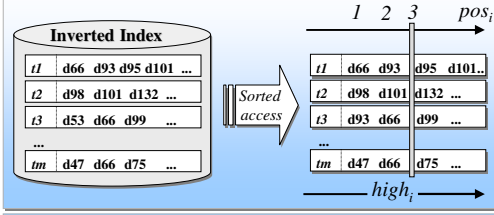
Probabilistic Top-k Query Processing

Scope: Efficient Support for Top-k Queries

- **Given:**
  - Query $q = t_1\, t_2 \ldots t_m$ with m (conjunctive) keywords
  - Document collection $D = \{d_1, \ldots, d_n\}$ of m-dim. vectors d_j
  - Term weights s_{ij}, e.g. $s_{ij} = tf_{ij} \log(idf_i)$
  - Similarity scoring function $score(q,d_j)$, e.g. $score(q,d_j) = \sum_{i=1}^{m} s_i(q_i, d_j)$
- **Find:**
  - Top k results documents with regard to $score(q,d_j)$

Inverted index contains for each t_i an **index list** L_i
- With all docs d_j that contain t_i sorted by s_{ij}
- Index lists are sequentially scanned only (no random access)
- Support for structured data

Naive approach: First join all lists by document id, then sort by scores → $O(mn^2)$



Candidate queue @ scan depth 3:
d66 → d93 → d98 → d47 → d101 with E(d66) = {1,2,3}, E(d93) = {1,3}, …

Fagin's Threshold Algorithm [1] as Baseline

TA-sorted(Query q, IndexLists L):
```
top-k :={ dummy_1, ..., dummy_k } with worstscore(dummy_v ) = bestscore(dummy_v )= 0;
min-k := 0;
candidates := ∅;
scan all lists L_i ∈ L for i = 1,..,m in parallel:
// e.g., round-robin or merged in descending order of s_i values
   consider item d at position pos_i in L_i;
   if d ∉ candidates then
      candidates := candidates ∪ {d};
      E(d) := {i};
   high_i := s_i(q_i,d); // current score in L_i
   E(d) := E(d) ∪ {i};
   bestscore(d):=aggr{ aggr{s_v(q_v,d) | v∈E(d) }, aggr{ high_v | v∉E(d) } };
   worstscore(d) := aggr{ s_v(q_v,d) | v∈E(d) };
   if worstscore(d) > min-k then
      if d ∉ top-k then
         remove argmin_d'{ worstscore(d') | d'∈top-k } from top-k;
         candidates := candidates ∪ {d'};
         add d to top-k
      min-k := min{ worstscore(d') | d' ∈ top-k };
   if bestscore(d) ≤ min-k then candidates := candidates − {d};
   threshold := max { bestscore(d') | d'∈ candidates };
   if threshold ≤ min-k then exit;
return top-k;
```

Score Distribution Estimators & Convolutions

- **TA family of algorithms based on invariant:**

$$\sum_{i\in E(d_j)} s_i(d_j) \le score(q,d_j) \le \sum_{i\in E(d_j)} s_i(d_j) + \sum_{i\notin E(d_j)} high_i$$

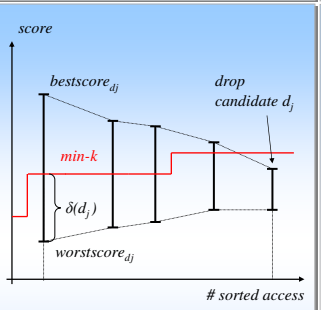
- If worstscore$_{dj}$ > min-k → add d_j to top-k
- If bestscore$_{dj}$ ≤ min-k → drop d_j from queue

- **Probabilistic threshold test:**

If $p(d_j) = P\left[\sum\{s_i \mid i \notin E(d_j)\} > \delta(d_j)\right] < \varepsilon$

with $\delta(d_j) := \min_k - \sum\{s_i \mid i \in E(d_j)\}$

→ Drop d_j from queue



- **Possible estimators for score distributions:**
  - ***Uniform*** in $[0, high_i]$ or in $[low_i, high_i]$
  - ***Truncated Poisson*** over n_i equidistant values with $P[d = v_j] = e^{-\alpha}\frac{\alpha^{j-1}}{(j-1)!}$
  - ***Arbitrarily distributed*** and approximated by an ***equi-width histogram*** with cells j=1,..,c with range $(l_j, h_j]$

→ **Consider convolution of score distributions over all t_i for $i \notin E(d_j)$**

Queue Management with Prob. Pruning

Prob-sorted(Query q, IndexLists L, RebuildPeriod r, QueueBound b):
```
top-k :={dummy_1, ..., dummy_k}; // with s (dummy_v )=0
min-k := 0;
candidates := ∅;
scan all lists L_i ∈ L for i = 1,..,m in parallel:
...same code as TA-sorted...
   // additional queue management and probabilistic pruning
   for all priority queues q for which d is relevant do
      insert d into q with priority bestscore(d);
   // periodic clean-up
   if step-number mod r = 0 then
      // dropping of queues; 2^m-1 unbounded queues                    Prob-cons
      if strategy = "Conservative" then
         for all priority queues q do
            if prob[top(q) can qualify for top-k] < ε then
               drop all elements of q;
      // garbage collection; single unbounded queue                    Prob-prog
      if strategy = "Progressive" then
         for all queue elements e in q do
            best(e) := bestscore of e with current high_i values;
            if best(e) < min-k then drop e from q;
            if prob[e can qualify for top-k]<ε then drop e from q;
      // rebuild; single bounded queue                                 Prob-smart
      if strategy = "Smart" then
         for all queue elements e in q do
            update bestscore(e) with current high_i values;
         rebuild bounded queue with best b elements;
         if prob[top(q) can qualify for top-k] < ε then exit;
      // no queue; greedy threshold approximation only                 Prob-aggr
      if strategy = "Aggressive" then
         if prob[virtual-element qualifies for top-k] < ε then exit;
   if all queues are empty then exit;
return top-k
```

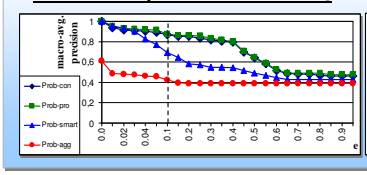
Evaluation on the TREC Benchmark

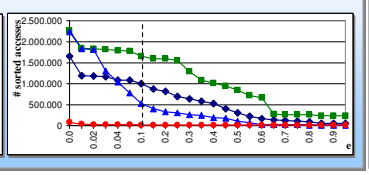
- **Gov data collection**
- n = 1.200.000 text documents
- m = 530.000 different terms
- 250.000.000 (doc, term) tuples incl. stemming & stop word removal

- **Performance at ε = 0.1**
- 87% average precision
- Sorted access savings of 57%
- Time savings of 83% due to decreased queuing overhead
- Scalability and wide range of application-specific tuning

	# sorted access	execution time	max. queue size	macro avg. prec.	rank distance	score distance
TA-sorted	2,263,652	148.7	10,849	1.0	0.0	0.0
Prob-con	**993,414**	**25.6**	**29,207**	**0.87**	**16.9**	**0.007**
Prob-pro	1,659,706	44.2	6,551	0.87	16.8	0.006
Prob-smart	**527,980**	**15.9**	**400**	**0.69**	**39.5**	**0.031**
Prob-agg	20,435	0.6	0	0.42	75.1	0.089

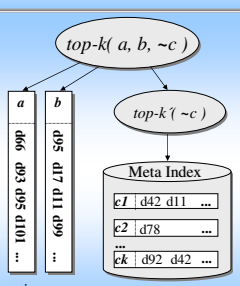
- **Performance plots as a function of ε**





Future Work: XML Support & Top-k Operator

- **Nested top-k operator with native support for XML:**
  - XPath with XLink/XPointer support and a distance-based ranking (HOPI [2] index)
- **Ontological query expansion**
  - Meta index lists for approximate matches, disambiguation, and a concept-based ranking [3]



References:
- [1] R. Fagin, A. Lotem, M. Naor: Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci. 66(4), 2003.
- [2] R. Schenkel, A. Theobald, G. Weikum: HOPI: An efficient connection index for complex XML document collections. EDBT 2004.
- [3] M. Theobald, R. Schenkel, G. Weikum: Exploiting structure, annotation, and ontological knowledge for classification of semistructured data. WebDB 2003.

Martin Theobald **R. 407**

MAX-PLANCK-GESELLSCHAFT · MPI INFORMATIK · International Max Planck Research School MPI for Computer Science · UNIVERSITÄT DES SAARLANDES · DFKI