CS276A

Information Retrieval

Lecture 8

Recap of the last lecture

- Vector space scoring
- Efficiency considerations
  - Nearest neighbors and approximations

This lecture

- Results summaries
- Evaluating a search engine
  - Benchmarks
  - Precision and recall

Results summaries

Summaries

- Having ranked the documents matching a query, we wish to present a results list
- Typically, the document title plus a short summary
- Title – typically automatically extracted
- What about the summaries?

Summaries

- Two basic kinds:
  - Static and
  - Query-dependent (Dynamic)
- A static summary of a document is always the same, regardless of the query that hit the doc
- Dynamic summaries 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, seldom used for search results: NLP used to synthesize a summary

Dynamic summaries

- Present one or more "windows" within the document that contain several of the query terms
- Generated in conjunction with scoring
  - If query found as a phrase, the occurrences of the phrase in the doc
  - If not, windows within the doc that contain multiple query terms
- The summary itself gives the entire content of the window – all terms, not only the query terms – how?

Generating dynamic summaries

- If we have only a positional index, cannot (easily) reconstruct context surrounding hits
- If we cache the documents at index time, can run the window through it, cueing to hits found in the positional index
  - E.g., positional index says "the query is a phrase in position 4378" so we go to this position in the cached document and stream out the content
- Most often, cache a fixed-size prefix of the doc
  - Cached copy can be outdated

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
  - Speed on complex queries

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
- eCommerce site: user finds what they want and make a purchase
  - 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 … more later

Happiness: elusive to measure

- Commonest proxy: *relevance* of search results
- But how do you measure relevance?
- Will detail a methodology here, then examine its issues
- Requires 3 elements:
  1. A benchmark document collection
  2. A benchmark suite of queries
  3. A binary assessment of either Relevant or Irrelevant for each query-doc pair

Evaluating an IR system

- Note: **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***

Standard relevance benchmarks

- TREC - National Institute of Standards and Testing (NIST) has run 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 Irrelevant
  - or at least for subset of docs that some system returned for that query

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	Not Relevant
Retrieved	tp	fp
Not Retrieved	fn	tn

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

Accuracy

- Given a query an engine classifies each doc as "Relevant" or "Irrelevant".
- Accuracy of an engine: the fraction of these classifications that is correct.

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

- 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
  - Precision usually decreases (in a good system)

Difficulties in using precision/recall

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

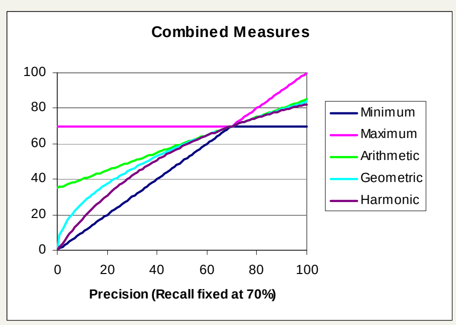
A combined measure: *F*

- Combined measure that assesses this 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 conservative average
  - See CJ van Rijsbergen, *Information Retrieval*

F_1 and other averages



Ranked results

- Evaluation of ranked results:
  - You can return any number of results
  - By taking various numbers of returned documents (levels of recall), you can produce a *precision-recall curve*

Precision-recall curves

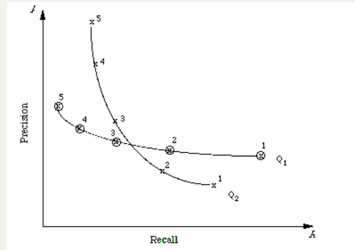
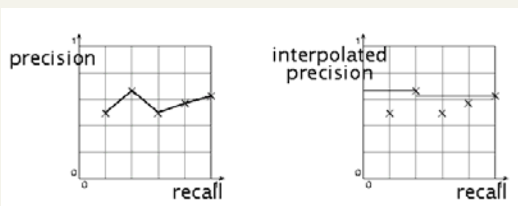


Figure 7.1. The precision-recall curves for two queries. The ordinals indicate the values of the control parameter λ

Interpolated precision

- If you can increase precision by increasing recall, then you should get to count that…



Evaluation

- There are various other measures
  - Precision at fixed recall
    - Perhaps most appropriate for web search: all people want are good matches on the first one or two results pages
  - 11-point interpolated average precision
    - The standard measure in the 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

Creating Test Collections for IR Evaluation

Test Corpora

TABLE 4.3 Common Test Corpora

Collection	*NDocs*	*NQrys*	*Size (MB)*	*Term/Doc*	*Q-D RelAss*
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 corpora 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?

Kappa measure for inter-judge (dis)agreement

- Kappa measure
  - Agreement among judges
  - Designed for categorical judgments
  - Corrects for chance agreement
- Kappa = [P(A) – P(E)] / [1 – P(E)]
- P(A) – proportion of time coders 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

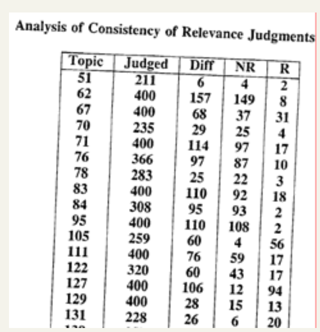
- P(A) = 370/400 = 0.925
- P(nonrelevant) = (10+20+70+70)/800 = 0.2125
- P(relevant) = (10+20+300+300)/800 = 0.7878
- P(E) = 0.2125^2 + 0.7878^2 = 0.665
- Kappa = (0.925 – 0.665)/(1-0.665) = 0.776

- For >2 judges: average pairwise kappas

Kappa Measure

- Kappa > 0.8 = good agreement
- 0.67 < Kappa < 0.8 -> “tentative conclusions” (Carletta 96)
- Depends on purpose of study

Interjudge Agreement: TREC 3


Analysis of Consistency of Relevance Judgments

Topic	Judged	Diff	NR	R
51	211	6	4	2
62	400	157	149	8
67	400	68	37	31
70	235	29	25	4
71	400	114	97	17
76	366	97	87	10
78	283	25	22	3
83	400	110	92	18
84	308	95	93	2
95	400	110	108	2
105	259	60	4	56
111	400	76	59	17
122	320	60	43	17
127	400	106	12	94
129	400	28	15	13
131	228	26	6	20

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

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?

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?

- Not really
- Makes experimental work hard
  - Especially on a large scale
- In some very specific settings, can use proxies
- Example below, approximate vector space retrieval

Approximate vector retrieval

- Given *n* document vectors and a query, find the *k* doc vectors closest to the query.
- Exact retrieval – we know of no better way than to compute cosines from the query to every doc
- Approximate retrieval schemes – such as cluster pruning in lecture 6
- Given such an approximate retrieval scheme, how do we measure its goodness?

Approximate vector retrieval

- Let $G(q)$ be the “ground truth” of the actual k closest docs on query q
- Let $A(q)$ be the k docs returned by approximate algorithm A on query q
- For precision and recall we would measure $A(q) \cap G(q)$
  - Is this the right measure?

Alternative proposal

- Focus instead on how *A(q)* compares to *G(q).*
- Goodness can be measured here in cosine proximity to *q:* we sum up $q \bullet d$ over $d \in A(q)$.
- Compare this to the sum of $q \bullet d$ over $d \in G(q)$.
  - Yields a measure of the relative "goodness" of *A* vis-à-vis *G*.
  - Thus *A* may be 90% "as good as" the ground-truth *G*, without finding 90% of the docs in *G*.
  - For scored retrieval, this may be acceptable:
  - Most web engines don't always return the same answers for a given query.

Resources for this lecture

- MIR Chapter 3
- MG 4.5