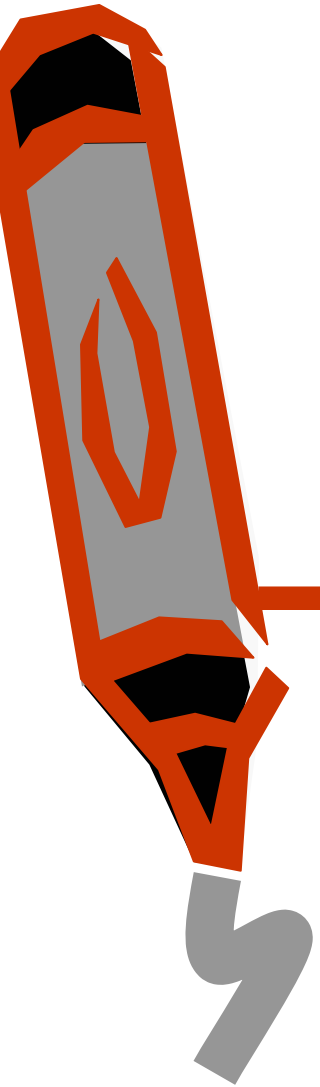
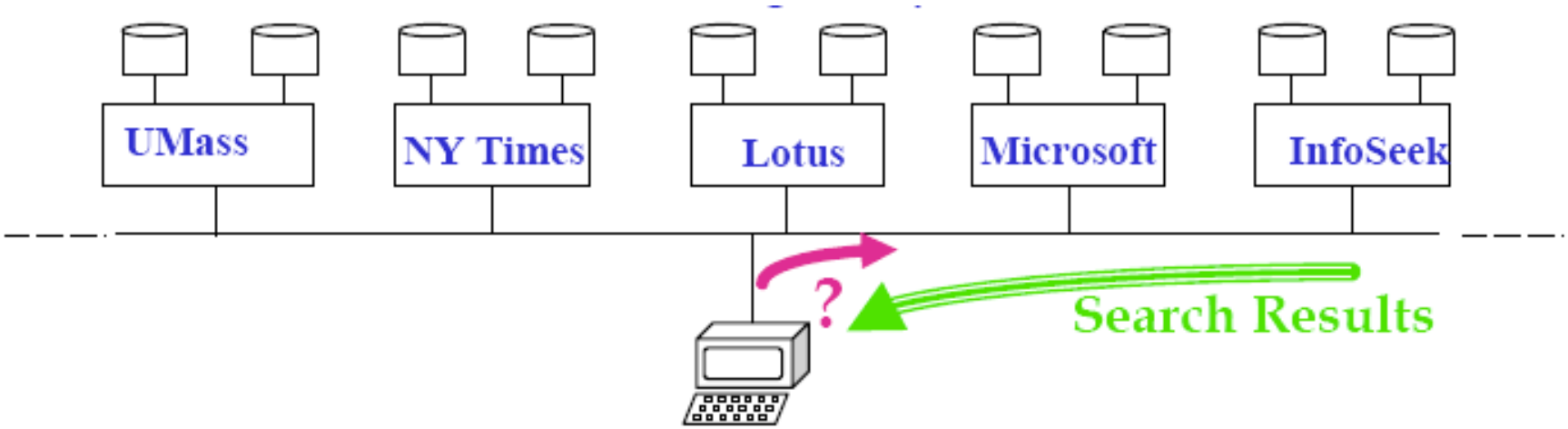


distributed IR

distributed IR

- IR is usually viewed as searching a single collection of documents
- What is a collection?
  - A single source, e.g., Wall Street Journal? (What time period?)
  - A single location, e.g., the UMass Physical Sciences Library?
  - A set of libraries, e.g., all UMass Amherst libraries?
- Distributed IR: searching when there is more than one collection
  - Local environments, e.g., a large collection is partitioned
  - Wide-area environments, e.g., corporate network, Internet



distributed IR

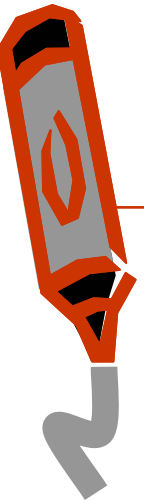
- Partition large collections across processors
  - To increase speed
  - Because of political or administrative requirements
- Networks, with hundreds or thousands of collections
  - Consider number of collections indexed on the Web
- Heterogeneous environments, many IR systems
- Economic costs of searching everything at a site
- Economic costs of searching everything on a network

issues

- Site description
  - Contents, search engine, services, etc
- Collection selection
  - Deciding which collection(s) to search
  - ranking collections for a query
  - selecting the best subset from a ranked list
- Searching
  - Interoperability, cooperativeness
- Result merging: Merging a set of document rankings
  - different underlying corpus statistics
  - different search engines with different output information
- Metrics:
  - Generality, effectiveness, efficiency, consistency of results, amount of manual effort, etc

collection selection approaches

- Single Site / LAN / Few Sites
  - Select everything
  - Group manually (and select manually)
  - Rule-based selection
  - Relevant document distribution (RDD)
  - Query Clustering
  - Query Probing
- Many Sites / WAN / Internet
  - Content-based collection ranking (and selection)



collection selection: select all

- Found in LANs, e.g. where a large collection is partitioned
- Works well with the unranked Boolean model
  - Result set is the union of all search results

- Can work with statistical models
  - Merge-sort all search results, to obtain merged ranked list
  - But, scores from different databases aren't comparable, due to different corpus statistics, e.g., *idf, avg_doclen*
  - Scores can be made comparable by imposing one set of corpus statistics on all databases, e.g., global statistics, first database

- Ignores costs of searching collections
  - e.g., time, money

- Does not scale to WAN / Internet
- Some parallelism by distributing search

collection selection : manually

- Collections are organized into groups with a common theme
  - e.g., financial, technology, appellate court decisions
- User selects which group to search

- Found in commercial service providers
  - e.g., Dialog, West
- Groupings determined manually
  - time consuming, inconsistent groupings, coarse groupings, not good for unusual information needs

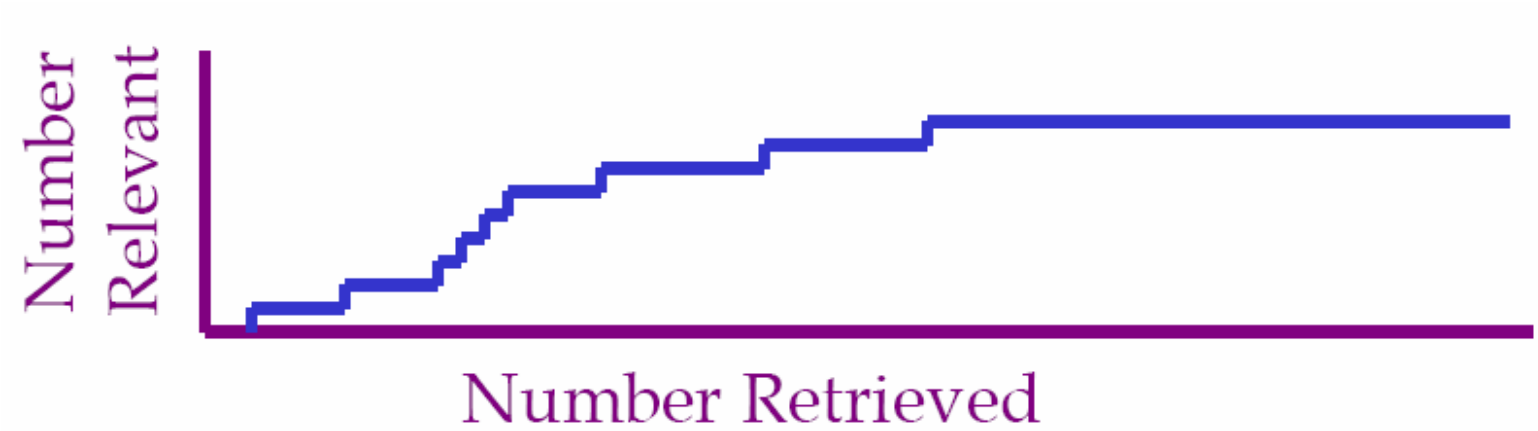
- Groupings determined automatically
  - Broker agents maintain a centralized cluster index by periodically querying collections on each subject
  - automatic creation, better consistency, coarse groupings
  - not good for unusual information needs

collection selection : rule-based

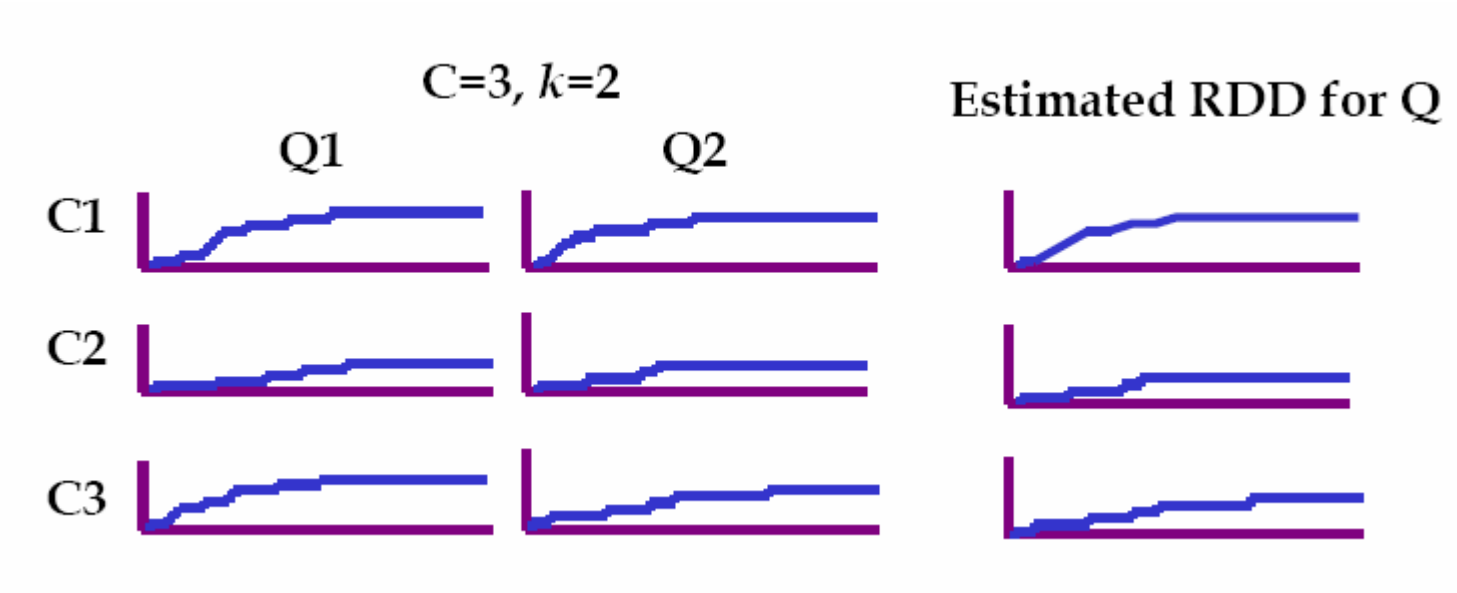
- The contents of each collection are described in a knowledge-base
  - few details provided by authors of such systems
- A rule-based system selects the collections for a query
  - few details provided by authors of how this works
- CONIT, a research system, never deployed widely
  - tested on static and homogeneous collections
  - time consuming to create
  - inconsistent selection if rules change
  - coarse groupings so not good for unusual information needs

collection selection: RDD

- RDD=relevant document distribution
- Build a database of queries and the distribution of relevant documents for each query in each collection (somehow)
- For a new query
  - Find the *k* nearest neighbors in the database (similar past queries)
  - Average the *k* relevant document distributions for each collection *C*, to get an estimate of how relevant documents are distributed
  - Use a maximization procedure over the |*C*| relevant document distributions to decide how much to retrieve from each collection



collection selection: RDD



- The estimated RDD is the average of the RDDs for the *k* most similar queries

- Note: Effectiveness depends on training queries being similar to expected queries

collection selection: query clustering

- Build a database of queries with relevance judgments for each query in each collection (somehow)

- Cluster the training queries for each collection, based on the *total* number of documents retrieved in common (*not* just relevant docs)

- Determine the average number of relevant documents in the top *L* retrieved for each query cluster in each collection

- For a new query
  - find the nearest query cluster centroid *q* in each collection *c*
  - the estimated utility of *c* is the average number of relevant retrieved by *q*
  - retrieve documents from *c* in proportion to estimated utility

- This method is designed for environments where there is little variety in queries and new collections are not added often

collection selection: query probing

- Send a lightweight *probe query* to each collection
  - each collection responds with term frequency information
    - e.g., collection size, *df* for proximity, *df* for co-occurrence, *df* for individual terms
  - client ranks collections, selects top *n*

$$S_i = c_1 df_1 + c_2 df_2 + 10 df_{co-occur} + 100 df_{prox}$$

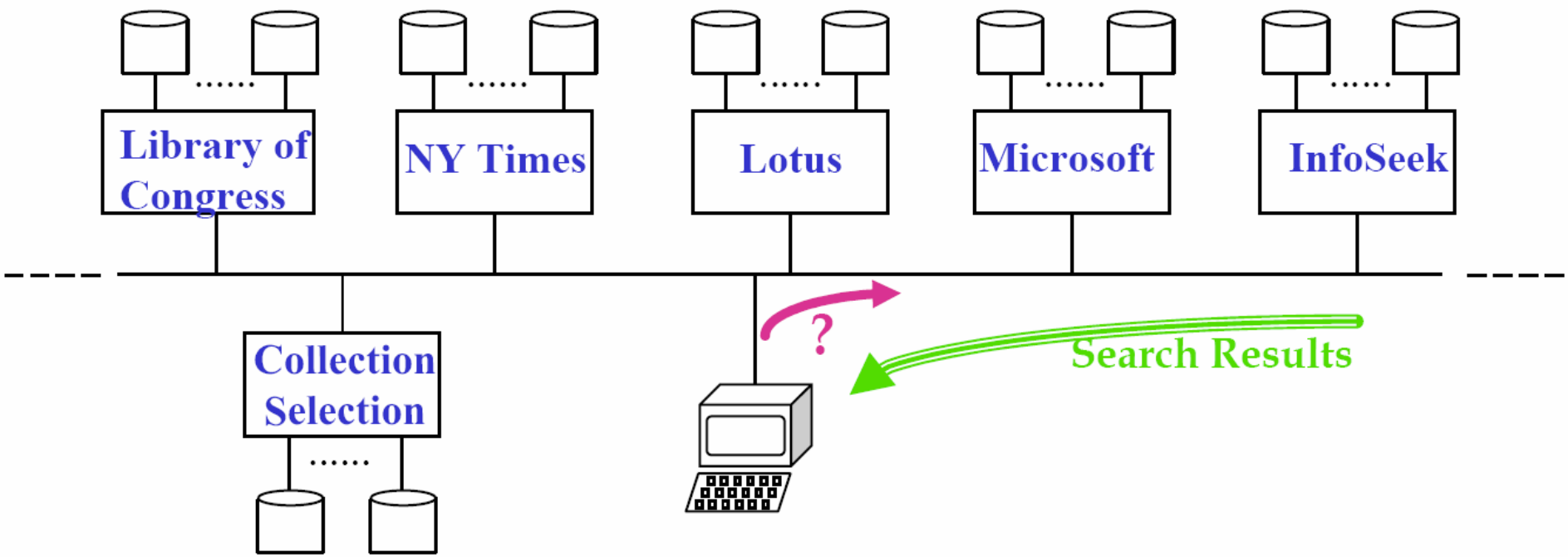
- Assumptions:
  - processing tiny probes is considerably cheaper than full queries
    - e.g., because probe is shorter
    - e.g., because probe does no ranking
  - client can estimate collection utility based on a few terms
  - probe bandwidth and latency is low

many collections

- Assumptions:
  - It is too expensive to search every collection
    - there may be hundreds or thousands
    - they may be dispersed widely
    - it may cost money to search some collections
  - Collections are managed independently
    - can't depend upon a collection to describe itself well
  - Collections are homogeneous and heterogeneous
  - Queries are heterogeneous
- Solution:
  - Search a centralized resource
  - Represent each collection by its vocabulary

many collections

- Search a centralized resource
- Represent each collection by its vocabulary



statistical profile of collection

Word	Frequency	Word	Frequency
Ireland	8	Prime	2
said	5	peace	2
past	5	only	2
Northern	5	negotiators	2
I	4	Mitchell	2
Blair	4	Minister	2
agreement	4	long	2
have	3	British	2
years	2	bloody	2
today	2	Ahern	2
reached	2	accord	2
quickly	2	30	2
prize	2		

statistical profile of collection

1987 WSJ (132 MB)	1991 Patent (254 MB)	1989 AP (267 MB)
stobb (1)	sto (1)	sto (7)
stochast (1)	stochast (21)	sto1 (4)
stock (46704)	stochiometr (1)	sto3 (1)
stockad (5)	stociometr (1)	stoaker (1)
stockard (3)	**stock (1910)**	stoand (1)
stockbridg (2)	stockbarg (30)	stober (6)
stockbrok (351)	stocker (211)	stocholm (1)
stockbrokag (1)	stockholm (1)	**stock (28505)**
stockbrokerag (101	stockigt (4)	stock' (6)
stockdal (8)	stockmast (3)	stockad (35)
stockhold (970)	stockpil (7)	stockard (12)

collection selection : rank and select (GIOSS)

- GIOSS = Glossary-Of-Servers Server
- Estimate the number of potentially relevant documents in collection C for Boolean AND query Q as:

$$|C| * \prod_{t \in Q} \frac{df_t}{|C|}$$

df_t = number of documents in C containing term t
$|C|$ = number of documents in C

- Requires that each collection C have an entry in a centralized index
  - centralized index is small, easy to maintain
- Automatic creation, consistent, dynamic grouping, good for most information needs

collection selection: rank and select

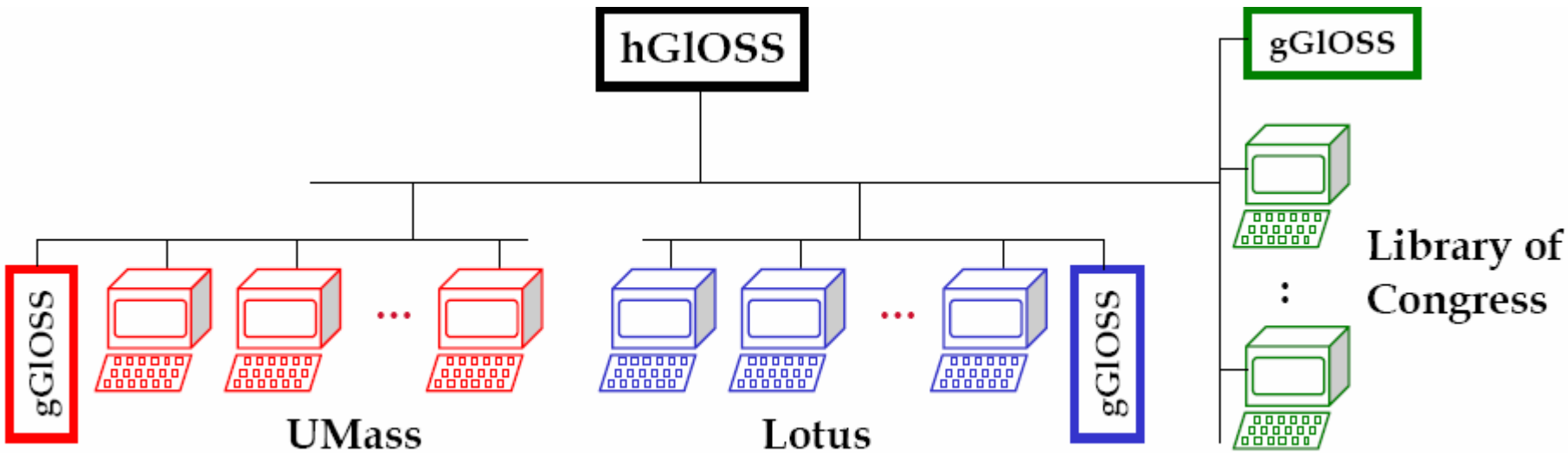
gGIOSS = generalized GIOSS

•Extends the GIOSS approach to the vector space model
• Each collection is represented by its centroid vector
• Standard inner product similarity measure of query to
each collection
• Rank collections accordingly

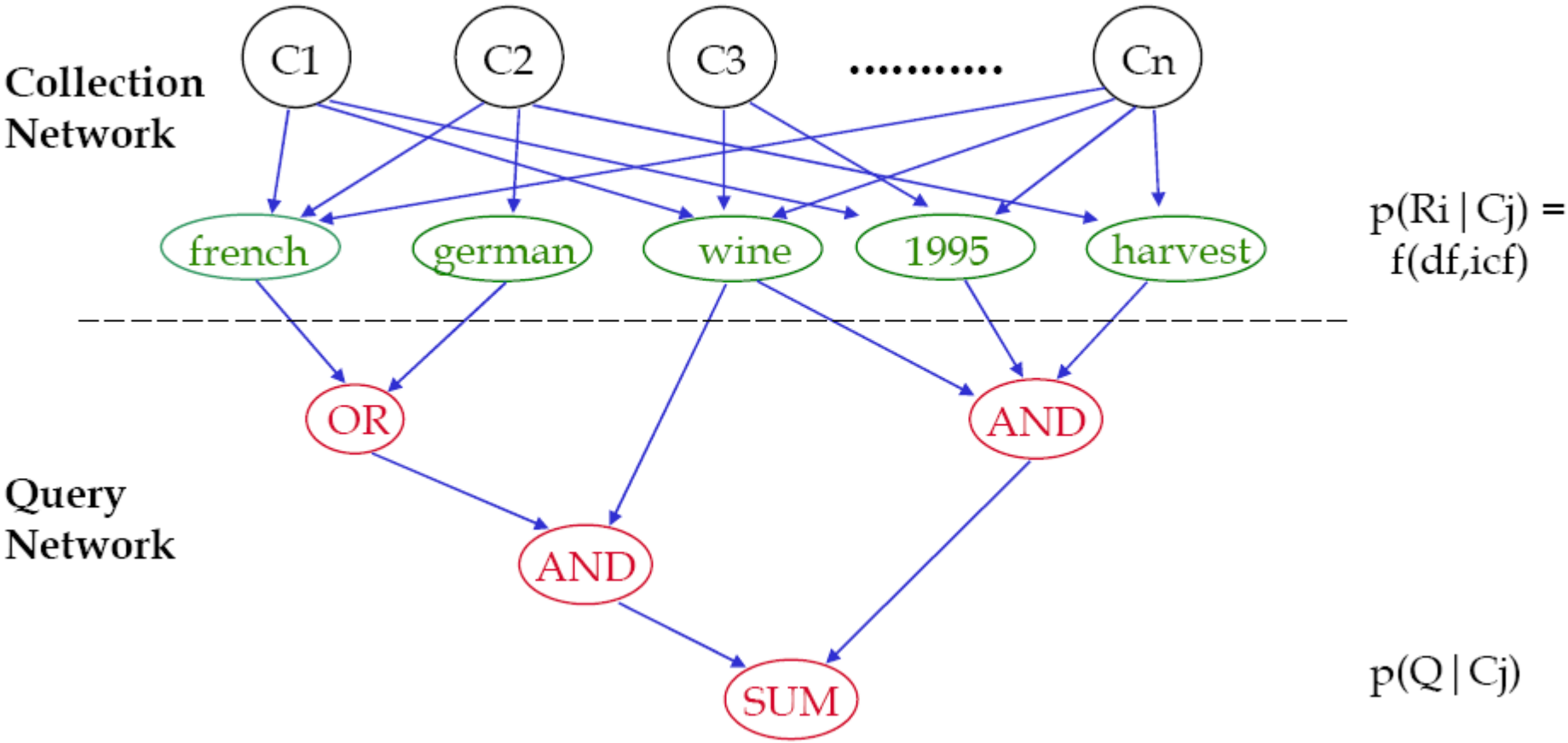
collection selection: rank and select

hGlOSS = hierarchical GlOSS

•Extends the gGlOSS approach to sets of gGlOSS indexes
• Each gGlOSS index is represented by its centroid vector



collection selection: inference nets



Estimating $p(R_i|C_j)$

- For documents, p(Ri|Dj):
  - ntf = tf / (tf + 0.5 + 1.5 * dl / avg_dl)
  - idf = 0.4 + 0.6 * log ((D + 0.5) / df) / log (D + 1)
- Mapping:

Documents	Collections
term frequency (tf)	document frequency (df)
document frequency (df)	collection frequency (cf)
document length (dl)	collection length (cl)
number of documents (D)	number of collections (C)

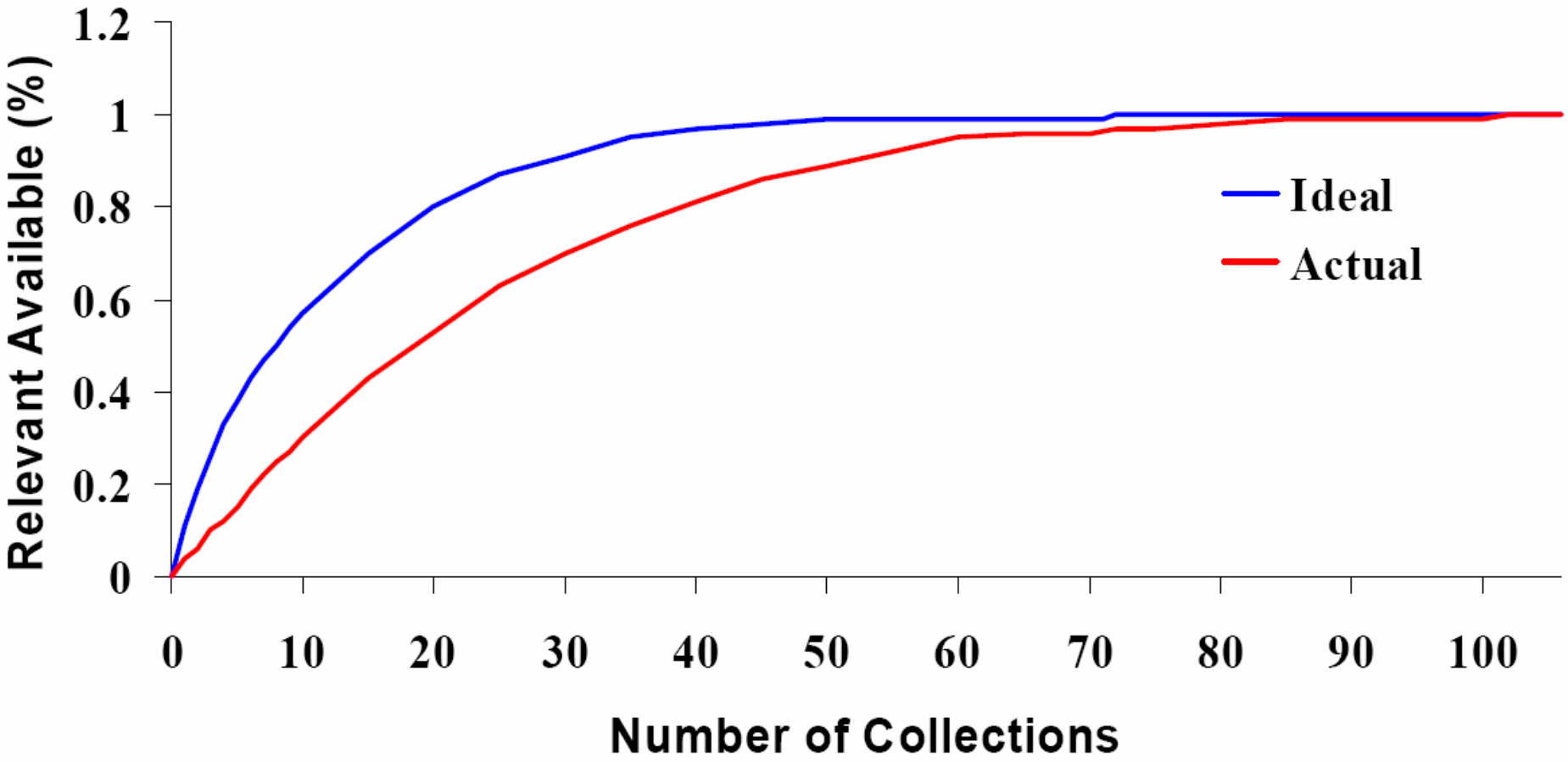
- For collections, p(Ri|Cj):
  - ntf = df / (df + 150 + 50 * cl / avg_cl)
  - icf = 0.4 + 0.6 * log((C + 0.5) / cf) / log (C + 1)

collection ranking

Query 63: Identify a machine translation system being developed or marketed in any country. Identify the developer or vendor, name the system, and identify one or more features of the system.

Rank	Collection	Score	#RelDocs
1	1989-90 Ziff Davis	0.571	153
2	1989-90 Ziff Davis	0.569	34
3	1991-92 Ziff Davis	0.563	77
4	1991 Patent	0.407	0
5	1989 AP	0.404	1
6	1988 AP	0.401	4
7	1988 WSJ	0.399	1
8	1987 WSJ	0.399	0
9	1988 Federal Register	0.341	0
10	1989 Federal Register	0.337	0

collection ranking



which collections to search

- Numerous options
  - Top *n*
  - Top group (clustering)
  - Cost-based selection
- Not discussed further

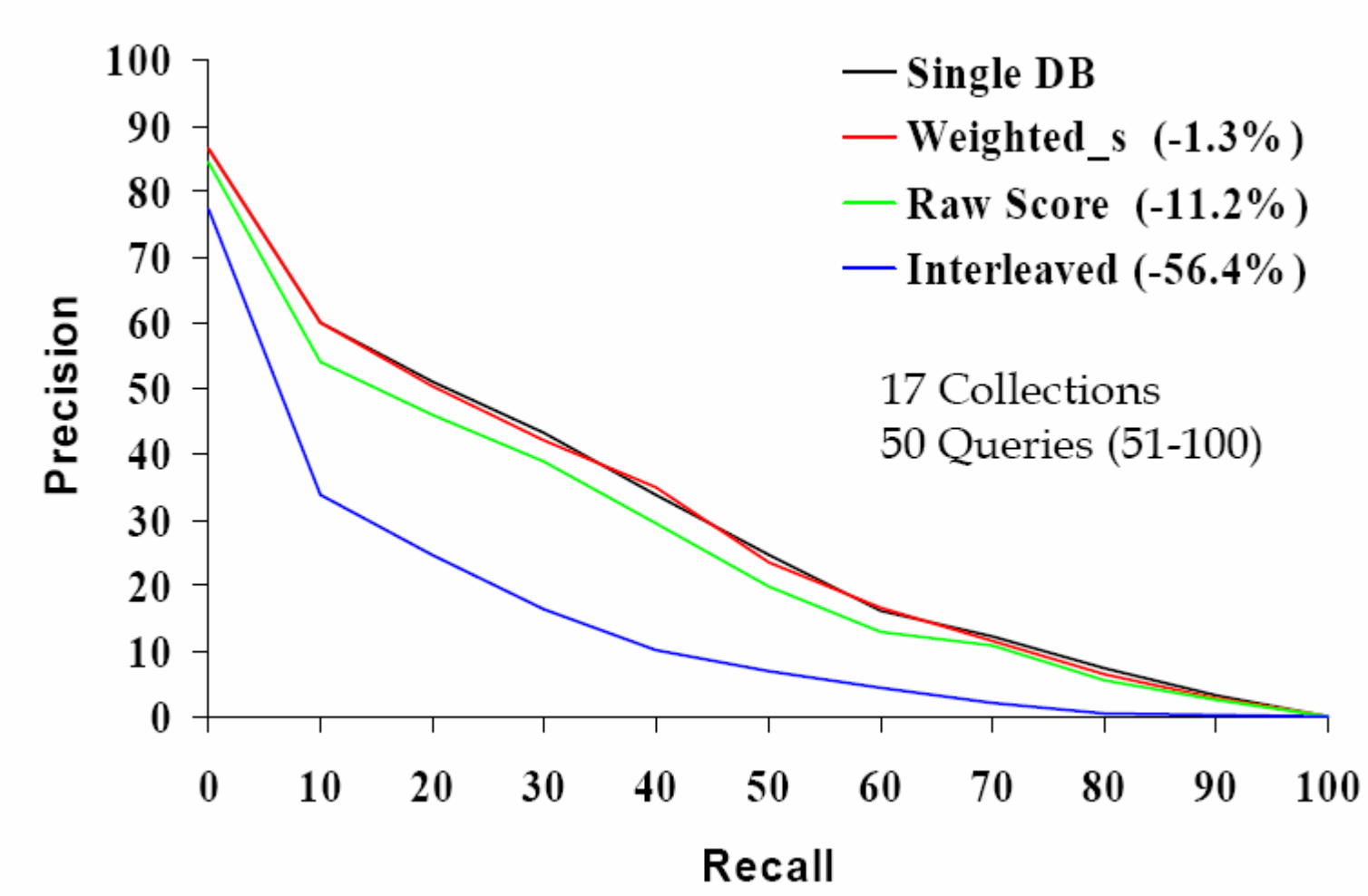
result merging

- Round-robin, and weighted round-robin
- Recompute scores at client
  - each collection sends back statistics with documents
    - e.g., *tf, doclen, maxtf, df,* etc
  - client computes a consistent *global* document score
  - simple, consistent, effective
  - ignores special indexing done in collection (if any)
  - patented by InfoSeek (now defunct)
- Heuristic reranking at client
  - client estimates a global document score, using local document score and information about its collection

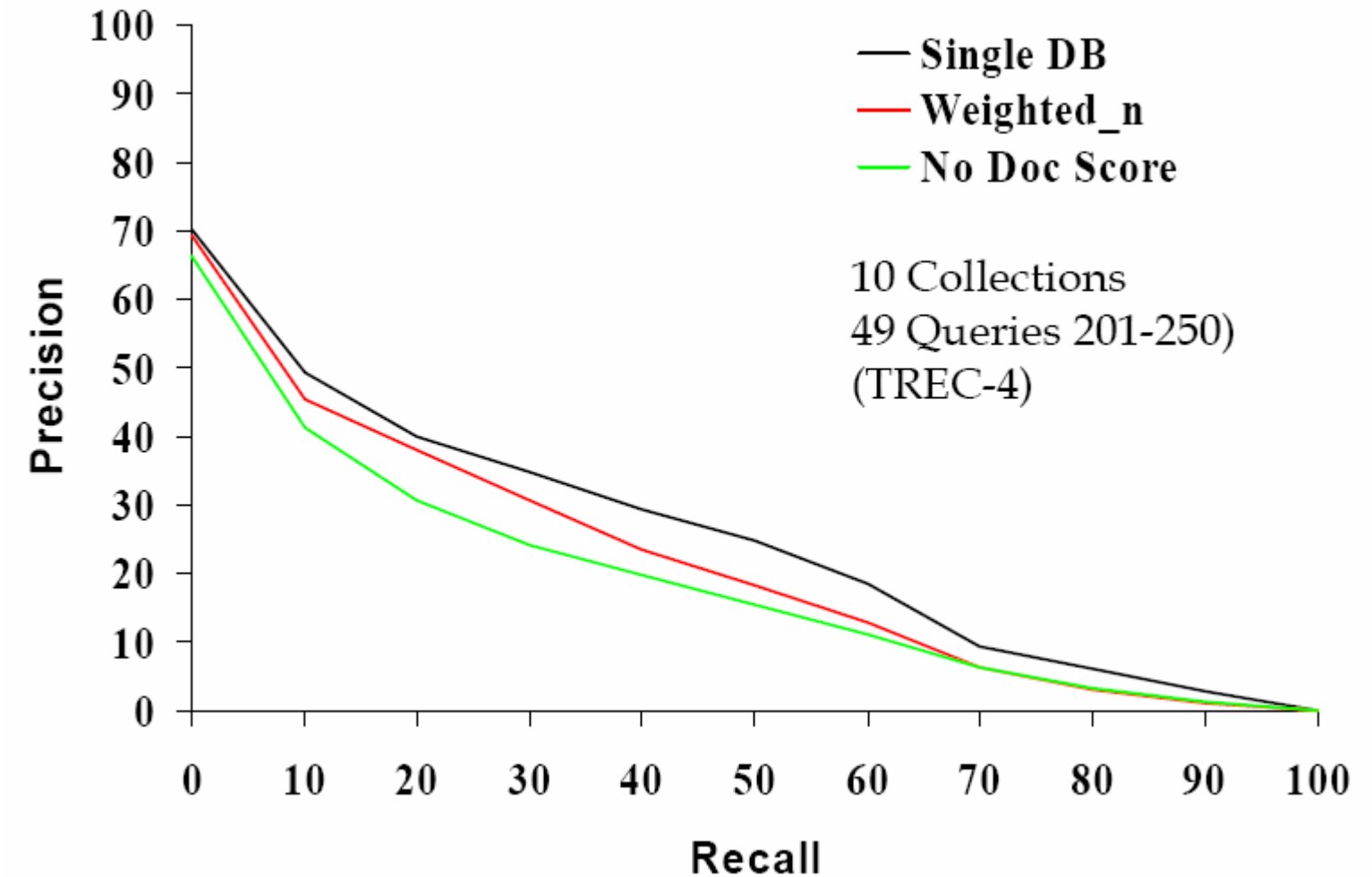
data fusion

- Weighted_s [Callan, SIGIR 1995]
  - Cj = C * (Cj – avgC) / avgC
  - Di’ = Di + Cj
  - No probabilistic interpretation
  - Collection scores were too similar
  - Sensitive to the number of collections ranked
- Weighted_n
  - Cj = (Cj - Cj_min) / (Cj_max - Cj_min)
  - Di’ = (Di + 0.4 * Di * Cj) / 1.4
- No Doc Score:
  - Di = 0.4 + 0.6 * (1 + MaxRankj - Ri) / MaxRankj
  - Di’ = (Di + 0.4 * Di * Cj) / 1.4

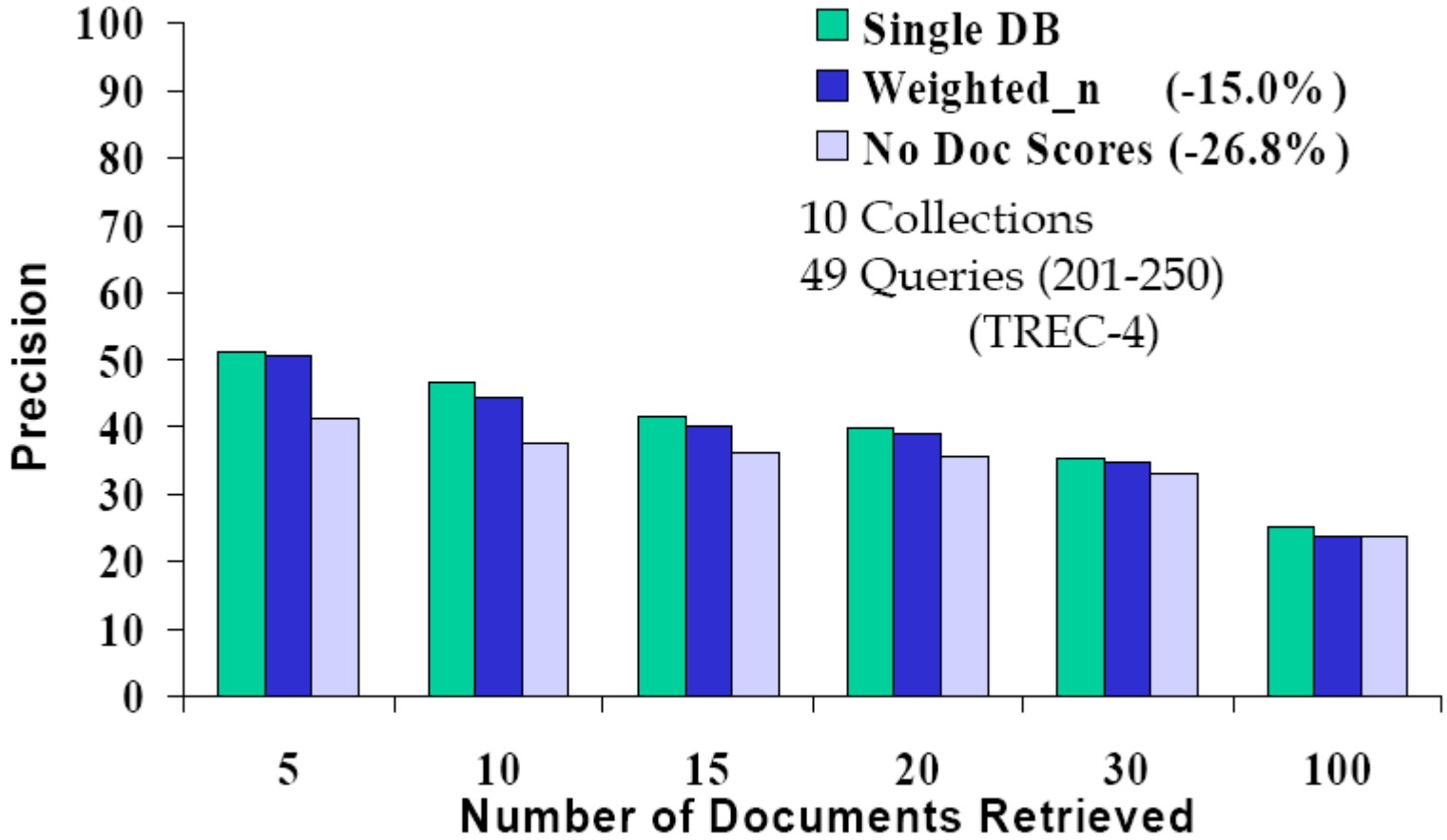
data fusion


Single DB
Weighted_s (-1.3%)
Raw Score (-11.2%)
Interleaved (-56.4%)
17 Collections
50 Queries (51-100)
Precision
Recall
0
10
20
30
40
50
60
70
80
90
100

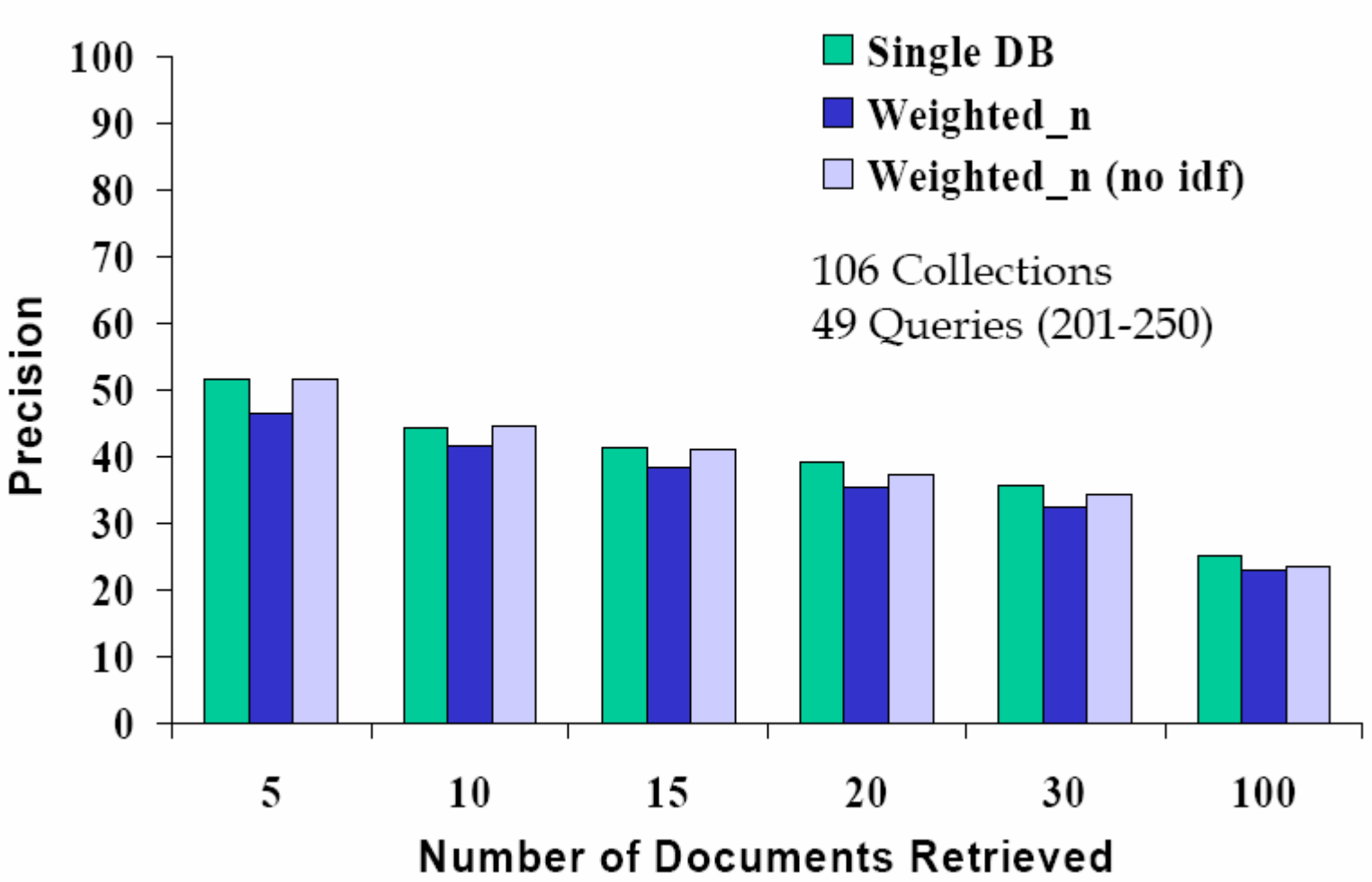
data fusion


Single DB
Weighted_n
No Doc Score
10 Collections
49 Queries 201-250)
(TREC-4)
Precision
Recall
0
10
20
30
40
50
60
70
80
90
100

data fusion


Single DB
Weighted_n (-15.0%)
No Doc Scores (-26.8%)
10 Collections
49 Queries (201-250)
(TREC-4)
Precision
100
90
80
70
60
50
40
30
20
10
0
5
10
15
20
30
100
Number of Documents Retrieved

data fusion


Single DB
Weighted_n
Weighted_n (no idf)
106 Collections
49 Queries (201-250)
Precision
100
90
80
70
60
50
40
30
20
10
0
5
10
15
20
30
100
Number of Documents Retrieved

result merging: RDD

- Recall RDD method for selecting collections
  - Based on clusters of past queries

- Determine cutoff λ_i for collections to maximize expected performance (based on RRDs)

- At rank *r*, select a collection randomly, based on number of documents remaining to be merged

- Assign to rank *r* the highest unselected document in that collection, and remove it from consideration
  - Weighted round-robin

- Advantages:
  - Respects original ranking within each collection
  - Merges based on estimated relevance of each collection (if number retrieved from each really reflects estimated relevance)
  - Does not use document scores, hence no need to normalize

distrib IR - state of art

- Representing collections by terms and frequencies is effective.

- Controlled vocabularies and schemas are not necessary.

- Collections and documents can be ranked with one algorithm
(using different statistics).
  - e.g., GlOSS, inference networks

- Rankings from different collections can be merged efficiently:
  - with precisely normalized scores (Infoseek's method), or
  - without precisely normalized document scores,
  - with only minimal effort, and
  - with only minimal communication between client and server.

- Large scale distributed retrieval can be accomplished now.

distrib IR - state of art

- Most error occurs in ranking collections, not merging
- Not clear that inverse collection frequency (*icf*) helps
  - but maybe we just don't have enough collections yet
- State of the art is about 100 collections
  - UMass has developed a 921 collection testbed
  - CMU is pushing this to thousands of collections
  - the major problem is *relevance judgements*, not data
- Significant improvements possible when fewer collections are searched, i.e. don't search Federal Register in TREC
  - Counter-intuitive at first blush (better results by ignoring data)
- Many open problems
- Language modeling approaches recently developed

open problems

- Multiple representations
  - stemming, stopwords, query processing, indexing
  - cheating / spamming
- Multiple retrieval algorithms
  - varying accuracy in rankings
- Thousands (millions?) of collections
- Effectiveness with 2-3 word queries
- How to integrate
  - relevance feedback
  - query expansion
  - browsing