IS4200/CS6200 Information Retrieval

David Smith
College of Computer and Information Science
Northeastern University

Evaluation

- Evaluation is key to building *effective* and *efficient* search engines
  - measurement usually carried out in controlled laboratory experiments
  - *online* testing can also be done
- Efficiency measures similar to database systems
  - e.g., indexing time, query throughput, index size
- Our focus is on *effectiveness* metrics

Evaluation Corpus

- *Test collections* consisting of documents, queries, and relevance judgments, e.g.,
  - CACM: Titles and abstracts from the Communications of the ACM from 1958-1979. Queries and relevance judgments generated by computer scientists.
  - AP: Associated Press newswire documents from 1988-1990 (from TREC disks 1-3). Queries are the title fields from TREC topics 51-150. Topics and relevance judgments generated by government information analysts.
  - GOV2: Web pages crawled from websites in the .gov domain during early 2004. Queries are the title fields from TREC topics 701-850. Topics and relevance judgments generated by government analysts.

Test Collections

Collection	Number of documents	Size	Average number of words/doc.
CACM	3,204	2.2 Mb	64
AP	242,918	0.7 Gb	474
GOV2	25,205,179	426 Gb	1073

Collection	Number of queries	Average number of words/query	Average number of relevant docs/query
CACM	64	13.0	16
AP	100	4.3	220
GOV2	150	3.1	180

TREC Topic Example

<top>
<num> Number: 794

<title> pet therapy

<desc> Description:
How are pets or animals used in therapy for humans and what are the benefits?

<narr> Narrative:
Relevant documents must include details of how pet- or animal-assisted therapy is or has been used. Relevant details include information about pet therapy programs, descriptions of the circumstances in which pet therapy is used, the benefits of this type of therapy, the degree of success of this therapy, and any laws or regulations governing it.

</top>

Relevance Judgments

- Obtaining relevance judgments is an expensive, time-consuming process
- Sources of variation similar to psychometrics, opinion polling, etc.
  - who does it?
  - what are the instructions?
  - what is the level of agreement?
- TREC judgments
  - depend on task being evaluated
  - generally binary
  - agreement good because of "narrative"

Pooling

- Exhaustive judgments for all documents in a collection is not practical
- Pooling technique is used in TREC
  - top *k results (for TREC, k varied between 50 and* 200) from the rankings obtained by different search engines (or retrieval algorithms) are merged into a pool
  - duplicates are removed
  - documents are presented in some random order to the relevance judges
- Produces a large number of relevance judgments for each query, although still incomplete

Query Logs

- Used for both tuning and evaluating search engines
  – also for various techniques such as query suggestion
- Typical contents
  – User identifier or user session identifier
  – Query terms - stored exactly as user entered
  – List of URLs of results, their ranks on the result list, and whether they were clicked on
  – Timestamp(s) - records the time of user events such as query submission, clicks

Query Logs

- Clicks are not relevance judgments
  - although they are correlated
  - biased by a number of factors such as rank on result list
- Can use clickthough data to predict *preferences* between pairs of documents
  - appropriate for tasks with multiple levels of relevance, focused on user relevance
  - various "policies" used to generate preferences

Example Click Policy

- *Skip Above and Skip Next*
  - click data

$$\begin{array}{l} d_1 \\ d_2 \\ d_3 \ \text{(clicked)} \\ d_4 \end{array}$$

  - generated preferences

$$\begin{array}{l} d_3 > d_2 \\ d_3 > d_1 \\ d_3 > d_4 \end{array}$$

Query Logs

- Click data can also be aggregated to remove noise
- *Click distribution* information
  - can be used to identify clicks that have a higher frequency than would be expected
  - high correlation with relevance
  - e.g., using *click deviation* to filter clicks for preference-generation policies

Filtering Clicks

- *Click deviation CD(d, p)* for a result *d* in position *p*:

$$CD(d, p) = O(d, p) - E(p)$$

O(d,p): observed click frequency for a document in a rank position p *over all instances of a given query*

E(p): expected click frequency at rank p *averaged across all queries*

Effectiveness Measures

A is set of relevant documents,
B is set of retrieved documents

	Relevant	Non-Relevant
Retrieved	$A \cap B$	$\overline{A} \cap B$
Not Retrieved	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$

$$Recall = \frac{|A \cap B|}{|A|}$$

$$Precision = \frac{|A \cap B|}{|B|}$$

Classification Errors

- *False Positive* (Type I error)
  - a non-relevant document is retrieved

$$Fallout = \frac{|\overline{A} \cap B|}{|\overline{A}|}$$

- *False Negative* (Type II error)
  - a relevant document is not retrieved
  - 1- *Recall*
- *Precision* is used when probability that a positive result is correct is important

F Measure

- *Harmonic mean* of recall and precision

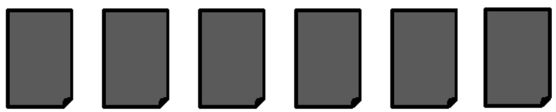
$$F = \frac{1}{\frac{1}{2}(\frac{1}{R} + \frac{1}{P})} = \frac{2RP}{(R+P)}$$

  - Generally used when averaging probabilities and proportions
  - Harmonic mean emphasizes the importance of small values, whereas the arithmetic mean is affected more by outliers that are unusually large
- More general form

$$F_\beta = (\beta^2 + 1)RP/(R + \beta^2 P)$$

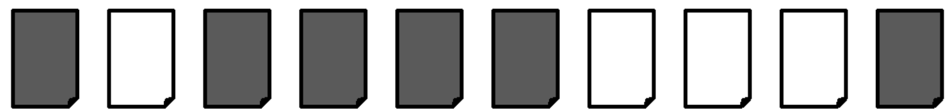
  - β is a parameter that determines relative importance of recall and precision

Ranking Effectiveness



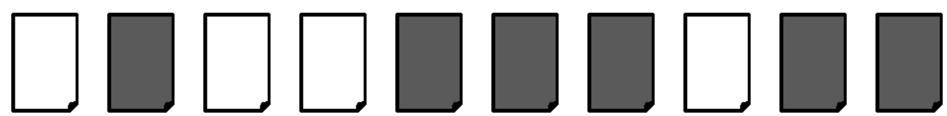
= the relevant documents

Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2

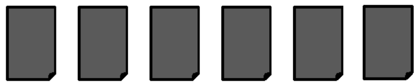


Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

Summarizing a Ranking

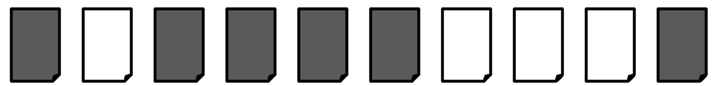
- Calculating recall and precision at fixed rank positions
- Calculating precision at standard recall levels, from 0.0 to 1.0
  - requires *interpolation*
- Averaging the precision values from the rank positions where a relevant document was retrieved

Average Precision



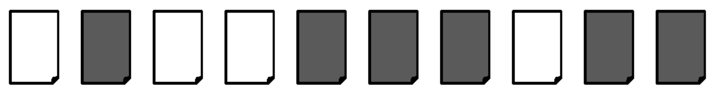
= the relevant documents

Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2

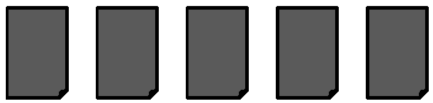


Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

$$\text{Ranking } \#1\text{: } (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$$

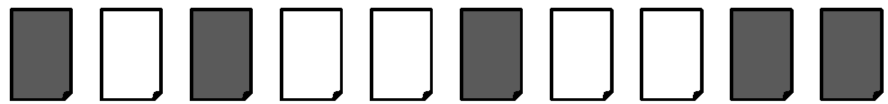
$$\text{Ranking } \#2\text{: } (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6)/6 = 0.52$$

Averaging Across Queries

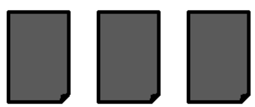


= relevant documents for query 1

Ranking #1

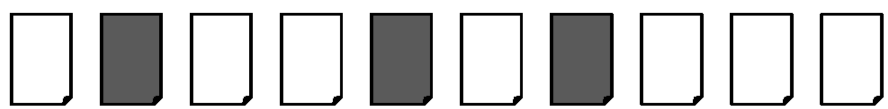


Recall	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.6	0.8	1.0
Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5



= relevant documents for query 2

Ranking #2

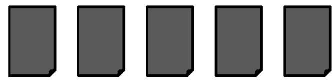


Recall	0.0	0.33	0.33	0.33	0.67	0.67	1.0	1.0	1.0	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.33	0.43	0.38	0.33	0.3

Averaging

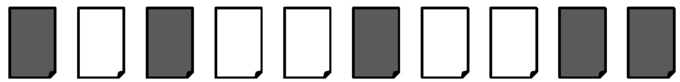
- *Mean Average Precision* (MAP)
  - summarize rankings from multiple queries by averaging average precision
  - most commonly used measure in research papers
  - assumes user is interested in finding many relevant documents for each query
  - requires many relevance judgments in text collection
- Recall-precision graphs are also useful summaries

MAP

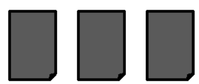


= relevant documents for query 1

Ranking #1

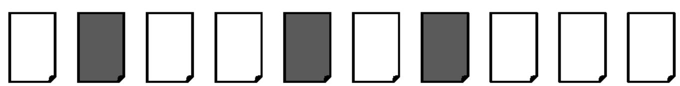


Recall	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.6	0.8	1.0
Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5



= relevant documents for query 2

Ranking #2

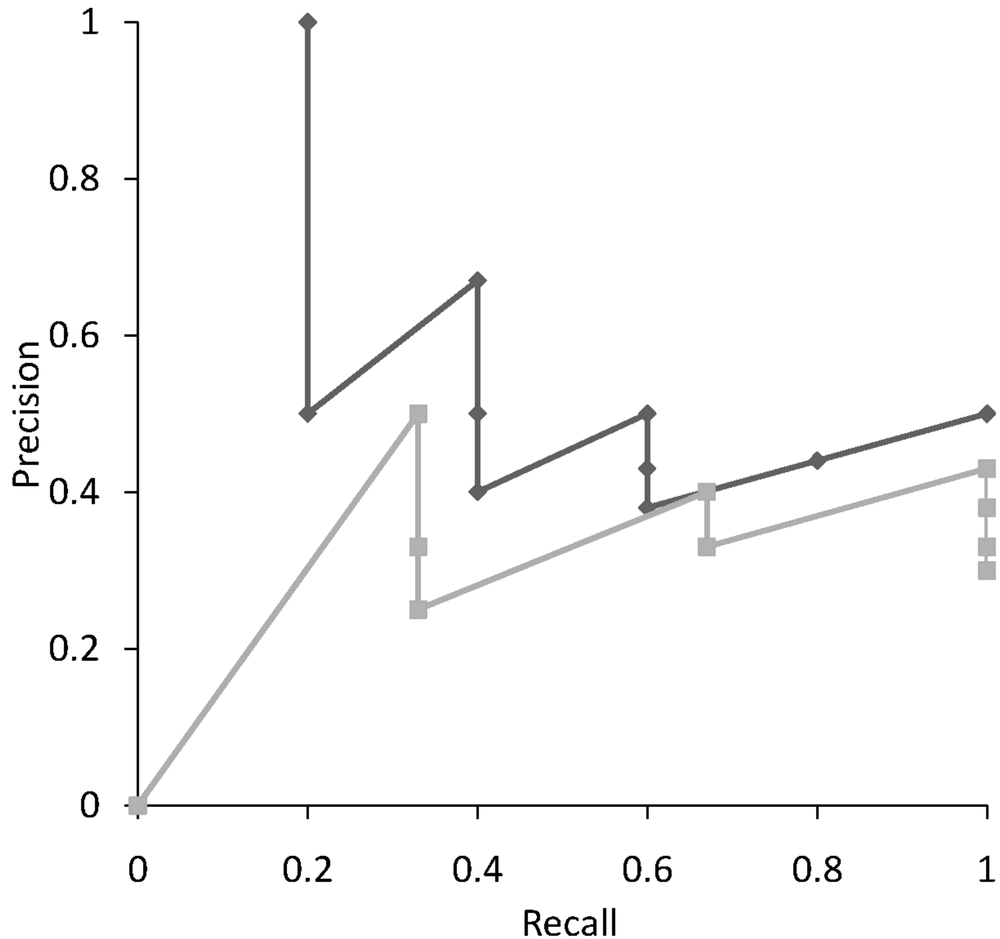


Recall	0.0	0.33	0.33	0.33	0.67	0.67	1.0	1.0	1.0	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.33	0.43	0.38	0.33	0.3

$$average\ precision\ query\ 1 = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$
$$average\ precision\ query\ 2 = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$mean\ average\ precision = (0.62 + 0.44)/2 = 0.53$$

Recall-Precision Graph



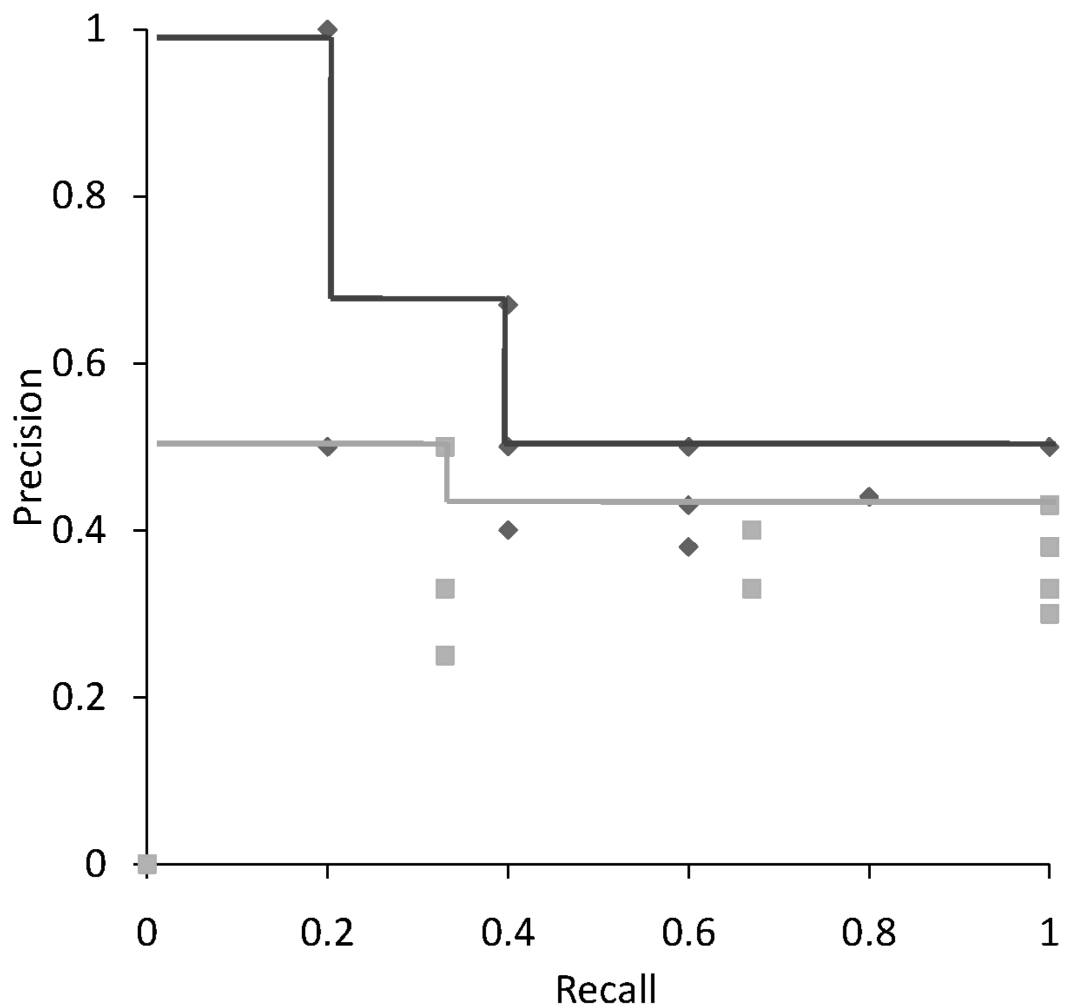
Interpolation

- To average graphs, calculate precision at standard recall levels:

$$P(R) = \max\{P' : R' \geq R \wedge (R', P') \in S\}$$

  – where *S* is the set of observed (*R,P*) points
- Defines precision at any recall level as the *maximum* precision observed in any recall-precision point at a higher recall level
  – produces a step function
  – defines precision at recall 0.0

Interpolation

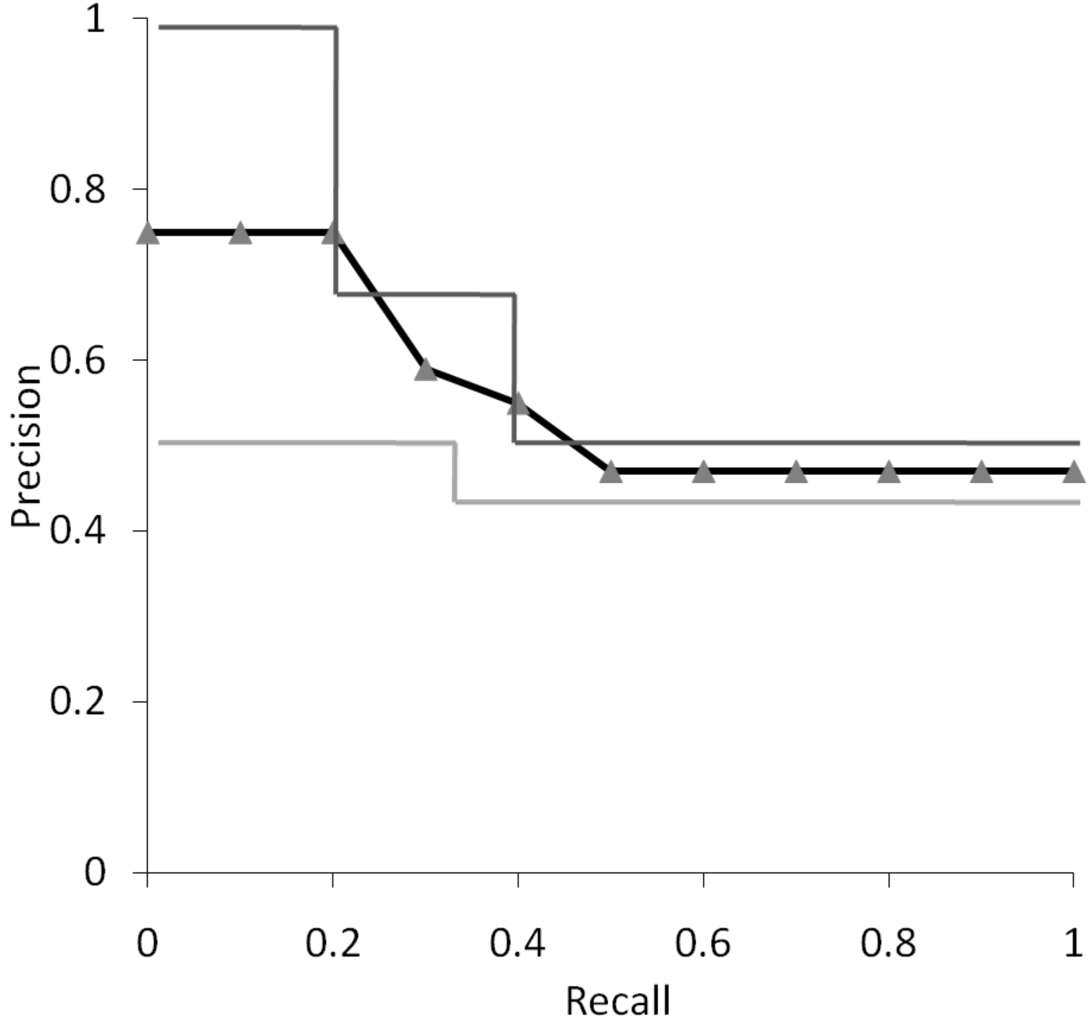


Average Precision at Standard Recall Levels

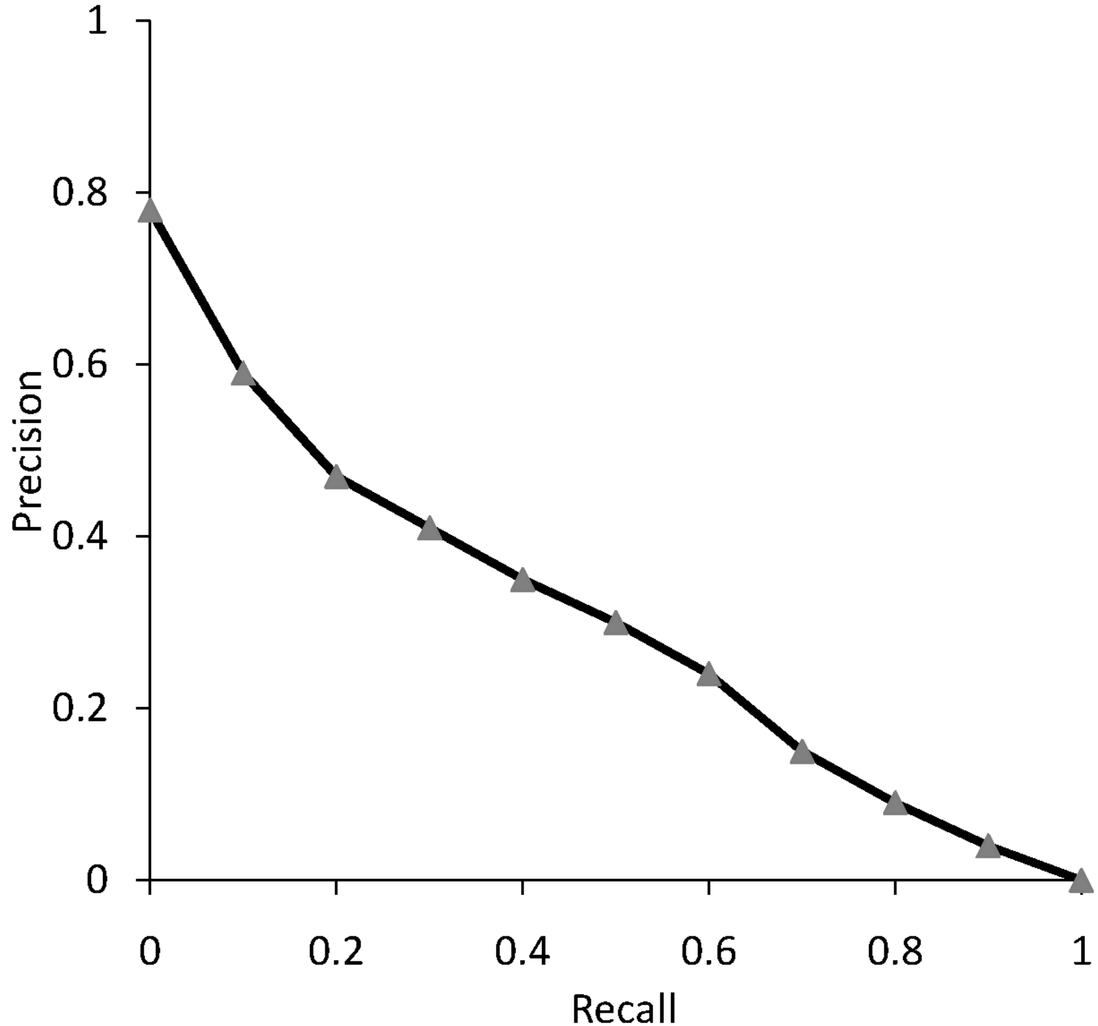
Recall	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Ranking 1	1.0	1.0	1.0	0.67	0.67	0.5	0.5	0.5	0.5	0.5	0.5
Ranking 2	0.5	0.5	0.5	0.5	0.43	0.43	0.43	0.43	0.43	0.43	0.43
Average	0.75	0.75	0.75	0.59	0.55	0.47	0.47	0.47	0.47	0.47	0.47

- Recall-precision graph plotted by simply joining the average precision points at the standard recall levels

Average Recall-Precision Graph



Graph for 50 Queries



Focusing on Top Documents

- Users tend to look at only the top part of the ranked result list to find relevant documents
- Some search tasks have only one relevant document
  - e.g., navigational search, question answering
- Recall not appropriate
  - instead need to measure how well the search engine does at retrieving relevant documents at very high ranks

Focusing on Top Documents

- Precision at Rank R
  - R typically 5, 10, 20
  - easy to compute, average, understand
  - not sensitive to rank positions less than R
- Reciprocal Rank
  - reciprocal of the rank at which the first relevant document is retrieved
  - *Mean Reciprocal Rank (MRR)* is the average of the reciprocal ranks over a set of queries
  - very sensitive to rank position

Discounted Cumulative Gain

- Popular measure for evaluating web search and related tasks
- Two assumptions:
  - Highly relevant documents are more useful than marginally relevant document
  - the lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined

Discounted Cumulative Gain

- Uses *graded relevance* as a measure of the usefulness, or *gain,* from examining a document
- Gain is accumulated starting at the top of the ranking and may be reduced, or *discounted*, at lower ranks
- Typical discount is 1/*log (rank)*
  - With base 2, the discount at rank 4 is 1/2, and at rank 8 it is 1/3

Discounted Cumulative Gain

- *DCG* is the total gain accumulated at a particular rank *p*:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

- Alternative formulation:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i}-1}{log(1+i)}$$

  – used by some web search companies
  – emphasis on retrieving highly relevant documents

DCG Example

- 10 ranked documents judged on 0-3 relevance scale:

  3, 2, 3, 0, 0, 1, 2, 2, 3, 0

- discounted gain:

  3, 2/1, 3/1.59, 0, 0, 1/2.59, 2/2.81, 2/3, 3/3.17, 0

  = 3, 2, 1.89, 0, 0, 0.39, 0.71, 0.67, 0.95, 0

- DCG:

  3, 5, 6.89, 6.89, 6.89, 7.28, 7.99, 8.66, 9.61, 9.61

Normalized DCG

- DCG numbers are averaged across a set of queries at specific rank values
  - e.g., DCG at rank 5 is 6.89 and at rank 10 is 9.61
- DCG values are often *normalized* by comparing the DCG at each rank with the DCG value for the *perfect ranking*
  - makes averaging easier for queries with different numbers of relevant documents

NDCG Example

- Perfect ranking:

  3, 3, 3, 2, 2, 2, 1, 0, 0, 0

- ideal DCG values:

  3, 6, 7.89, 8.89, 9.75, 10.52, 10.88, 10.88, 10.88, 10

- NDCG values (divide actual by ideal):

  1, 0.83, 0.87, 0.76, 0.71, 0.69, 0.73, 0.8, 0.88, 0.88

  – NDCG $\leq$ 1 at any rank position

Using Preferences

- Two rankings described using preferences can be compared using the *Kendall tau coefficient (τ):*

$$\tau = \frac{P - Q}{P + Q}$$

  - *P* is the number of preferences that agree and *Q* is the number that disagree
- For preferences derived from binary relevance judgments, can use *BPREF*

BPREF

- For a query with *R* relevant documents, only the first *R* non-relevant documents are considered

$$BPREF = \frac{1}{R} \sum_{d_r} (1 - \frac{N_{d_r}}{R})$$

  - d_r is a relevant document, and N_{dr} gives the number of non-relevant documents
- Alternative definition

$$BPREF = \frac{P}{P+Q}$$

Efficiency Metrics

Metric name	Description
Elapsed indexing time	Measures the amount of time necessary to build a document index on a particular system.
Indexing processor time	Measures the CPU seconds used in building a document index. This is similar to elapsed time, but does not count time waiting for I/O or speed gains from parallelism.
Query throughput	Number of queries processed per second.
Query latency	The amount of time a user must wait after issuing a query before receiving a response, measured in milliseconds. This can be measured using the mean, but is often more instructive when used with the median or a percentile bound.
Indexing temporary space	Amount of temporary disk space used while creating an index.
Index size	Amount of storage necessary to store the index files.

Hypothesis Testing

- One-sample
  - "Is the system's response time under 1 sec.?"
- Two-sample
  - "Does the system perform equally well on these two types of queries?"
- Paired-sample
  - "Is this technique better than that?"

Terminological Prelude

- Populations
  - Population distributions
  - "All possible files". How big?
- Samples
  - Sampling distributions
  - "Files on my system"
- Statistics
  - Functions of data
  - "Size of my files"
- Models
  - Parameters

Significance Tests

- Given the results from a number of queries, how can we conclude that ranking algorithm A is better than algorithm B?
- A significance test enables us to reject the *null hypothesis* (no difference) in favor of the *alternative hypothesis* (B is better than A)
  - the *power* of a test is the probability that the test will reject the null hypothesis correctly
  - increasing the number of queries in the experiment also increases power of test

Significance Tests

1. Compute the effectiveness measure for every query for both rankings.

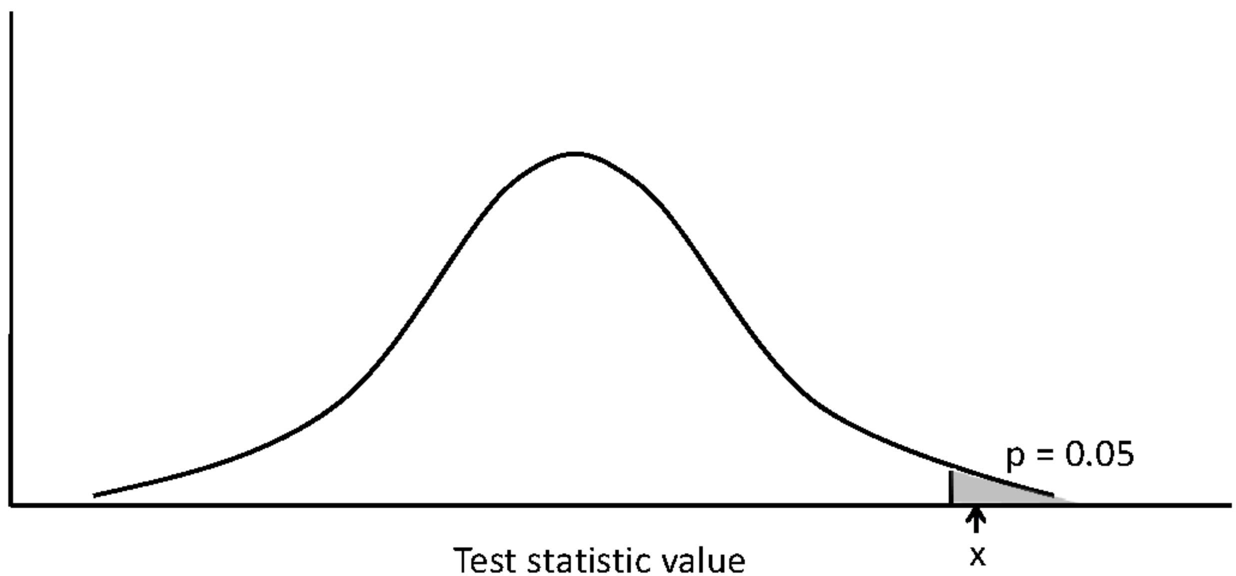
2. Compute a *test statistic* based on a comparison of the effectiveness measures for each query. The test statistic depends on the significance test, and is simply a quantity calculated from the sample data that is used to decide whether or not the null hypothesis should be rejected.

3. The test statistic is used to compute a *P-value*, which is the probability that a test statistic value at least that extreme could be observed if the null hypothesis were true. Small P-values suggest that the null hypothesis may be false.

4. The null hypothesis (no difference) is rejected in favor of the alternate hypothesis (i.e., B is more effective than A) if the P-value is $\leq \alpha$, the *significance level*. Values for α are small, typically .05 and .1, to reduce the chance of a Type I error.

One-Sided Test

- Distribution for the possible values of a test statistic assuming the null hypothesis



  - shaded area is *region of rejection*

Example Experimental Results

Query	A	B	B-A
1	25	35	10
2	43	84	41
3	39	15	-24
4	75	75	0
5	43	68	25
6	15	85	70
7	20	80	60
8	52	50	-2
9	49	58	9
10	50	75	25

t-Test

- Assumption is that the difference between the effectiveness values is a sample from a normal distribution
- Null hypothesis is that the mean of the distribution of differences is zero
- Test statistic

$$t = \frac{\overline{B-A}}{\sigma_{B-A}}.\sqrt{N}$$

  – for the example,

$$\overline{B-A} = 21.4,\ \sigma_{B-A} = 29.1,\ t = 2.33,\ \text{p-value}{=}.02$$

Wilcoxon Signed-Ranks Test

- Nonparametric test based on differences between effectiveness scores
- Test statistic

  $$w = \sum_{i=1}^{N} R_i$$

  R_i is a signed-rank, N is the number of differences $\neq 0$

  – To compute the signed-ranks, the differences are ordered by their absolute values (increasing), and then assigned rank values

  – rank values are then given the sign of the original difference

Wilcoxon Example

- 9 non-zero differences are (in rank order of absolute value):

  2, 9, 10, 24, 25, 25, 41, 60, 70

- Signed-ranks:

  -1, +2, +3, -4, +5.5, +5.5, +7, +8, +9

- w = 35, p-value = 0.025

Notation

- P is a population
- $S = [s_1, s_2, ..., s_n]$ is a sample from P

- Let $X = [x_1, x_2, ..., x_n]$ be some numerical measurement on the s_i
  - distributed over P according to unknown F

- We may use Y, Z for other measurements.

Plug-in principle

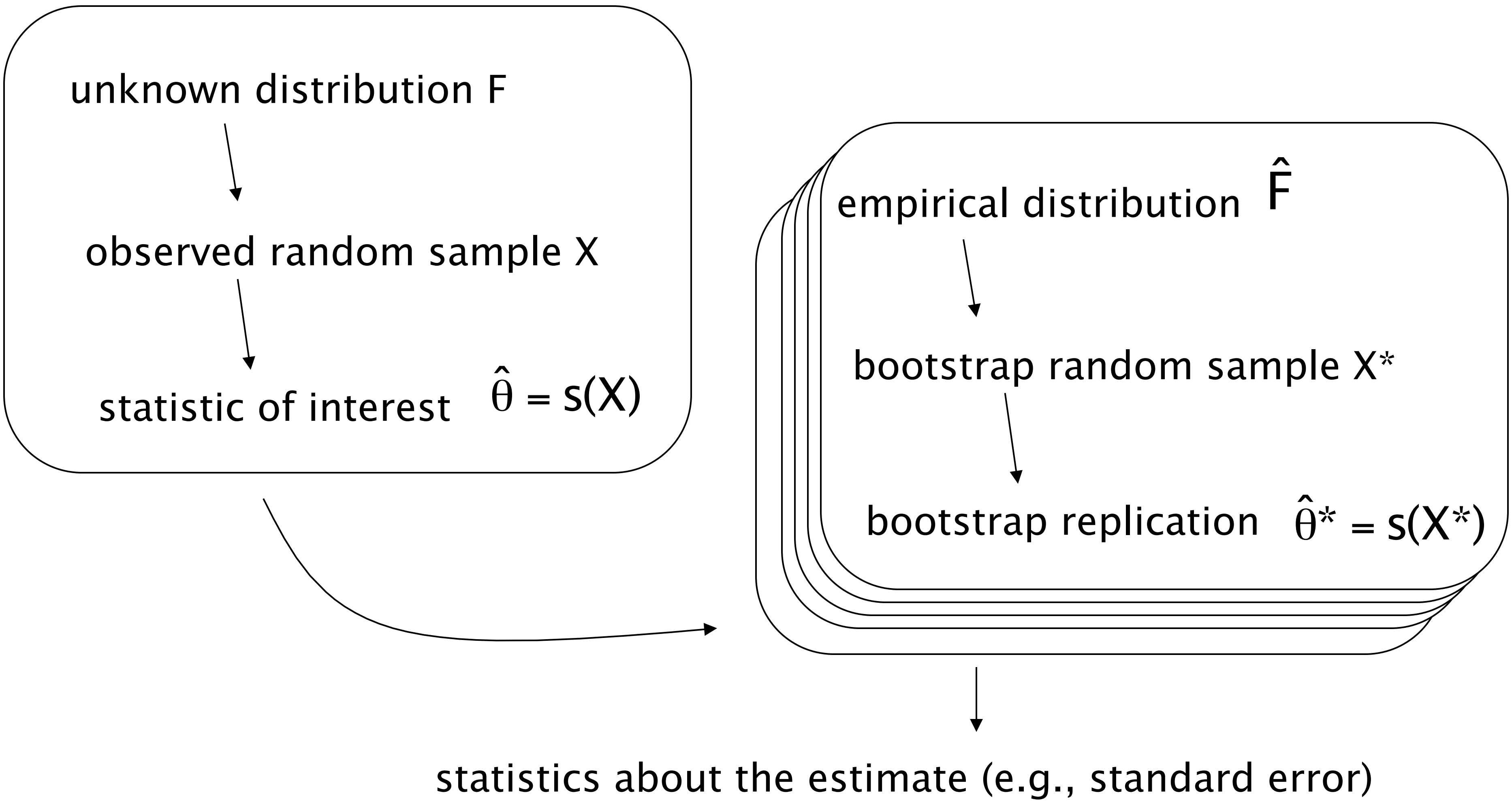
- We don't have (and can't get) P
  - We don't know F, the true distribution over X
- We **do** have S (the sample)
  - We **do** know $\hat{F}$, the sample distribution over X

- Estimating a statistic: use $\hat{F}$ for F

The Bootstrap

- Simulates the sampling distribution
- Proposed by Efron in 1979
  - Anticipated by permutation tests, jackknife, cross-validation
- From original sample of size n, draw B samples of size n *with replacement* and calculate the statistic on each

Bootstrap world



Bootstrap sample

- X = [3.0, 2.8, 3.7, 3.4, 3.5]
- X* could be:
  - [2.8, 3.4, 3.7, 3.4, 3.5]
  - [3.5, 3.0, 3.4, 2.8, 3.7]
  - [3.5, 3.5, 3.4, 3.0, 2.8]
  - …

Draw n elements with replacement.

Reflection

- Imagine doing this with a pencil and paper.
- The bootstrap was born in 1979.
- Typically, sampling is costly and computation is cheap.
- In (empirical) CS, sampling isn't even necessarily all that costly.

Paired-Sample Permutation Test

- “Is ranking algorithm F better than G?”
- Match the ranked list for each query
- Swap the lists in each pair with 0.5 probability
- Replicate this process B=100s of times
- Calculate the mean difference in metric (MAP, NDGC, P@10, etc.) for each replicate
- P-value: $\#\{\theta^*(b) >= \theta\}/B$

A Two-Sample Bootstrap Test

- "Is this system better at long than short queries?"
- H_0: equality of two distributions F and G, samples **z** and **y** or size *n* and *m*
- Test statistic θ = **z** – **y**
- Pool samples into x; draw *B* samples with replacement; call first *n* in resample observations from F
- p-value: #{θ*(b) >= θ}/B

Sign Test

- Ignores magnitude of differences
- Null hypothesis for this test is that
  - $P(B > A) = P(A > B) = ½$
  - number of pairs where B is "better" than A would be the same as the number of pairs where A is "better" than B
- Test statistic is number of pairs where B>A
- For example data,
  - test statistic is 7, p-value = 0.17
  - cannot reject null hypothesis

Setting Parameter Values

- Retrieval models often contain parameters that must be tuned to get best performance for specific types of data and queries
- For experiments:
  – Use *training* and *test* data sets
  – If less data available, use *cross-validation* by partitioning the data into *K* subsets
  – Using training and test data avoids *overfitting* – when parameter values do not generalize well to other data

Finding Parameter Values

- Many techniques used to find optimal parameter values given training data
  - standard problem in machine learning
- In IR, often explore the space of possible parameter values by *brute force*
  - requires large number of retrieval runs with small variations in parameter values (*parameter sweep)*
- *Learning to rank* techniques are efficient procedures for finding good parameter values with large numbers of parameters

Online Testing

- Test (or even train) using live traffic on a search engine
- Benefits:
  - real users, less biased, large amounts of test data
- Drawbacks:
  - noisy data, can degrade user experience
- Often done on small proportion (1-5%) of live traffic

Summary

- No single measure is the correct one for any application
  - choose measures appropriate for task
  - use a combination
  - shows different aspects of the system effectiveness
- Use significance tests
  - t-test, permutation test
- Analyze performance of individual queries

Query Summary