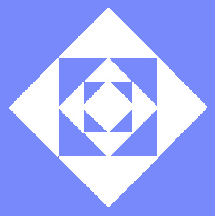


Experiments in TREC – the world championship for Search Engines

David Carmel

IBM Haifa Research Lab

Introduction

- In order to deliver high quality search engine, one must be able to measure and quantify search quality.
- Likewise, when contemplating which search system to buy, organizations would like to measure the quality provided by candidate systems.
- The evaluation of information retrieval (IR) systems is the process of assessing how well a system meets the information needs of its users.
- Evaluation criteria:
  - Effectiveness
    - How precise is the answer set returned by he system to the information need
  - Efficiency
    - Retrieval time, indexing time, index size
  - Usability
    - Learnability, novice use, expert use

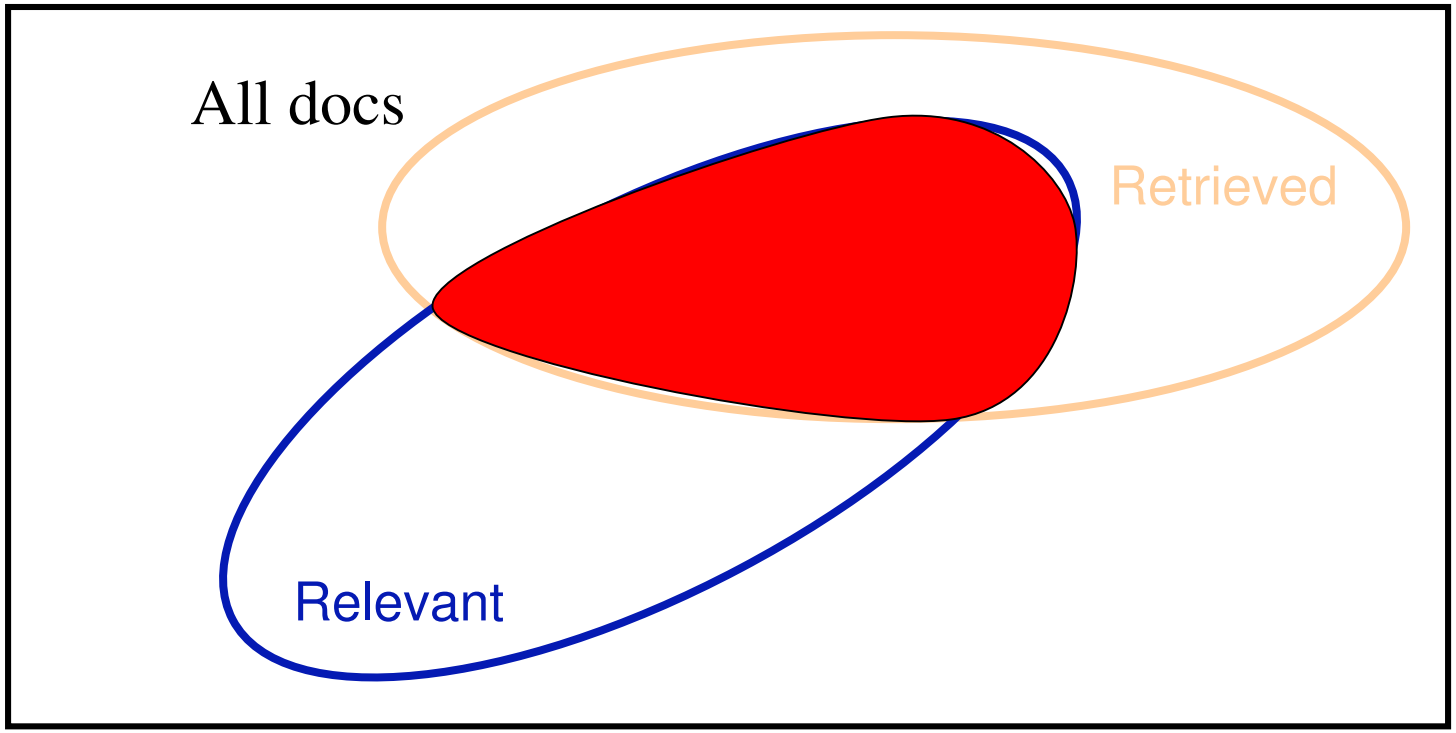
Standard Effectiveness Measures

- Recall
  - How much of what is relevant was found?
  - *All the true?*
- Precision
  - How much of what was found is relevant?
  - *Nothing but the true?*
- Fallout
  - How much of what was irrelevant was rejected?

Precision vs. Recall

$$Precision = \frac{|Rel\ Retrieved|}{|Retrieved|}$$

$$Recall = \frac{|RelRetrieved|}{|Rel\ in\ Collection|}$$



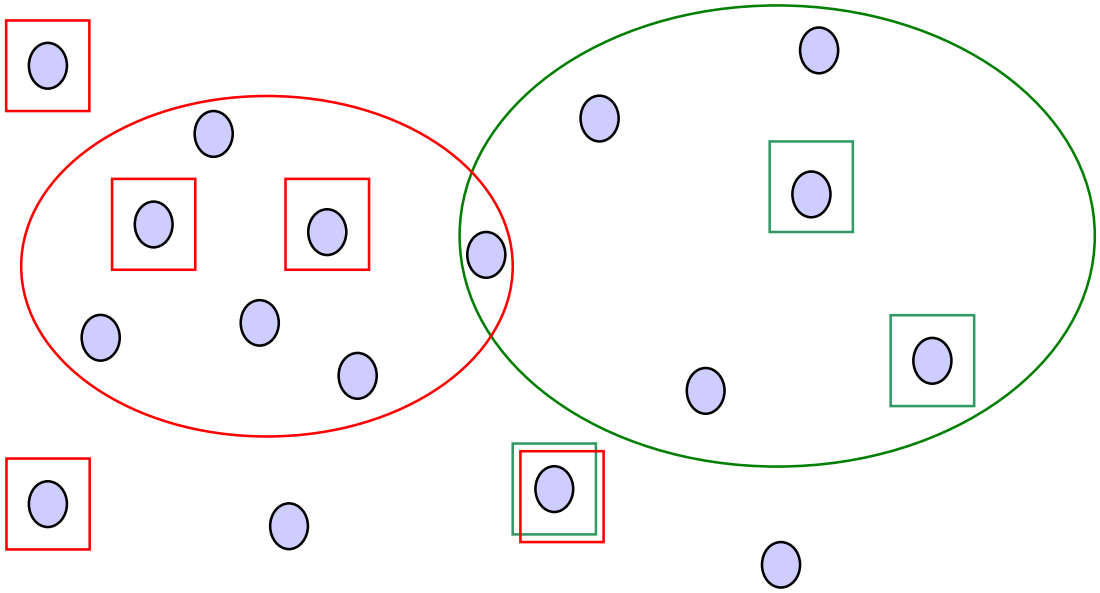
What is the meaning of Relevance?

- In what ways can a document be judged as relevant to a topic?
  - Answer precise question precisely.
  - Partially answer question.
  - Suggest a source for more information.
  - Give background information.
  - Remind the user of other knowledge.
  - Others ...
- Assumption
  - Binary relevance
    - Every document is either relevant or it is not
  - Unchanging, known relevance
    - The relevance of each doc to each topic is known
    - But only used for evaluation, not for retrieval!

Quality Evaluation by Relevance Judgments – the Cranfield Paradigm

- Fix a corpus (a collection of documents), and a set of topics.
- Scan the **entire corpus**, and mark each document as relevant/irrelevant with respect to each topic.
- Given a search system, run the queries derived from the topics, and score the system with respect to number (and rank) of returned relevant documents.
- Assign a score to the search system, aggregating its achievements over the set of topics.

Example



Quality Evaluation by Relevance Judgments – the Cranfield Paradigm

- Fix a corpus (a collection of documents), and a set of topics.
- Scan the **entire corpus**, and mark each document as relevant/irrelevant with respect to each topic.
- Given a search system, run the queries derived from the topics, and score the system with respect to number (and rank) of returned relevant documents.
- Assign a score to the search system, aggregating its achievements over the set of topics.

Quality Evaluation by Relevance Judgments, Using Pools

Following approach is useful in IR competitions, when several search systems are pitted against each other.

- Fix a corpus and a set of topics.
- For each topic:
  - Run all systems and form a “pool” of results – the union of the results returned by all systems.
  - Scan the **entire pool**, and mark each document as relevant/irrelevant with respect to the topic.
  - Re-examine the results returned by each individual system, and score the system with respect to number (and rank) of returned relevant documents.
- Assign a score to each search system, aggregating its achievements over the set of topics.

What is TREC?

- Text REtrieval Conference/Competition
  - Run by NIST (National Institute of Standards & Technology (USA))
  - 2004 was the 13th year - 14th TREC in early November
- Collection: >6 Gigabytes (5 CRDOMs), >1.5 Million Docs
  - Newswire & full text news (AP, WSJ, LA-Times, FT)
  - Government documents (federal register, Congressional Record)
  - Radio Transcripts (FBIS)
  - Web “subsets”
- Tracks
  - Ad-Hock
  - Web
  - Cross-language
  - Question Answering
  - tera-byte collection (500GB) …

TREC (cont)

- Documents are typically distributed in April
- Topics are distributed June 1
  - Queries are formed from topics using standard rules
- Top 1000 selections for each topic are due August 15
- Relevance judgments available in October
  - done only for those documents retrieved -- not entire collection!
  - Results are judged by “Information Specialists”
- Results presented in November each year

Sample TREC topic

<num> Number: 168
<title> Financing AMTRAK

<desc> Description:
A document will address the role of the Federal Government in financing the operation of the National Railroad Transportation Corporation (AMTRAK)

<narr> Narrative: A relevant document must provide information on the government's responsibility to make AMTRAK an economically viable entity. It could also discuss the privatization of AMTRAK as an alternative to continuing government subsidies. Documents comparing government subsidies given to air and bus transportation with those provided to aMTRAK would also be relevant.

Juru main Features

- Full text search engine purely written in Java
- Based on Guru (Maarek et. al 1989)
- Scoring model – a variation of SMART Lnu.ltu formula
- Proximity search (Lexical Affinities)

- Deployed in many IBM applications
  - Websphere Portal
  - Lotus WorkPlace
  - And many others

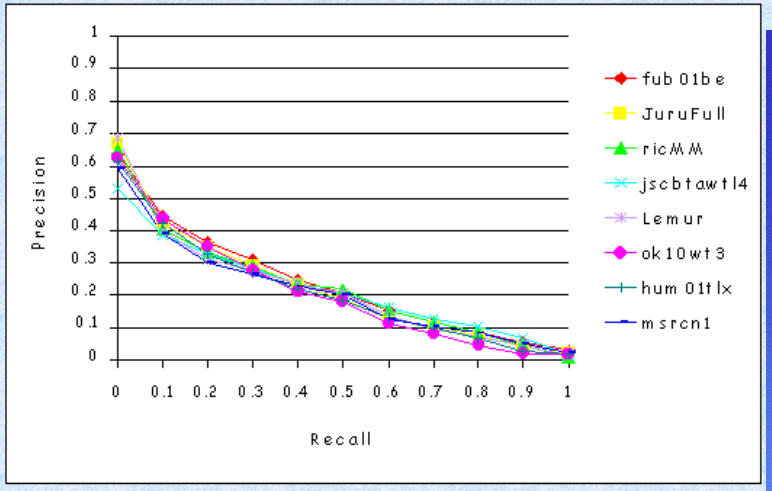
Juru at Trec 10 (2001)

- Participated in the ad-hoc task of the Web Track
- A document (HTML page) is indexed by:
  - its content
    - (3*title+2*strong_text+1*regular_text)
  - its set of “descriptions” (how others cite this document)
    - A description - the *anchor text* associated with a document’s in-link.
- Query formulation:
  - Topic titles only
  - Special query stop-word filtering
    - “find”, “describe”, “discuss”…
  - Extract *lexical affinities* in addition to individual terms
    - Lexical affinities *(LAs)* - pairs of words found in close proximity to each other
  - Example: query 501: “Deduction and Induction in English”

deduct,	induct,	english,
deduct*induct,	deduct*english,	english*induct

WebTrack 2001 – Juru results

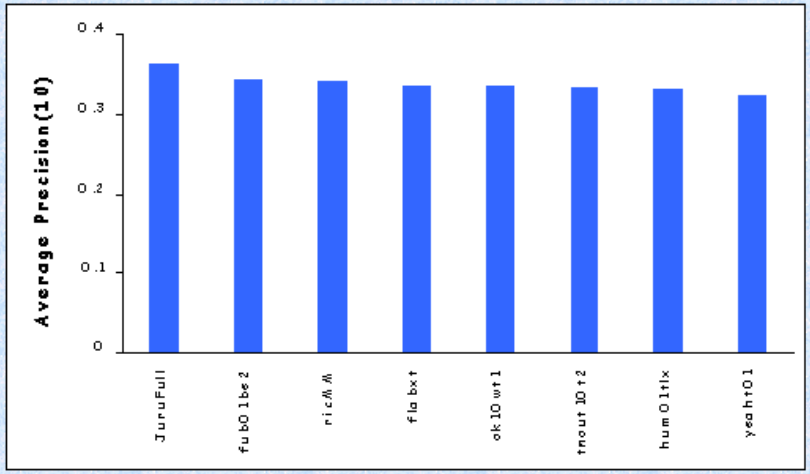
Automatic Ad Hoc, Title-Only



Top 8 groups by mean average precision of best run.

Text REtrieval Conference

Automatic Ad Hoc, Title-Only

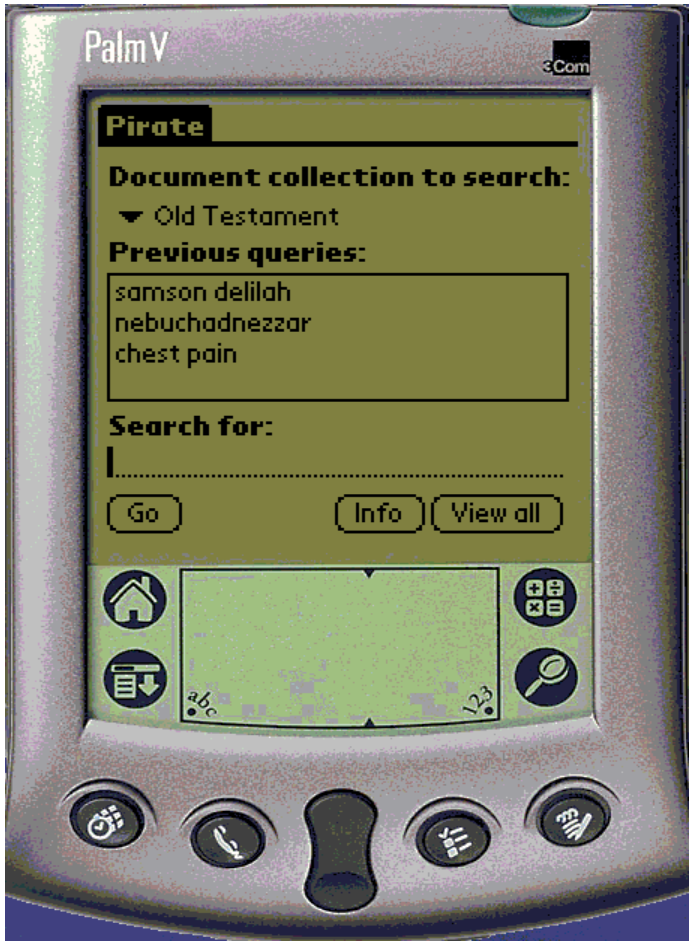


Top 8 groups by average Precision(10) of best run.

Text REtrieval Conference (TREC)

Static Index Pruning - Motivation

- Given a collection of documents, and a memory of limit size.
- Find - the optimal search index for this collection, restricted to the given size.
  - Prune the index in a away that a human cannot distinguish between the results of a search engine whose index is pruned and one whose index is not pruned.
  - Based on our paper: *Static Index Pruning for IR Systems, SIGIR 2001*



Experiments

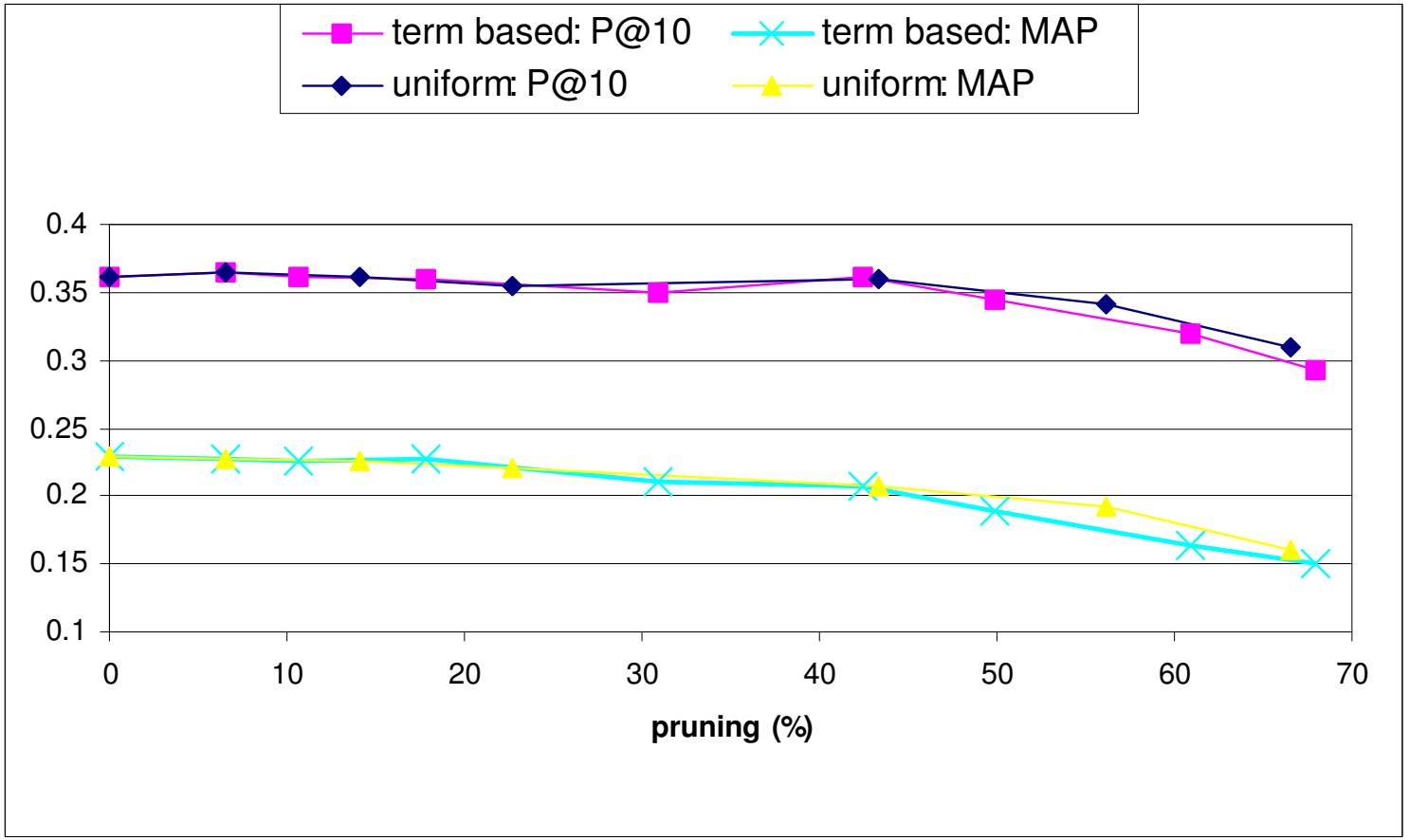
- Data – Web Track (10G)
- A sequence of pruned indices were created by invoking the pruning algorithms on Juru's original index.
- Measure the difference in performance between the original index and the pruned index in terms of
  - Similarity (between the top k results)
    - Symmetric difference
    - A variation of Kendall's-tau (Fagin et al. 2001)
  - Precision:
    - P@K
    - Avg. Precision

The effect of pruning on precision (WebTrack)

ε	Index size	Pruning (%)	MAP	P@ 10
0	3.53 GB	0	0.211	0.362
0.05	3.15 GB	10.7	0.207	0.362
0.1	2.9 GB	17.8	0.205	0.360

Best P@10 and second best MAP in WebTrack

The impact of pruning on precision



Trec-10: Summary

- We can reduce the index size by 40% while attaining the same P@10 and with only a slight decrease in the mean average precision.
- Pruning enables search on large text collections on low-end machines.
- Web search engines can significantly decrease the burden of storing extremely large indices.

Juru at WebTrack 2002: Topic Distillation

*"Topic distillation involves finding a list of **key resources** for a particular topic.*

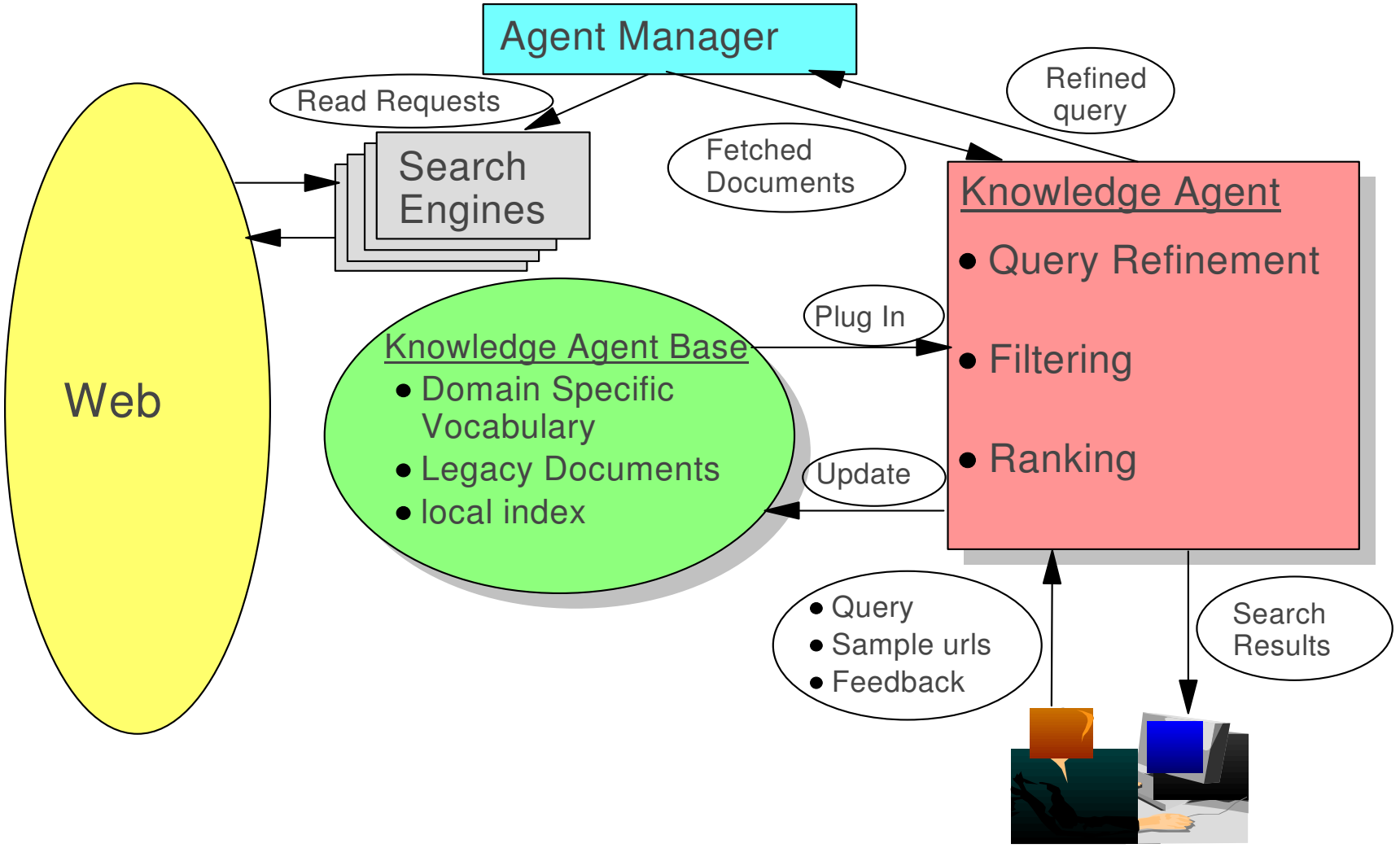
*A key resource is a page which, if someone built me a (short) list of key URLs in a topic area, **I would like to see included**. … "*

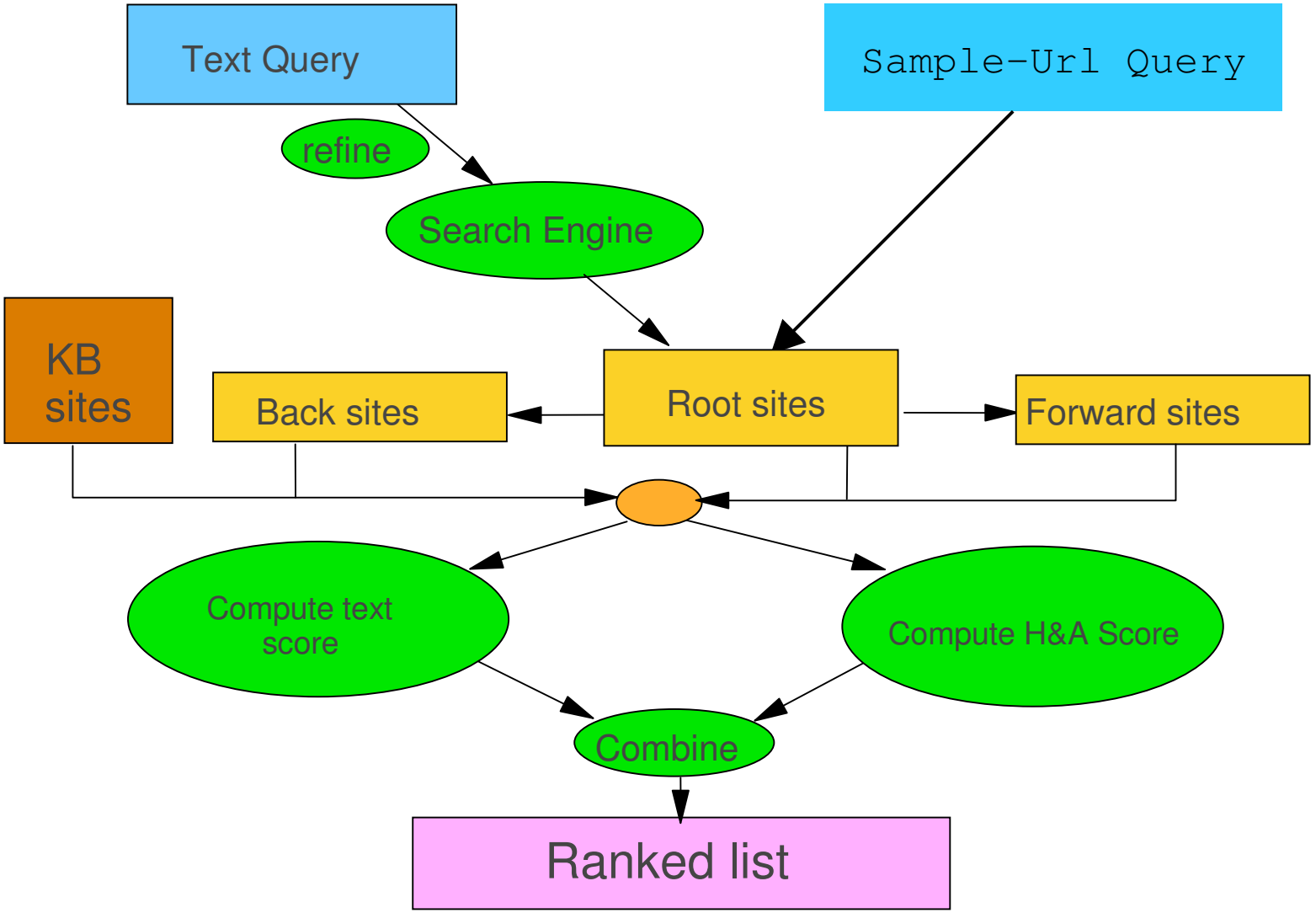
The WebTrack guidelines page
http://www.ted.cmis.csiro.au/TRECWeb/guidelines_2002.html

Knowledge Agents

- Provide domain-specific search in the context of dynamic domains
- Take the role of an expert librarian who assists in advanced search by utilizing domain-specific knowledge.
- Domains are defined by the users and can be of any granularity and specialty.
- Knowledge is acquired from search results
- This knowledge is persistently saved and can improve future searches within the agent's domain.

KA Architecture





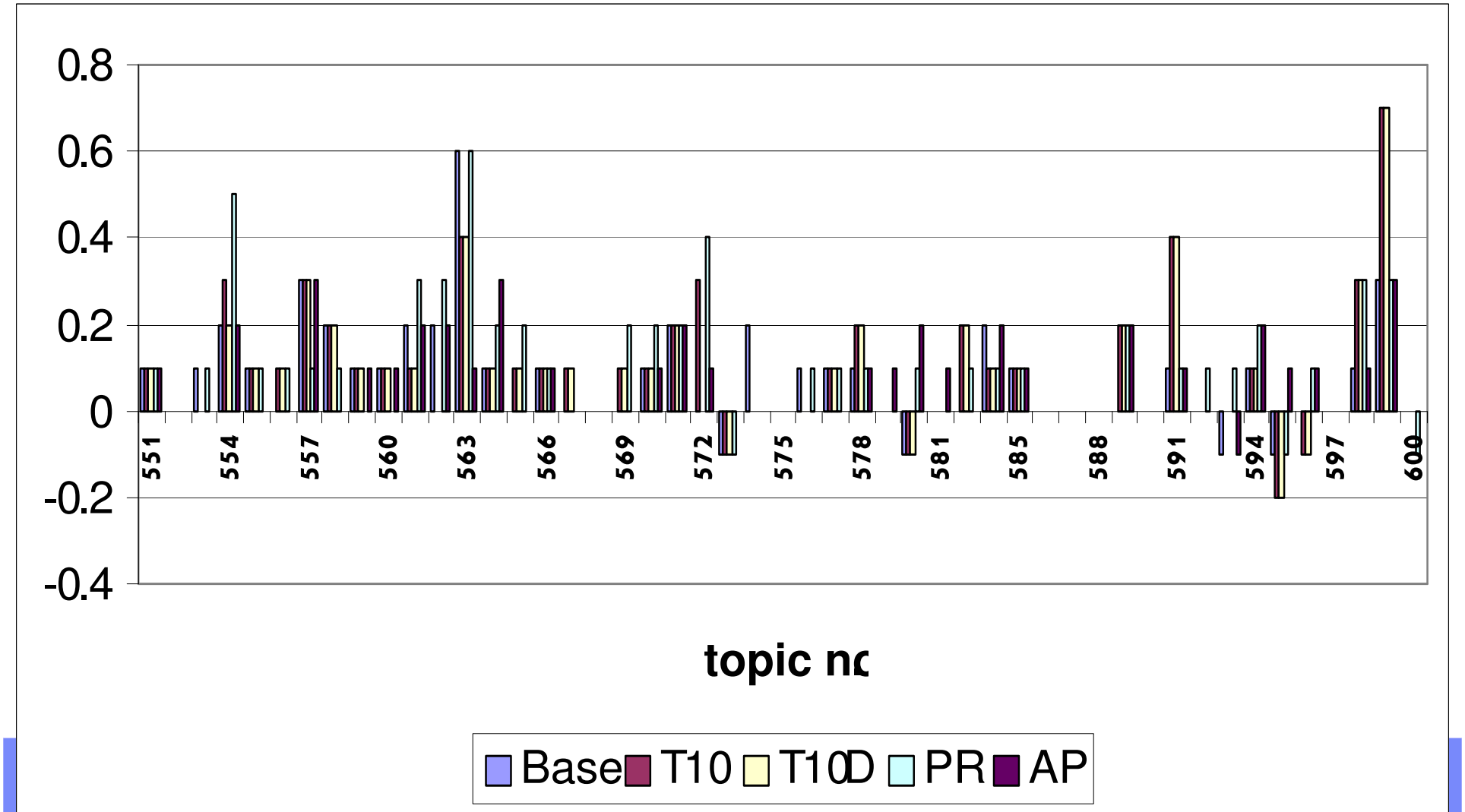
Topic Distillation using Knowledge Agents

- For each topic we trained a KA (one training query)
- Two types of training queries:
  - The topic title (free text)
  - AND of all title terms
- Root set collected using the *Juru* search engine over the ".gov" domain
- Forward/Backward sets collected using the".gov" connectivity data
- Textual scoring is based on *Juru* scores
- Two types of topological scoring:
  - Knowledge agents H&A scoring mechanism
  - Static ranking based on number of in-links
- Top 100 pages inserted into the agent's KAB

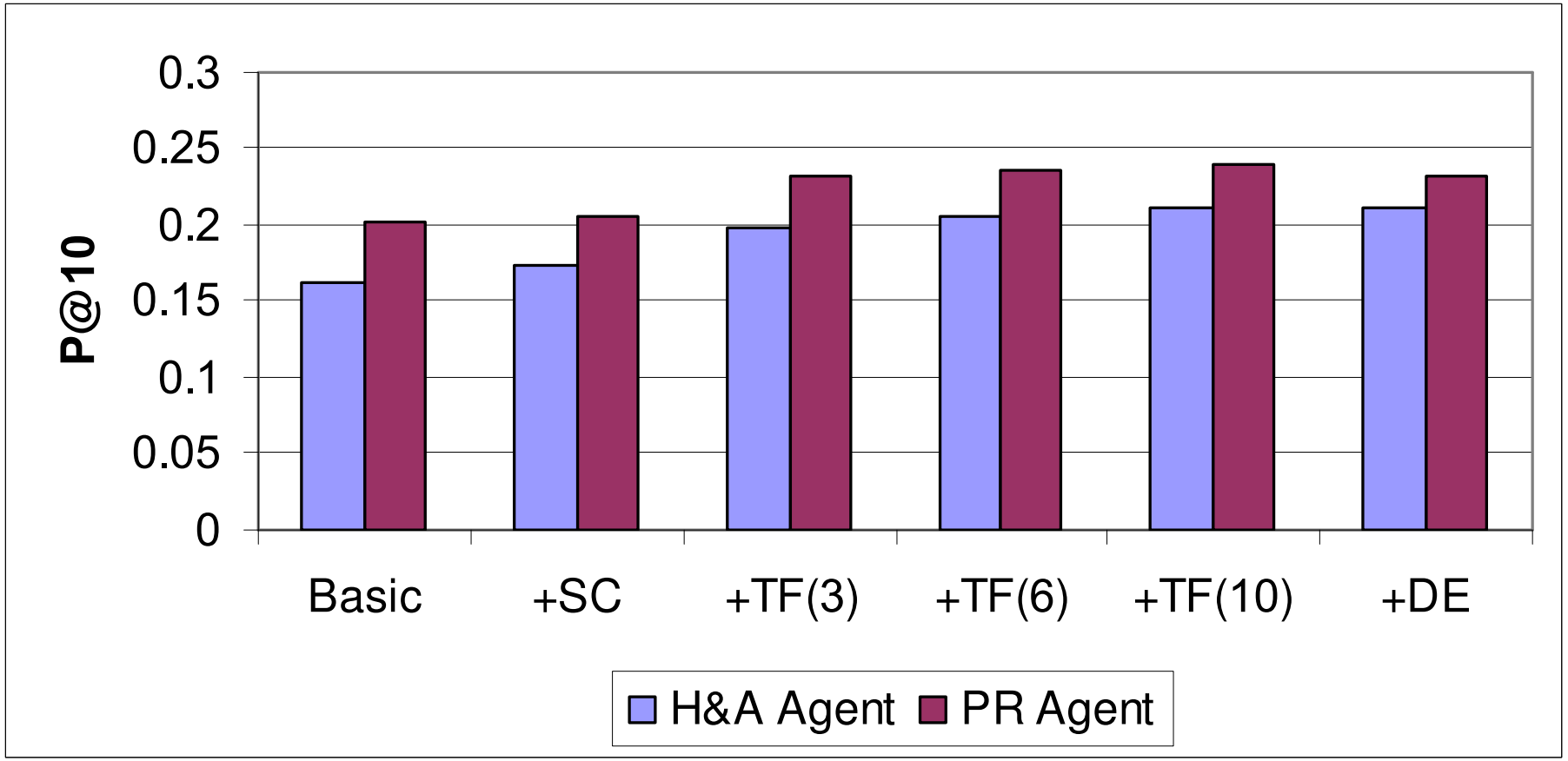
Distillation Filters

- Invoked on the result set -
  - Identify pages that exhibit features of a good topical page
  - Boost their relevance score
- Three distillation filters:
  - Duplicate Elimination Filter:
    - Filter out duplicate results from the top results
  - Site compression filter:
    - The score of each page is altered to reflect the scores of pages within *its logical site* that are accessible by *one or two links*
  - Title filter
    - Filter out pages from the top-10 results with a title **NOT similar** to the query (*frail* pages).

The difference between P@10 of our runs and median P@10 of all participants



Filter contribution



Summary

- This experiment describes how KA can be used for the Topic Distillation task.
- Sophisticated link-based measures:
  - Do not significantly improve search results in comparison to standard text-based relevance scoring
- Post Filtering of search results is a must:
  - Filtering results by additional measures:
    - duplicate-elimination filter
    - site-compression filter
    - title filter
- One ranking flavor does not fit all queries:
  - Informational vs. Navigational queries (Broder 02)
  - Parameters governing the ranking formula should be set based on the nature of the query

Juru at Trec 12 (2003): QUEry Sensitive Tuner

- Tunes the query parameters according to the query's characteristics
- Classifies queries into "informational" vs. "navigational" by considering:
  - Query's length (# of terms)
  - *edf* - expected document frequency – number of documents containing all query terms
- The main idea:
  - Long queries / small *edf ("informational")*:
    - standard IR techniques are sufficient
  - Short queries / large *edf ("navigational")*:
    - Ranking based on textual scoring is not enough
    - Documents' static scores and anchor data should be used in order to distill the best results

QUEry Sensitive Tuner (details)

- Treats separately queries containing one, two or three+ terms.
- For any query length, it maintains :
  - A threshold on the query *edf*
  - Two sets of values for the ranking parameters
    - An informational set and a navigational set
- A query with an *edf* lower than threshold
  - classified as ‘informational’
  - parameters are set using the informational parameters set
- A query with an *edf* higher than the threshold
  - classified as ‘navigational’
  - parameters are set using the navigational parameters set

Query Parameters tuned by QUEST

- Boosts for different token types:
  - Textual tokens (title, strong, mid, regular)
  - Anchor tokens (Different site, Same site ,Same Dir)
  - URL tokens
  - Snippet Tokens
  - Textual tokens receive high boosts for informational queries while anchor tokens, URL and snippet tokens receive high boosts for navigational queries
- Lexical Affinity weight
  - Determines the relative significance of a lexical-affinity to the document's score compared to a simple keyword
  - Higher value for informational queries
- Static Score coefficient
  - Determines the relative weight between the textual score and the static score of the page
  - Higher value for navigational query
- Cohesiveness coefficient
  - Determines the relative weight between the page's score and its cohesiveness score
  - Higher value for navigational query

Approximating *edf* per query

- For one-term query the *edf* can be precisely determined
  - Since the document frequency (*df*) of each term is stored within the index
- For multi-term query
  - Assuming independent query terms:

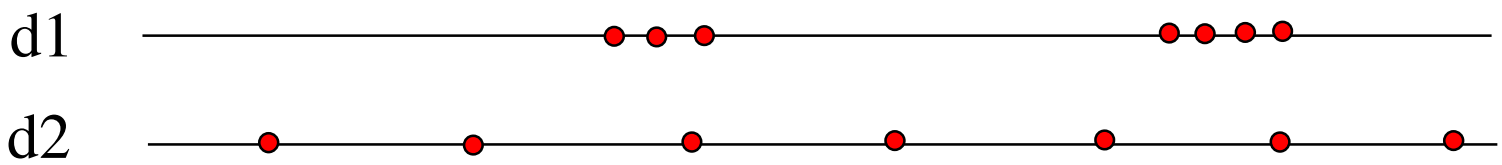
$$edf(q) = \frac{\prod_{i=1}^{k} df(q_i)}{|D|^k} |D|$$

Since query terms are usually dependent:

$$edf(q) = \frac{\theta(k) \prod_{i=1}^{k} df(q_i)}{|D|^{k-1}}$$

The Cohesiveness Filter

- identify pages that focus on the desired topic
  - In contrast to pages that
    - just mention it in passing,
    - just mention it in the context of a broader topic.
  - Achieved by identifying pages in which the query terms are uniformly distributed over the entire page.
  - Especially important when dealing with results that contain query terms of high frequency



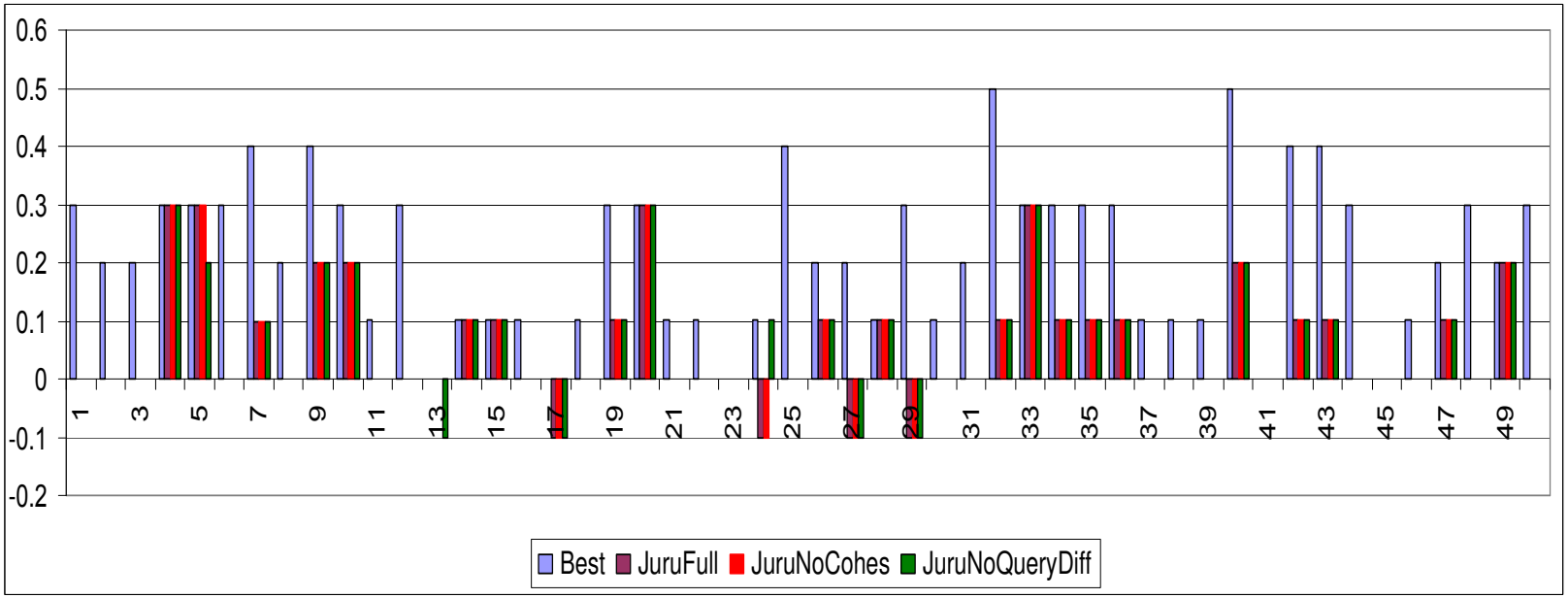
cohes(d2) >> cohes(d1)

Results

- We used the Juru search engine to index and search the ".gov" domain.
- Each page was indexed based on:
  - content
  - anchor descriptions
  - URL
  - snippet
- Each page is ranked by a linear combination of its textual score, and its link topology score (a static score)
  - The Static Score weight is determined by QUEST according to the query type
  - The static score is based on the in-links to the page - *n*.

$$St(p) = \begin{cases} 1.0 & n \geq N \\ \sqrt{n/N} & otherwise \end{cases}$$

Comparison to the Best results and Median result



Summary

- Our experiments this year focused on improving the ranking algorithm of our system.
- We experimented with
  - The QUEST algorithm that tunes the query parameters
  - The cohesiveness filter that tries to find topical pages
    - By identifying those in which the query terms are uniformly distributed over the entire page.
- Our results demonstrate that link analysis and anchor-text data slightly improved the results, in contrast to last year's results.

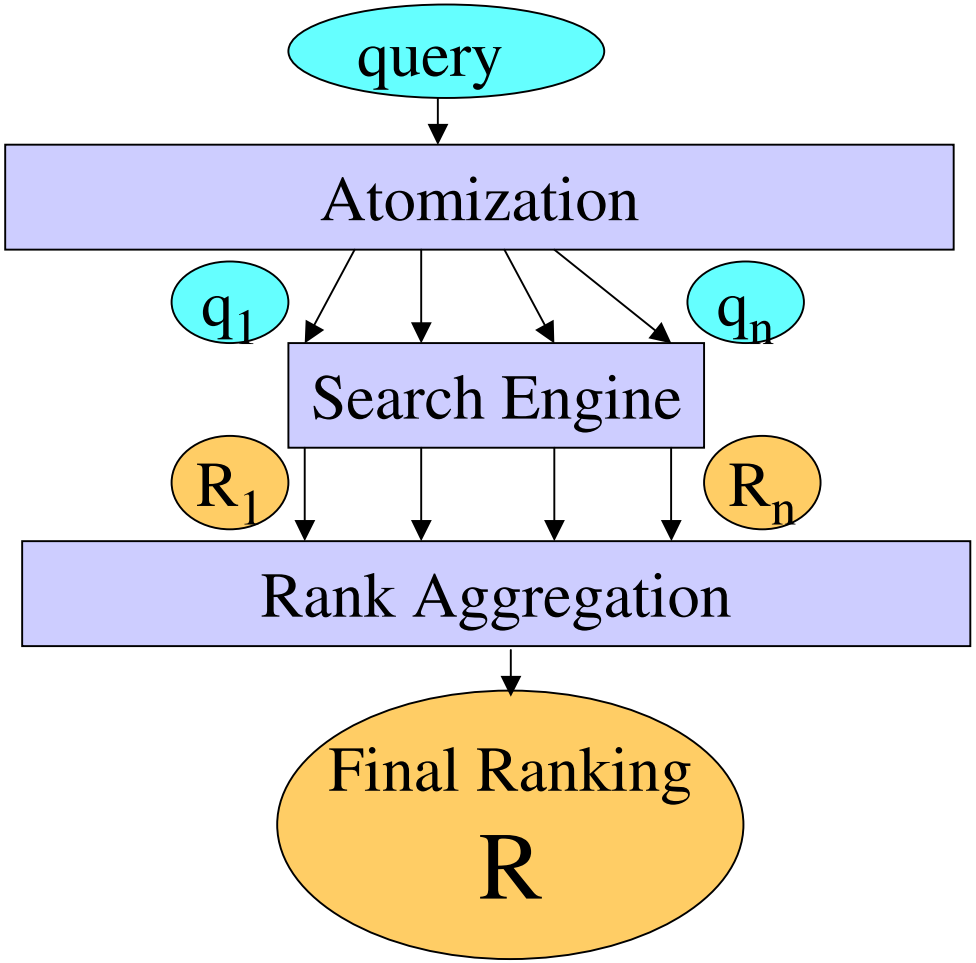
Juru at TREC-13 – Robust Track

- Goal – improve the consistency of the SE by focusing on poorly performing queries.
- New technology: Query Atomization and aggregation:
  - Based on ideas from Machine Learning
  - New complementary ranking mechanism
    - Query is broken into sub-queries (words, lexical affinities, others)
    - Documents are ranked based on rank aggregation of the sub-queries
- Prediction of Query Difficulty
    - Based on the “agreement” between sub-queries
    - Provide feedback:
      - To the user – how good the results are
      - To the SE - should the query be refined?

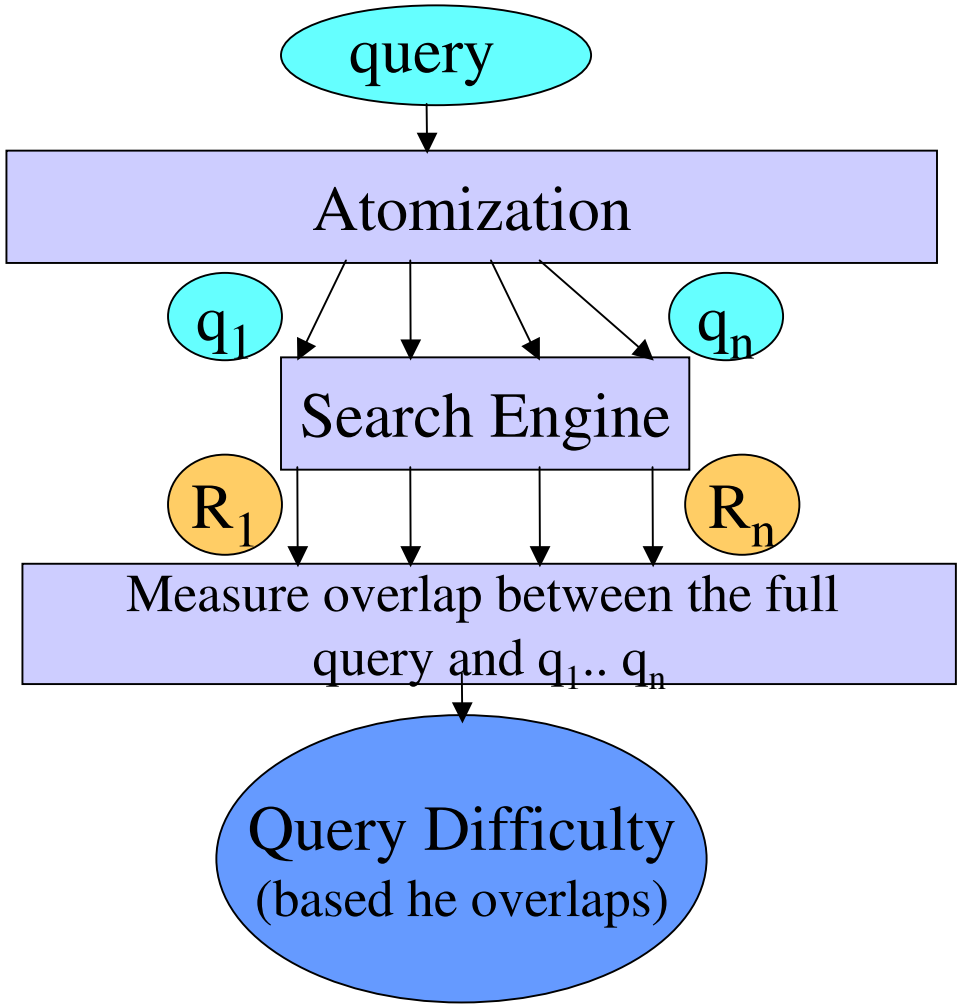
Advantages:

- Feedback to the user:
  - The user can rephrase the query
- Feedback to the search engine:
  - Automatic query Expansion
  - Use of different ranking functions for different queries based on Merit Prediction
- A method for deciding which search engine to use.

Rank Aggregation



Merit Prediction



Query Prediction by Histogram

IBM Labs in Haifa

	ID of top 10 documents retrieved:										Number of overlaps
Full query: "**Magnetic Levitation Maglev**"	77002	39741	76311	35273	87941	33402	47457	1013	22953	17382	
Sub-query 1: Magnetic	39741	6794	50129	1013	43506	69131	9273	47457	77266	17948	3
Sub-query 2: Levitation	47457	47947	39741	89657	35273	77402	69775	77002	87941	28369	4
Sub-query 3: Maglev	77003	35274	75402	45525	17881	70077	1010	24524	71172	76499	0
Sub-query 4: levitation magnetic	39741	77002	47457	30123	87941	16481	60688	89657	20896	22162	4
Sub-query 5: levitation maglev	67036	9391	39741	94709	76311	47457	35273	1013	61496	77002	6
Sub-query 6: magnetic maglev	12944	1013	26549	89657	39741	35273	77266	6123	33402	84398	3

Overlap histogram

Number of overlapping queries	1	0	0	2	2	0	1	0	0	0	0
Number of overlaps	0	1	2	3	4	5	6	7	8	9	10

Binary histogram

Number of overlapping queries	1	0	0	1	1	0	1	0	0	0	0
Number of overlaps	0	1	2	3	4	5	6	7	8	9	10

Histogram rank

1	2	5	7	8	9	10	0	3	4	6

Query Prediction = Histogram-Rank * (learned)weight-Vector

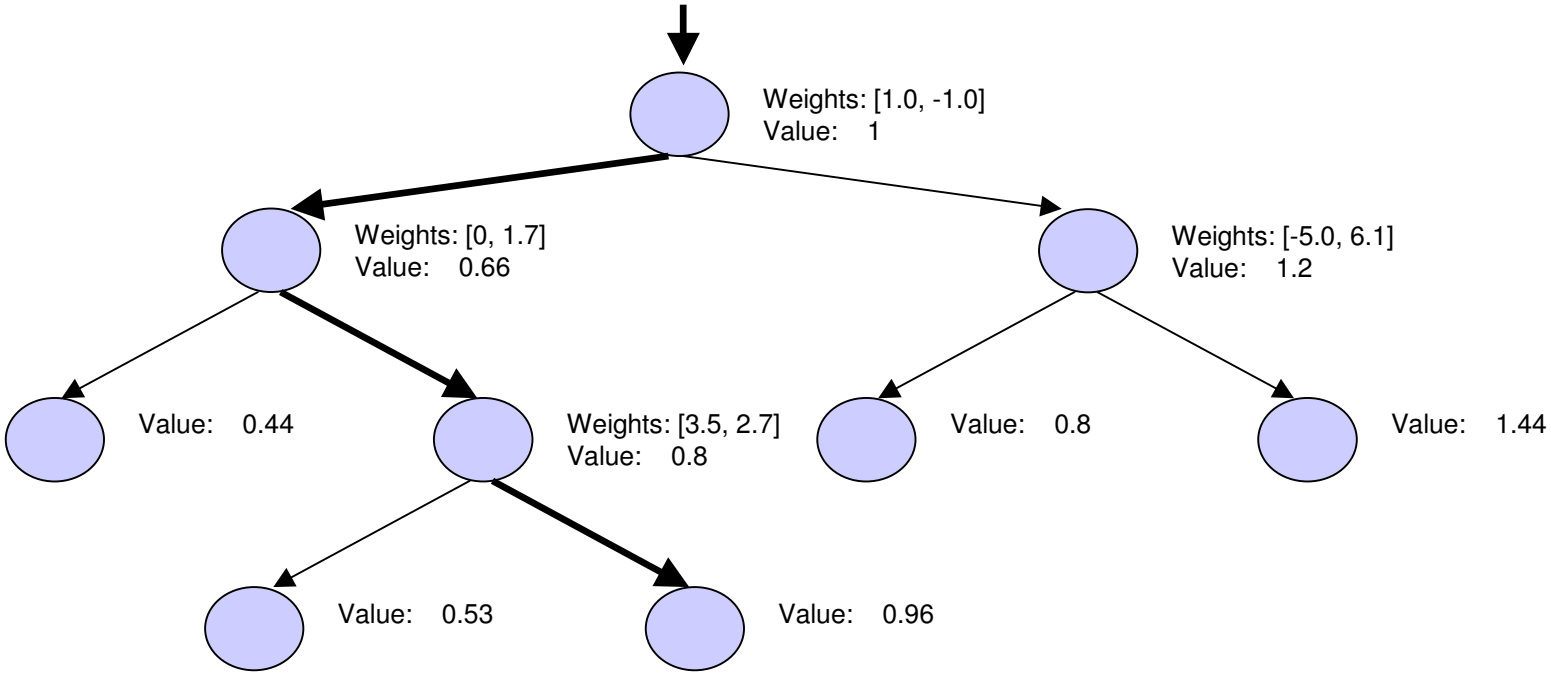
How do we learn the Weight Vector?

- Given a set of queries, each associated with relevant results
- Compute the Histogram Rank and the actual precision for each query (eg. P@10, MAP)
- Compute the weight vector so as to minimize the mean square error between the predicted and the actual precision.

Number of overlaps per sub-query:

Log(df) per sub-query:

3	4	0	4	6	3
8	5	3	8	5	8



Difficulty: 0.96

How do we learn the Decision Tree?

- Given a set of queries, each associated with relevant results
- Compute:
  - the number of overlaps and *df* for each sub-query
    - *df*: the document-frequency i.e., number of docs containing the sub-query.
  - the actual precision for each query (eg. P@10, MAP)
- Compute the decision tree so as to minimize the mean square error between the predicted and the actual precision.

Results

- We used the TREC data for evaluation (Trec collections/200 marked queries)
- Trec queries were ranked according to the actual and the predicted precision
- Similarity between the ranks is measured using kendall-tau
- High similarity reflects good prediction
- Very good histogram-based prediction for description queries (~7 terms per query)
- Good tree-based prediction for title queries (2-3 terms per query)

	Query type	Tree		Histogram	
		MAP	P10	MAP	P10
TREC	Title	0.284	0.271	0.161	0.144
	Description	0.222	0.231	0.357	0.282
	Title + Description	0.200	0.194	0.306	0.265
Web	Title	0.118	0.175	0.110	0.155
	Description	0.202	0.098	0.093	0.140
	Title + Description	0.192	0.182	0.142	0.283
TREC+Web	Title	0.323	0.273	0.216	0.183
	Description	0.273	0.293	0.421	0.362
	Title + Description	0.273	0.265	0.345	0.312