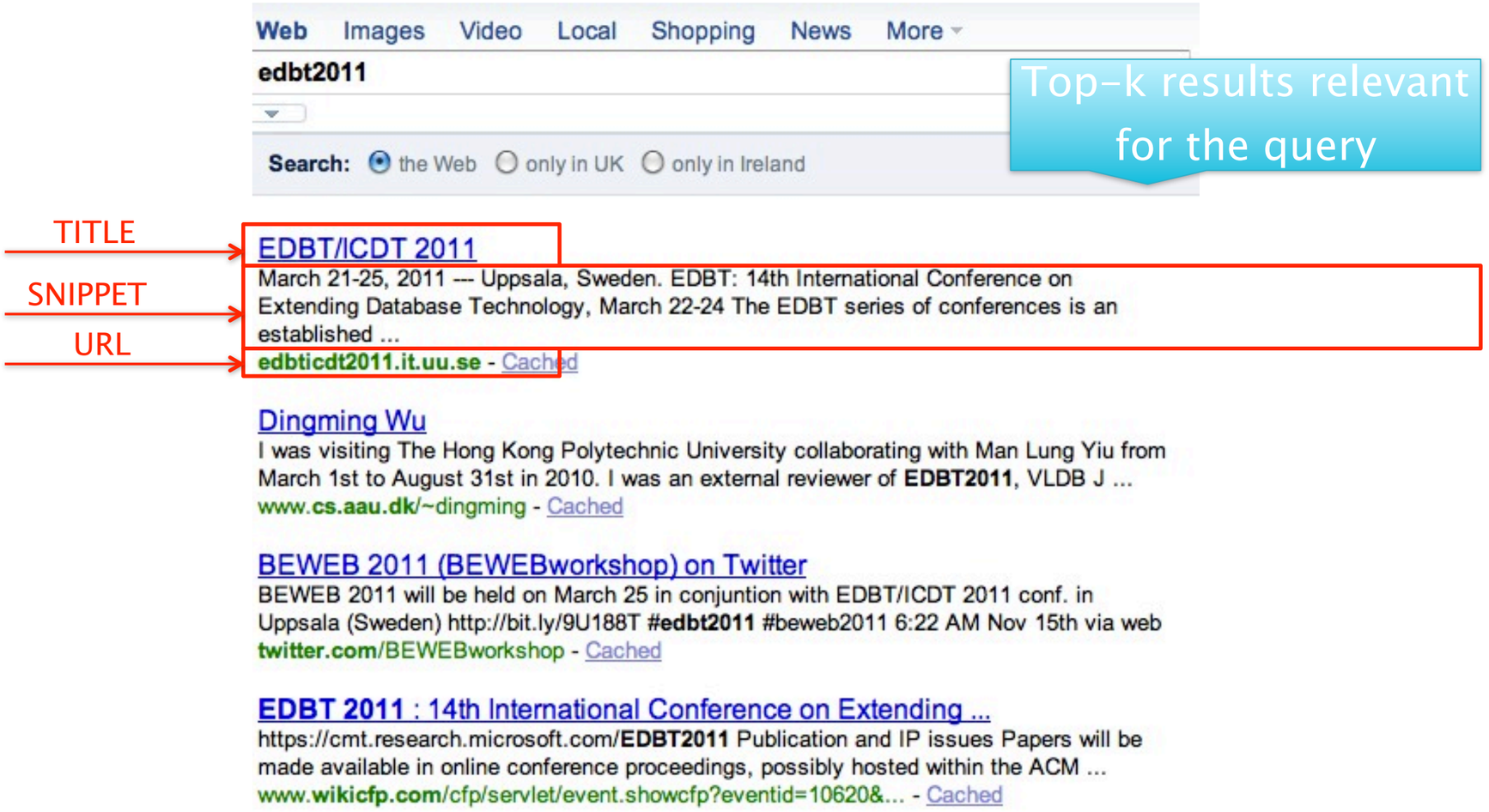
Caching query-biased snippets for efficient retrieval

Salvatore Orlando[+], Raffaele Perego[*], Fabrizio Silvestri[*]
[*]ISTI - CNR, Pisa, Italy
[+]Università Ca' Foscari Venezia, Italy

Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri: **Caching query-biased snippets for efficient retrieval. EDBT 2011: 93-104**

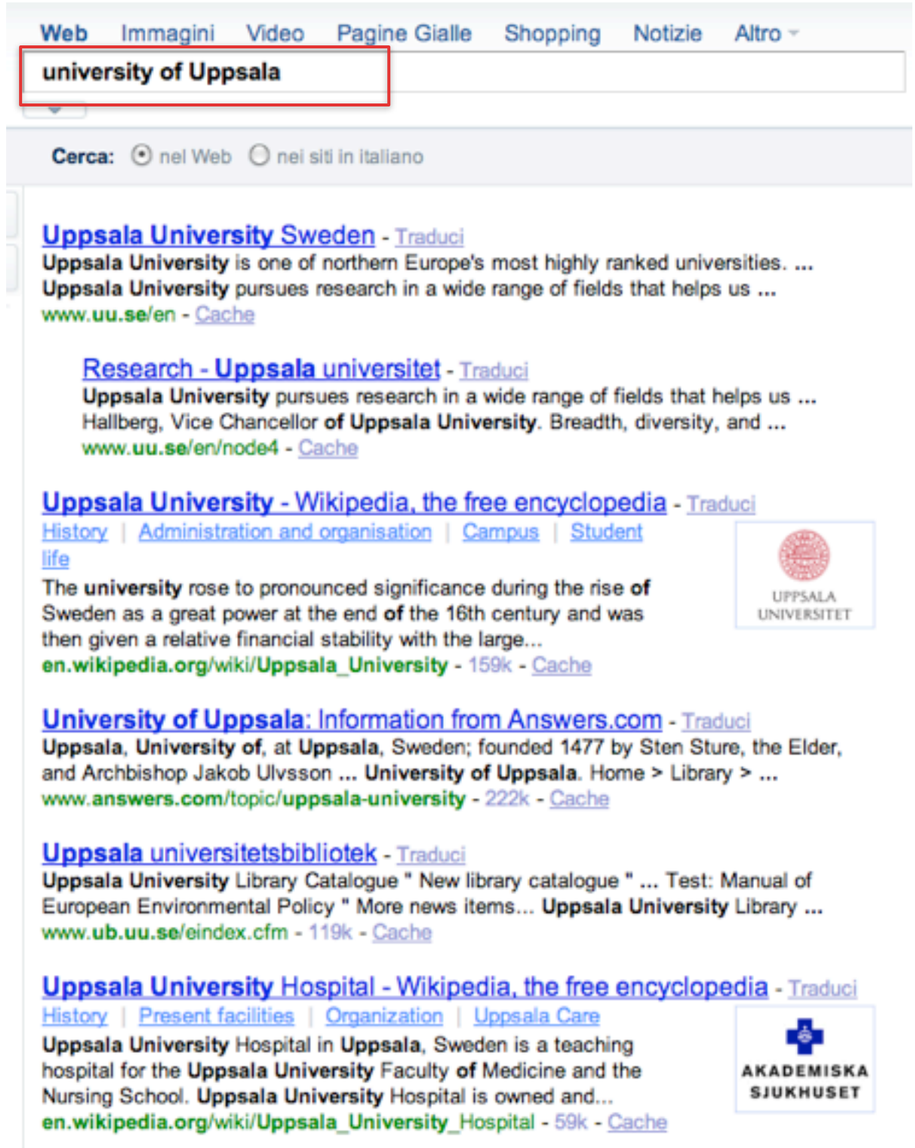
Search Engine Results Page

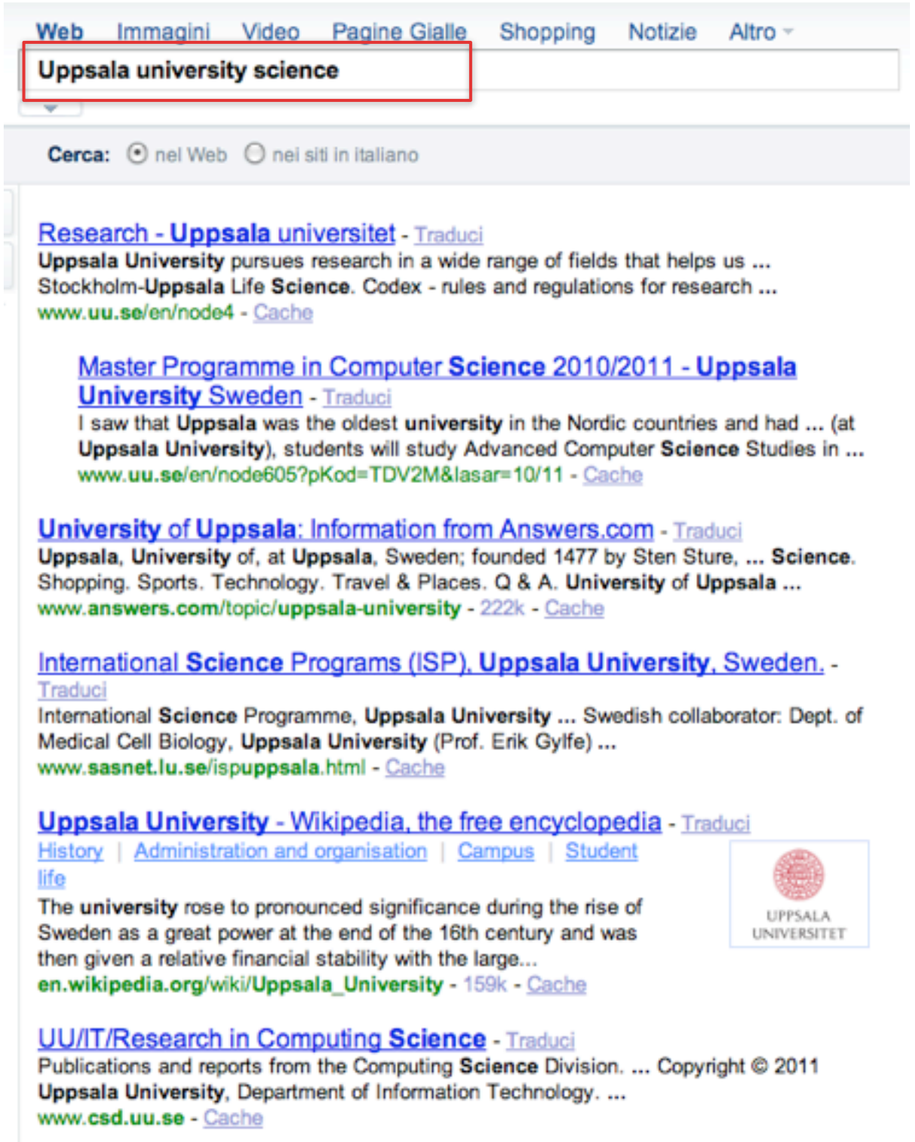


Search Engine Results Page

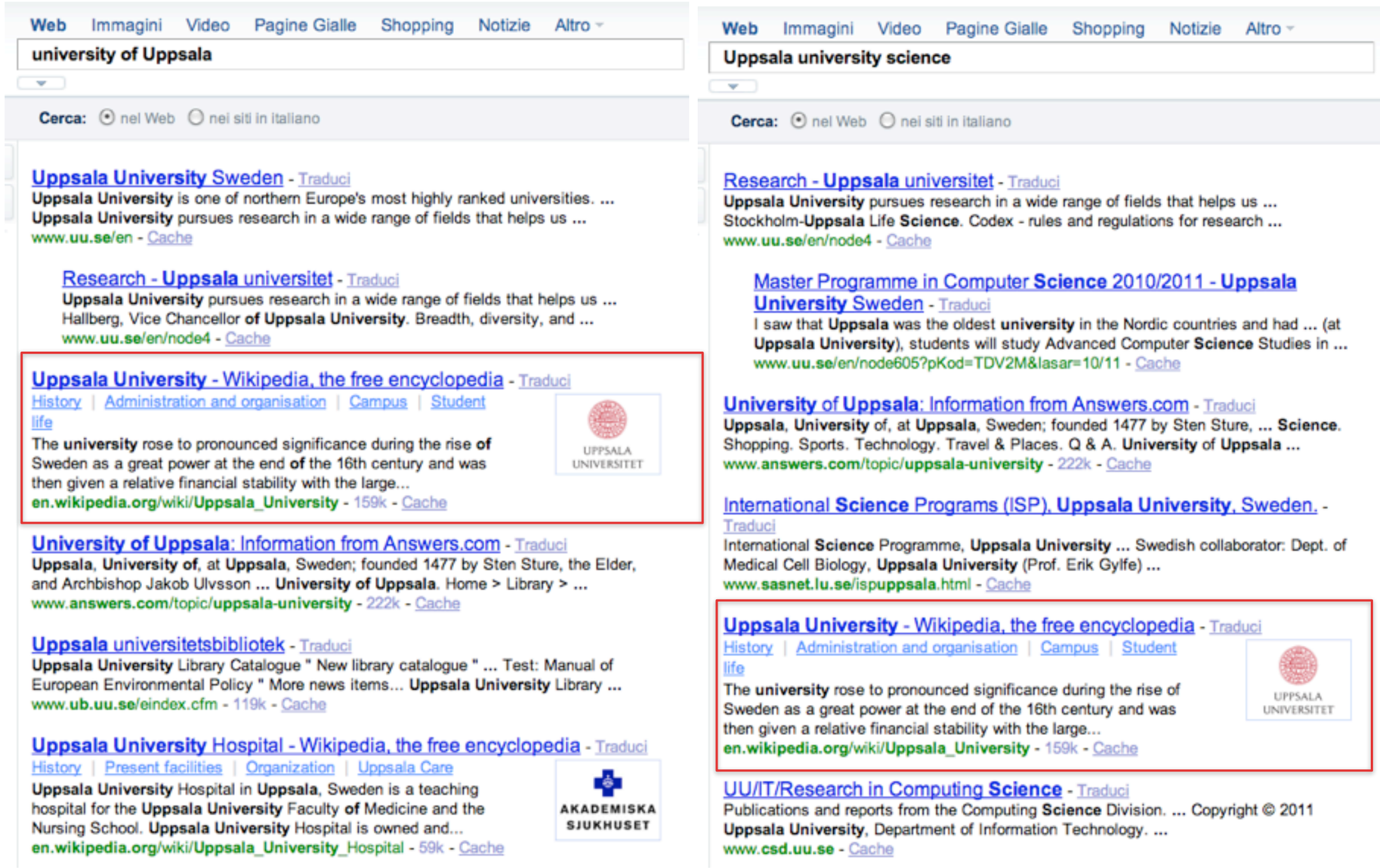
- Very important to estimate relevance of the result for the user's query
  - high-quality, query-biased
  - expensive to generate

Different queries....

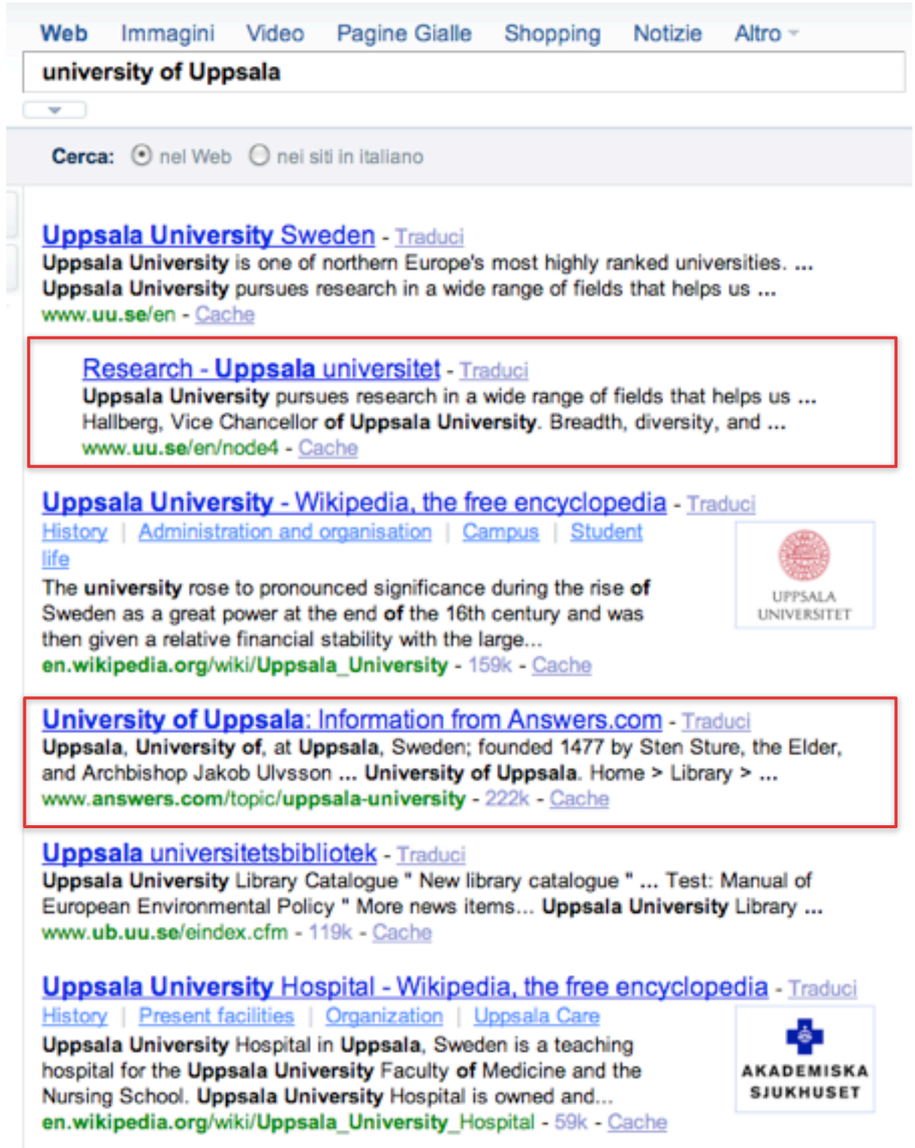


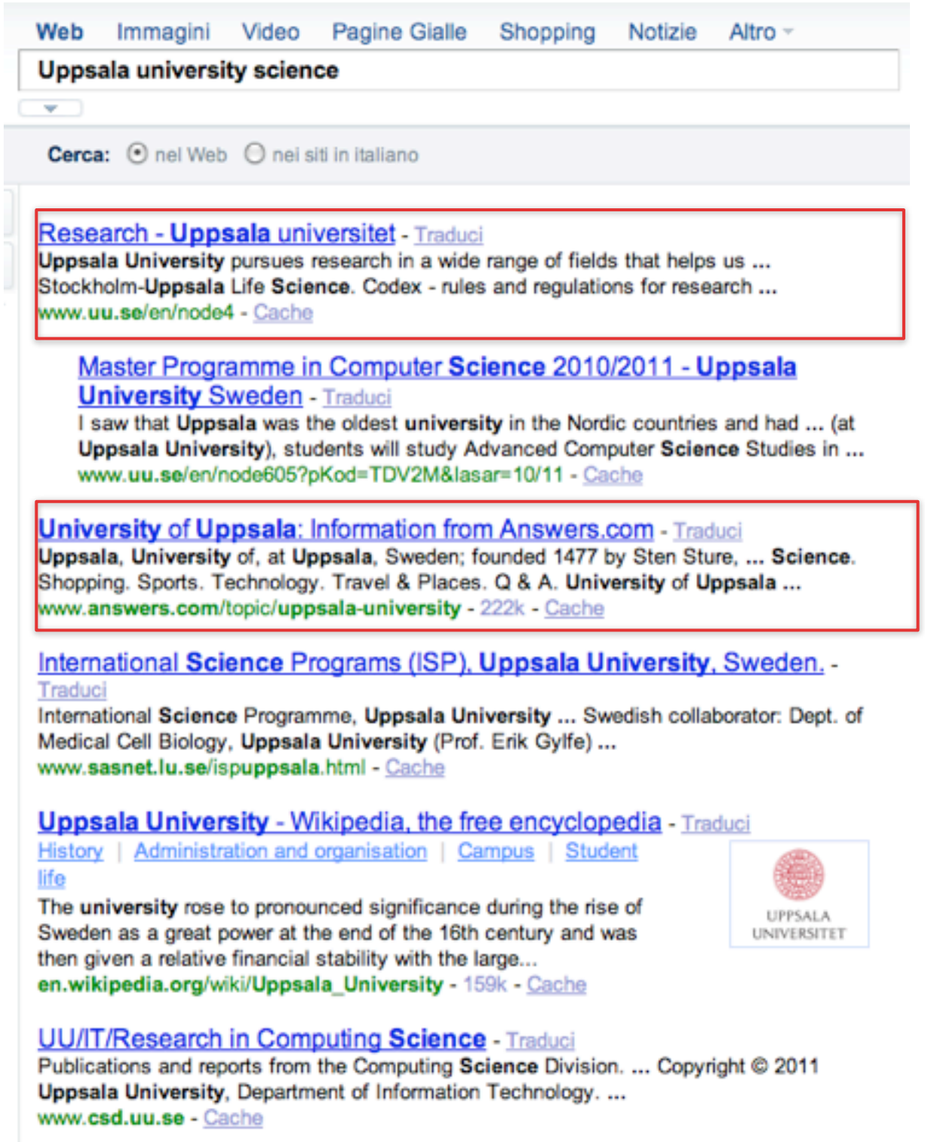


Same title, same URL, same snippet (A)

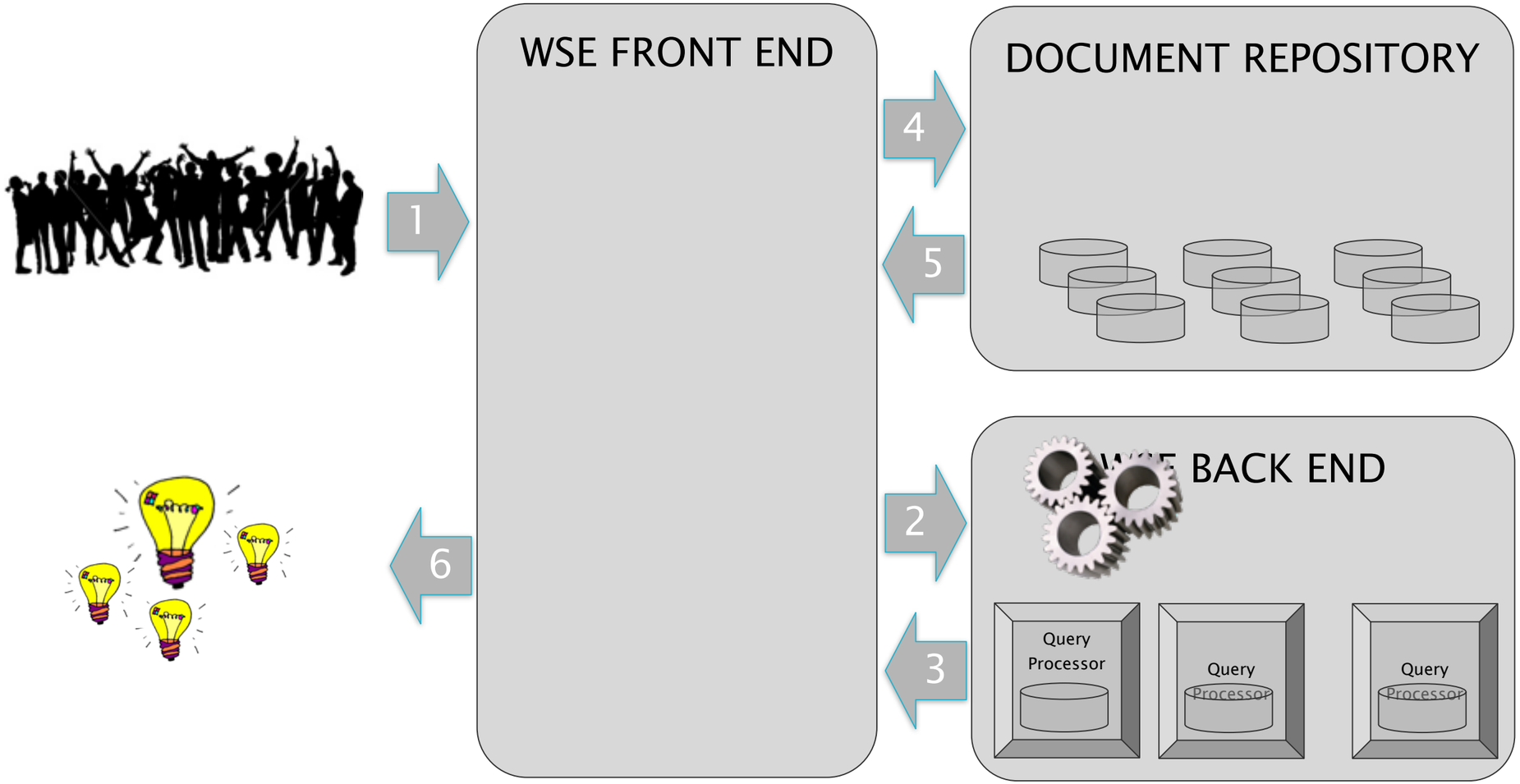


Same title, same URL, different snippets (B)

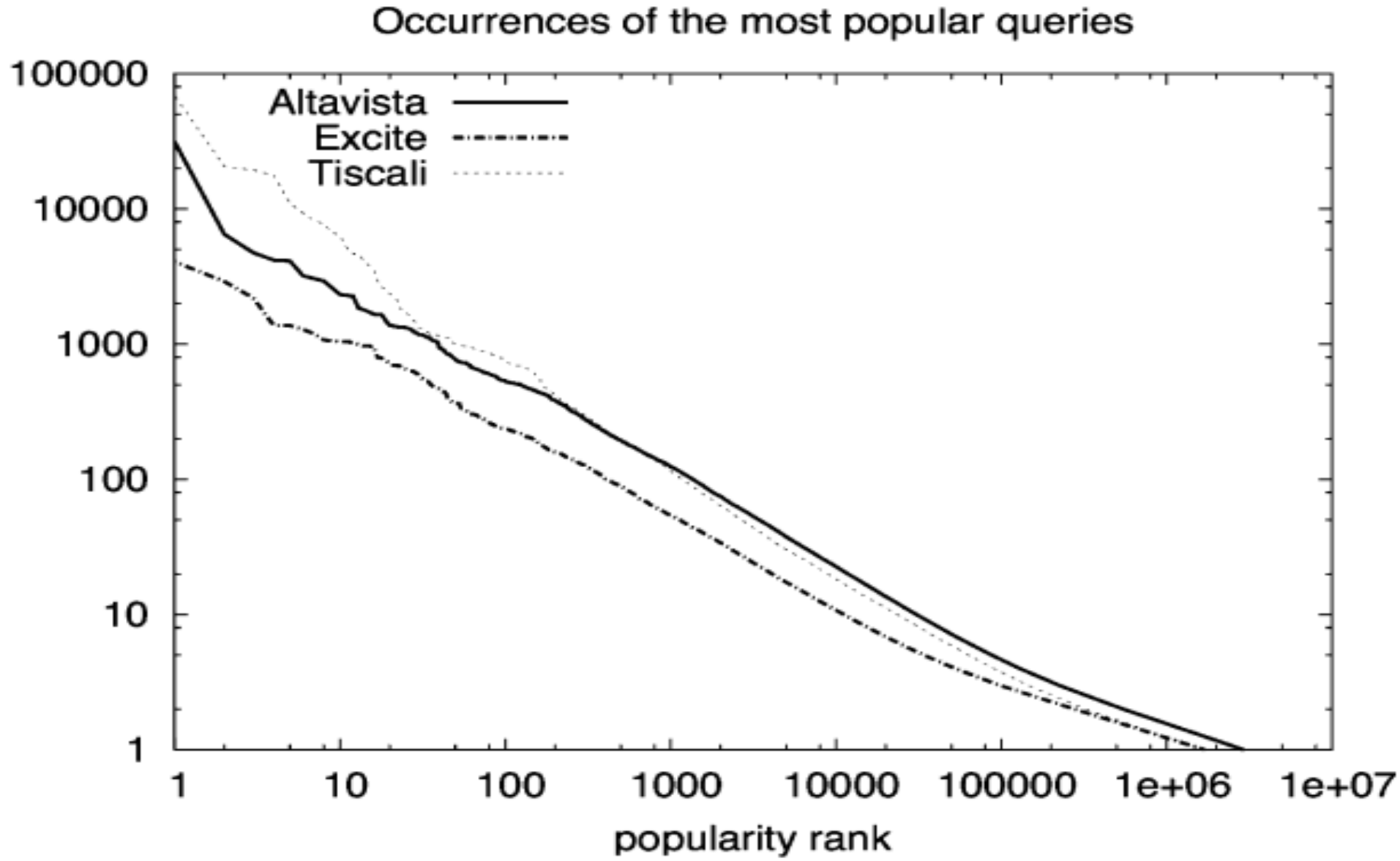




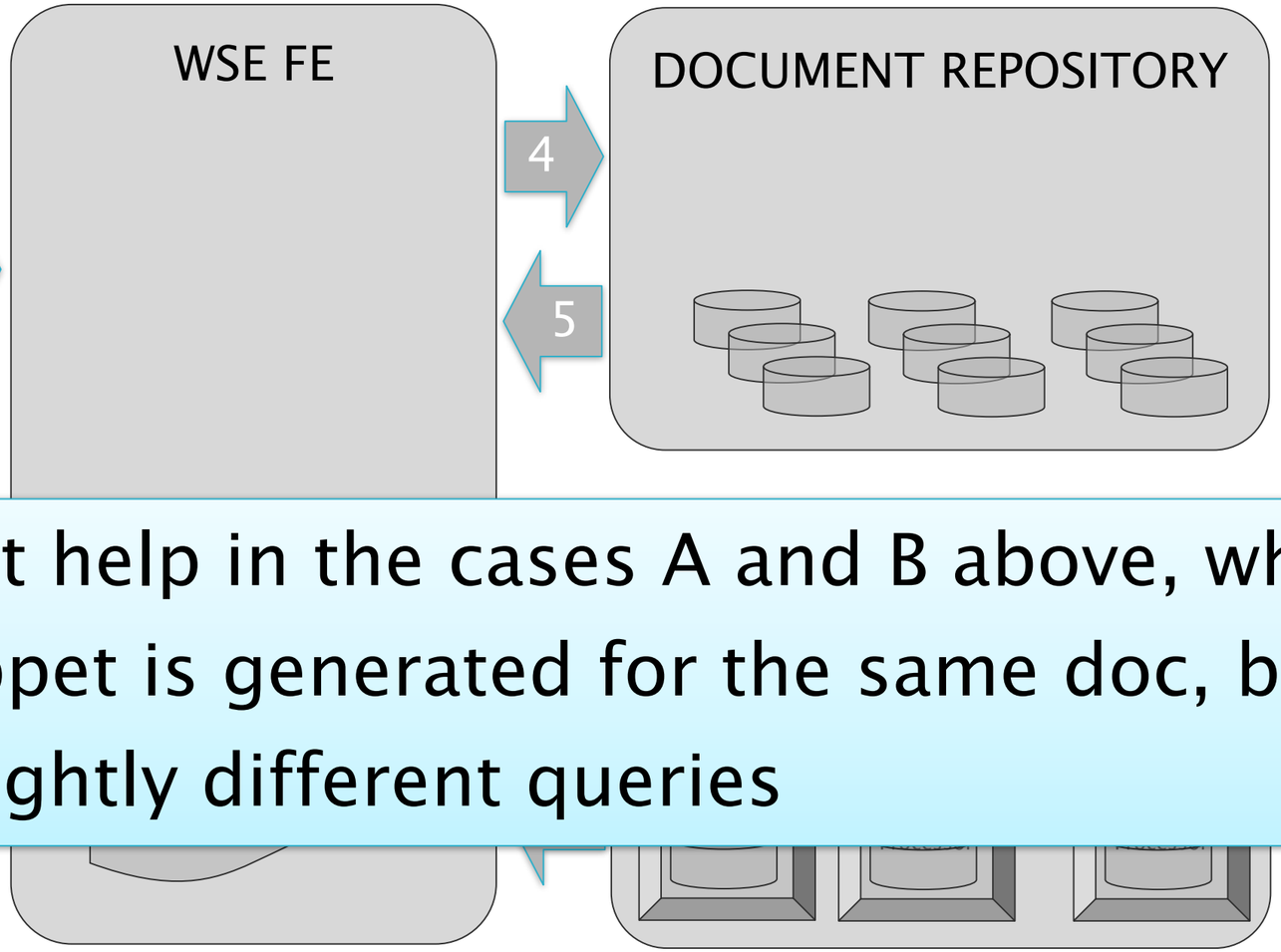
Web Searching process



WSE Front End caching

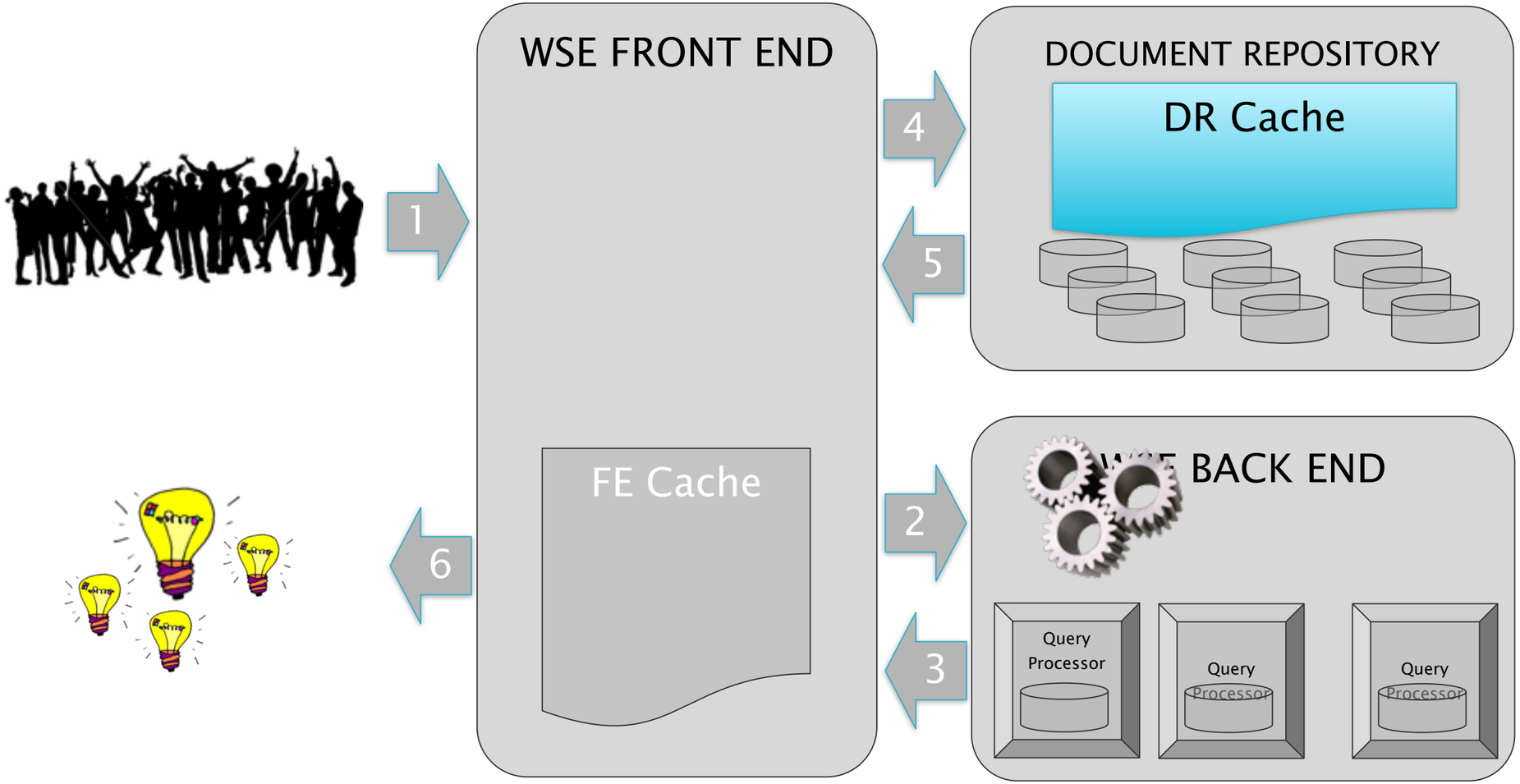


WSE Front End caching



SERPs' caching does not help in the cases A and B above, when the same or a similar snippet is generated for the same doc, but for slightly different queries

Doc Repository Caching strategies



Doc Repository Caching

Given q and docID, the DR has to: 1) retrieve document content and URL, 2) generate an effective query-biased snippet

- A DR cache can substantially reduce disk accesses
- Two cache organizations discussed in literature:
  - $DR_{Cachedoc}$ storing integral copies of most accessed documents*
  - $DR_{Cachesurr}$ storing surrogates of most accessed documents**

We propose $DR_{CacheSsnip}$ whose entries store supersnippets built on the basis of the past queries submitted to the WSE

*Andrew Turpin, Yohannes Tsegay, David Hawking, and Hugh E. Williams. Fast generation of result snippets in web search. SIGIR '07

**Y. Tsegay, S. J. Puglisi, A. Turpin, and J. Zobel, Document Compaction for Efficient Query Biased Snippet Generation. ECIR '09

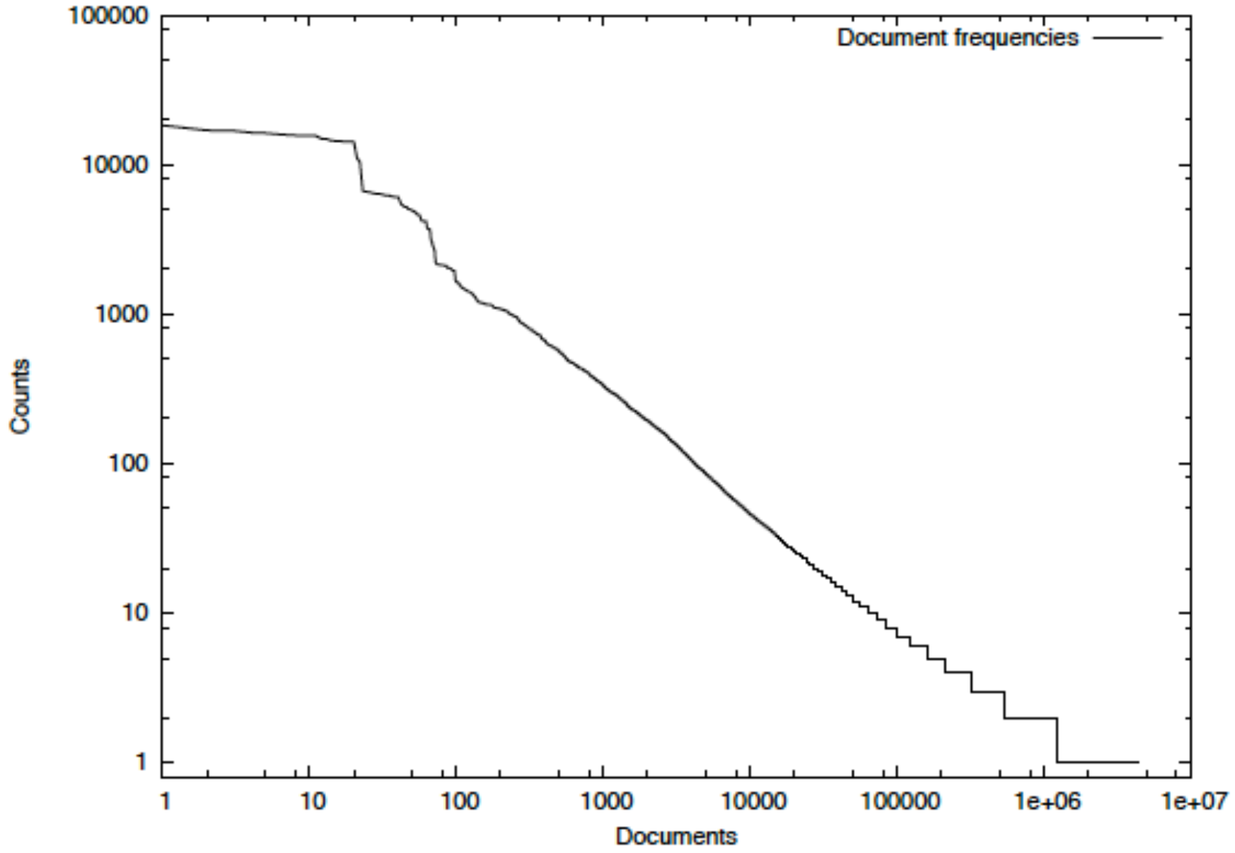
The Query Log used

- MSN RFP 2006 query log
  - queries from a US Microsoft search site sampled over one month (May 2006)
  - submitted to Yahoo! BOSS to retrieve top-10 results
  - 9,000,000 queries
  - 4,447,444 distinct queries
  - 25,897,247 distinct URLS -> high URLs sharing!

Query Log	Queries	Distinct Queries	Distinct Urls (top 10)
D1			
training	1,000,000	601,369	4,317,603
test	500,000	310,495	2,324,295
D2			
training	9,000,000	4,447,444	25,897,247
test	4,500,000	2,373,227	12,134,453

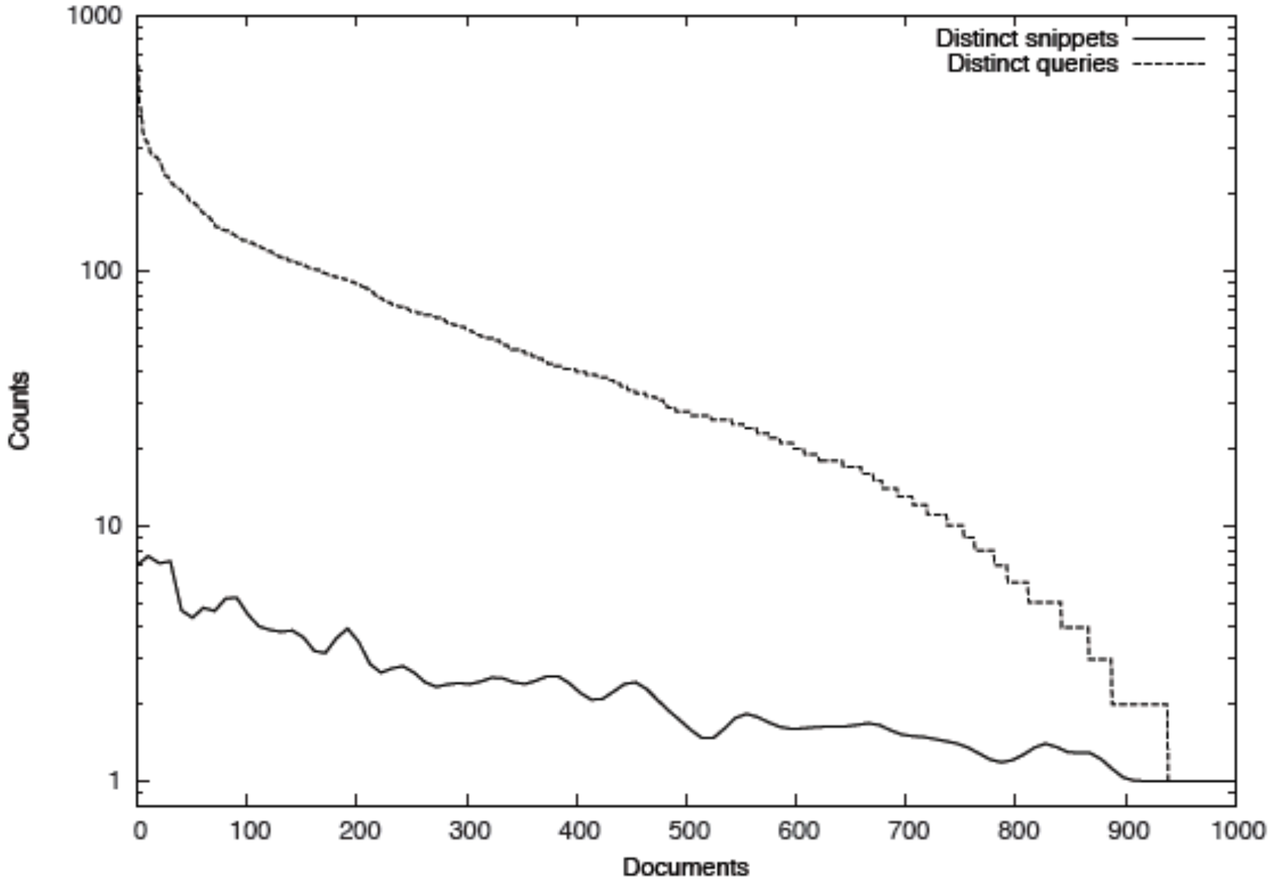
Fact 1: URLs occurrences in SERPs

- Double Pareto distribution



Fact 2: top-1000 most frequently retrieved documents

- snippets generated for the same document are a few (<9)
- a document may answer several different queries



Evidences

1. Relevant documents are characterized by a few snippets
2. Relevant documents are shared by different queries
3. Snippets are made of a few sentences
4. Different snippets from the same document share common sentences

Q: What do you get when you cross an Elephant with a Rhino?

A: I haven't a goddamn clue.

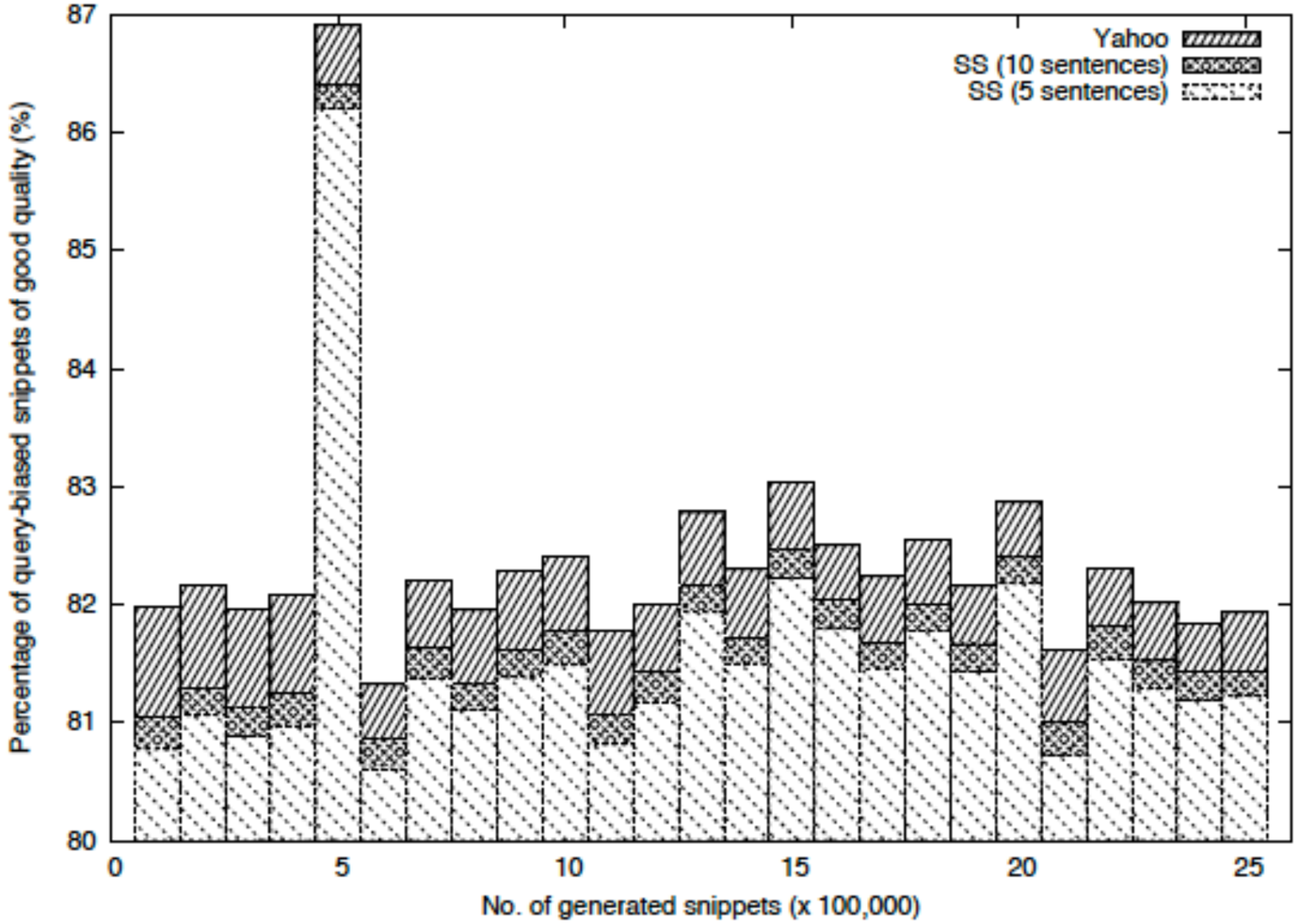
Definition of QL-based supersnippet

Given the set Q_d of all the past queries in QL for which document d was returned, we define supersnippet ss_n of d the set of the n most frequent sentences occurring in the snippets $S_{d,q}$ generated for answering the queries in Q_d

Claim

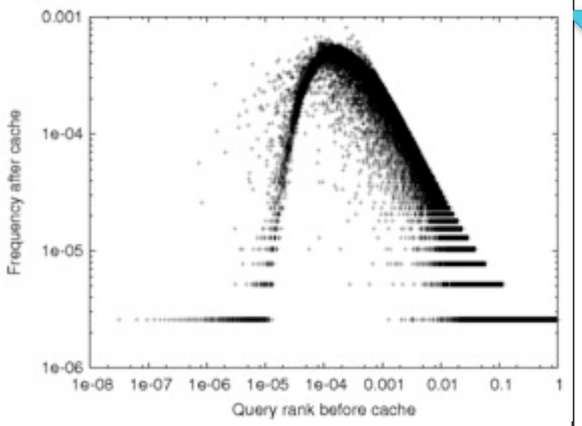
Caching LRU supersnippets ss_n is an effective and efficient strategy for DR_{Cache}

Effectiveness: $DR_{CacheSsnip}$ vs. Yahoo! BOSS



Efficiency

SERPs cache filtering out most recently submitted queries



256K (0.38)

FECache #Entries (Hit Ratio)	DRCache	Size (in MB)	Hit Ratio DRCache
	doc	256M	0.083
		512M	0.1
		1024M	0.134
		2048M	0.17
	surr	256M	0.104
		512M	0.135
		1024M	0.177
		2048M	0.23
	ss_5	256M	0.139
		512M	0.194
		1024M	0.266
		2048M	0.34
	ss_{10}	256M	0.13
		512M	0.189
		1024M	0.266
		2048M	0.334

SS Cache peculiarities

```
CacheLookup(q, DocID)
    if InCache(DocID) then
        SS <- RetrieveSS(DocID)
        snippet <- GenerateS(q,SS)
        if Quality(q,SS) < Threshold then
                        snippet <- UpdateSS (SS, DocID, q)
        UpdateLRU(DocID)
    else
        SS <- null
        snippet <- UpdateSS (SS, DocID, q)
        ReplaceLRU(SS, DocID)
```

SS Cache peculiarities

```
CacheLookup(q, DocID)
    if InCache(DocID) then
        SS <- RetrieveSS(DocID)
        snippet <- GenerateS(q,SS)
        if Quality(q,SS) < Threshold then
                    snippet <- UpdateSS (SS, DocID, q)
        UpdateLRU(DocID)
    else
        SS <- null
        snippet <- UpdateSS (SS, DocID, q)
        ReplaceLRU(SS, DocID)
```

A “Quality” Miss may occur when the quality of the snippet generated from the available SS is below a given threshold

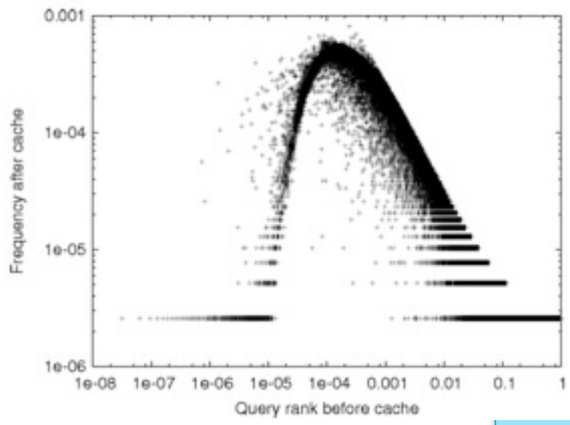
Quality Miss count

- We count a Quality Miss when even a single query term is not present in the snippet
  - Also Yahoo! BOSS snippets sometimes do not contain ALL query terms
    - misspellings, disjunctive queries, etc.
- We underestimate the Hit Ratio

What about considering a Quality Miss only when our snippets' quality is lower than Yahoo!'s one ?

Refined Efficiency Measurement

SERPs cache filtering out most recently submitted queries



128K (0.36)

DRCache	Size (in MB)	Hit Ratio DRCache
ss_5	256M	0.21
	512M	0.29
	1024M	0.36
	2048M	0.42
ss_{10}	256M	0.2
	512M	0.28
	1024M	0.356
	2048M	0.419
	256M	0.19

Total Hit Ratio: ~62%

Conclusion & Future Work

- Novel technique for scaling up search engine performance by snippet caching
  - motivated by the analysis of a large real-world query log
  - enabling the generation of effective snippets even for queries and URLs not previously seen
  - achieving high Hit Ratios (cumulatively up to 0.62!)
  - evaluated by means of large-scale experiments

- Open issues
  - study of dynamic vs static policies for managing supersnippets
    - quality of SSs suffers for aging?
  - study of diversification in supersnippets?
  - heuristics, machine learning approaches for managing Quality Misses

Questions?