlines

UNITED AIRLINES

United **Airline Fleet**

United **Airline Schedule**

United Airlines **Reservations**

United **Airline Jobs**

Reference

ALL RESULTS

[Cheap Flight Tickets](#) · [www.CheapOair.com](#)
 CheapOair - The Only Way to Go!! Find Over 18 Million Exclusive Fares.

[Fly United Airlines](#) · [www.OneTravel.com/United-Airline](#)
 Save \$10 Instantly on **United Airlines** Flights. Book Now, Hurry!

Best match

[United Airlines - Airline Tickets, Airline Reservations, Flight ...](#)
[www.united.com](#) · Official site
Airline tickets, **airline** reservations, flight airfare from **United Airlines**. Online reservations, **airline** ticket purchase, electronic tickets, flight search, fares and availability ...

Flights Redeem miles
 Check In Online Children, pets, & assistance
 My itineraries Change your travel plans
 Baggage Special deals

Customer service 800-864-8331

RELATED SEARCHES

United Airlines **Flight Status**

US Airways

Continental Airlines

Alternative results presentations?

- An active area of HCI research
- An alternative: <http://www.searchme.com/> / copies the idea of Apple's Cover Flow for search results
 - (searchme recently went out of business)



Resources for this lecture

- IIR 7
- IIR 8
- MIR Chapter 3
- MG 4.5
- Carbonell and Goldstein 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. SIGIR 21.