Top-k Query Evaluation with Probabilistic Guarantees

Martin Theobald
Gerhard Weikum
Ralf Schenkel

Max-Planck Institute for Computer Science
Saarbrücken
Germany

Outline

- Computational Model
- Fagin's NRA as baseline
- Probabilistic threshold test & score predictions
- Queuing strategies for prob. candidate pruning
- Probabilistic guarantees for top-k results
- Experiments

Related Work

- Thresholding algorithms for top-k queries
  - Fagin [99, 01, 03]: TA, NRA, CA
  - Balke, Güntzer & Kießling [00, 01]
- Variants for multimedia search & structured databases
  - Natsev et al. [01], Agrawal & Chaudhuri [03], Chaudhuri & Gravano [04]
- R-tree-like multidimensional indexes & range queries
  - Hjaltason & Samet [99,03], Böhm et al. [01], Bruno & Gravano [02], Ciaccia & Patella [02], Amato et al. [03]
- Multidimensional nearest-neighbor queries
  - Donjerkovic & Ramakrishnan [99], Ciaccia & Patella [00], Tao & Faloutsos & Papadias [03], Singitham et al. [04]

Computational Model for Top-k Queries over an m-Dimensional Data Space

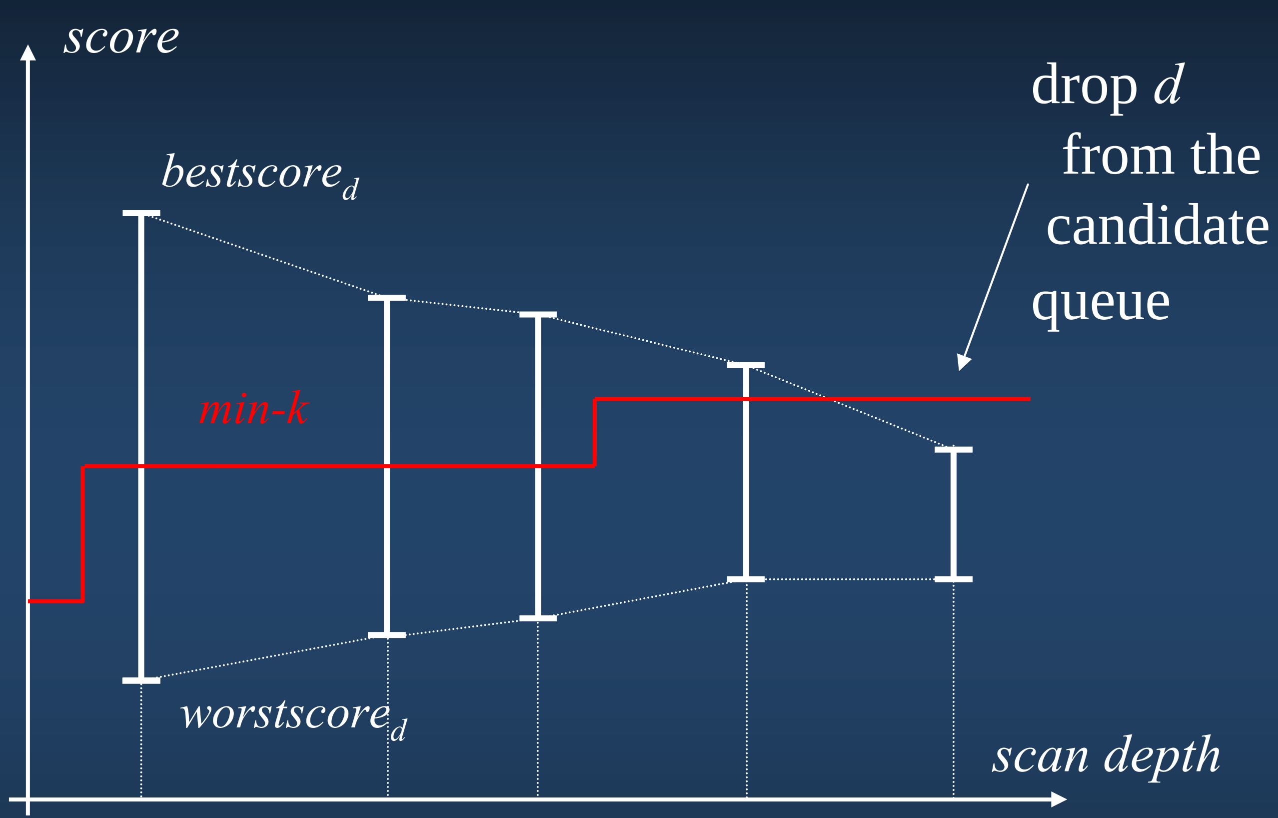
- Cartesian product space $D_1 \times \ldots \times D_m$ and a data set $D \subseteq D_1 \times \ldots \times D_m$ of m-dimensional data points with $s_i \in D \subseteq \Re^m$ or $s_i \in D \subseteq [0,1]^m$
- Monotonous score aggregation function
  $aggr: (D_1 \times \ldots \times D_m) \times (D_1 \times \ldots \times D_m) \rightarrow [0,1] \text{ or } \Re_0^+$
- Partial match queries
  - Weak dimensions can be compensated by stronger ones
- Possible score aggregation functions
  - *min, max, sum, product* (sum over $log\ s_i$), *cosine* (normalized vectors)
- Application examples
  - Multimedia data, text documents, web documents, structured records (e.g., product catalogs)
- Access model
  - Inverted index over long index lists
    → expensive random IO's & mainly sorted accesses

Fagin's NRA$_{[PODS\ '01]}$ at a Glance

- ***NRA(q,L):***

top-k := ∅; candidates := ∅; min-k := 0;
scan all lists L_i (i = 1..m) in parallel:
    consider item d at position pos_i in L_i;
    E(d) := E(d) ∪ {i};
    $high_i$:= $s_i(q_i,d)$;
    worstscore(d) := $aggr\{s_\nu(q_\nu,d) | \nu \in E(d)\}$;
    bestscore(d):= $aggr\{aggr\{s_\nu(q_\nu,d) | \nu \in E(d)\}, aggr\{high_\nu | \nu \notin E(d)\}\}$;
    if worstscore(d) > min-k then
        remove $argmin_{d'}$\{worstscore(d')|d' ∈ top-k\} from top-k;
        add d to top-k
        min-k := min\{worstscore(d') | d' ∈ top-k\};
    else if bestscore(d) > min-k then
        candidates := candidates ∪ \{d\};
    threshold := max \{bestscore(d') | d' ∈ candidates\};
    if threshold ≤ min-k then exit;

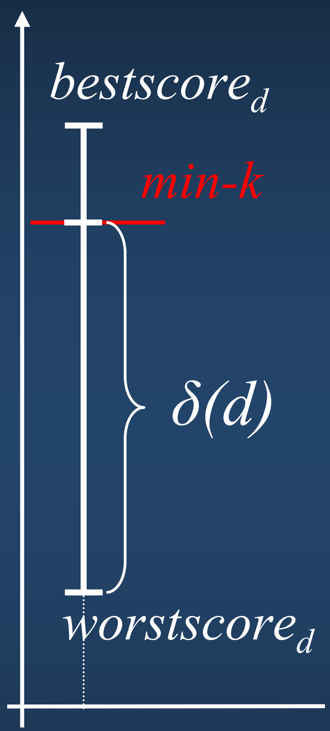
Evolution of a Candidate's Score



- *Approximate top-k*

  "What is the probability that *d* qualifies for the top-k ?"

Safe Thresholding vs. Probabilistic Guarantees



- NRA based on invariant

$$\underbrace{\sum_{i \in E(d)} s_i(d)}_{worstscore_d} \le s(d) \le \underbrace{\sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} high_i}_{bestscore_d}$$

- Relaxed into *probabilistic threshold test*

$$p(d) := P\left[\sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} s_i(d) > \min_k\right] \le \varepsilon$$

- Or equivalently, with $\delta(d) := \min_k - \sum\{s_i \mid i \in E(d)\}$

$$p(d) = P\left[\sum_{i \notin E(d)} s_i(d) > \delta(d)\right] \le \varepsilon$$

Probabilistic Score Predictions

- Assuming feature independence
  - Uniform
  - Zipf
  - Poisson
  - Histograms

- With feature correlations
  - Multi-dimensional histograms
  - Moment-Generating Functions & Chernoff-Hoeffding bounds *[Siegel `95]*

Guarantees with Uniform Distributions

- Consider score intervals *[0, $high_i$]*
- Treat each s_i as random variable S_i and predict $P\left[\sum_i S_i > \delta\right]$
- For two random variables S_1 and S_2
  - Density functions $f_1(x)=1/high_1$ and $f_2(x)=1/high_2$
  - Consider convolution $f(x) = \int_0^x f_1(z) f_2(x-z)\, dz$
  - ..but each factor is non-zero in $0 \le z \le high_1$ and $0 \le x\text{-}z \le high_2$

    → awkward amount of case differentiations
- Instead, consider *moment-generation functions (i > 1)*
  - Of the form $M_i(s) = \int_0^s e^{sx} f_i(x)\, dx = E\left[e^{sS_i}\right]$
  - Consider convolution $M(s) = \prod_i M_i(s)$
- Apply *Chernoff-Hoeffding* bounds on the tail probabilities

  $P\left[\sum_i S_i > \delta\right] \le \inf_{s \ge 0}\{e^{-s\delta} M(s)\}$
- Ability to capture **feature correlations** or **heterogeneous distributions**

Guarantees with Poisson Estimators

- Approximate *tf·idf* scores by a Poisson distribution truncated over $[0, high_i]$
- Reasonably fits large web corpora, e.g., "*.Gov*"
- Descretize all s_i in index list L_i with n_i items:
  - Let S_i be a discrete RV with n_i equi-distant values $v_j = 1 - j \cdot high_i / n_i$
  - then $P\left[S_i = v_k\right] = e^{-\lambda_i} \frac{\lambda_i^{\,k}}{k!}$ and $\sum_{j=1..l} P[S_i = v_j]$ approximated by *Incomplete Gamma Fct.*
- Consider convolution *(i>1)*
  - where $P\left[S = v_{k'}\right] = e^{-\lambda_r} \frac{\lambda_r^{\,k}}{k!}$ with $\lambda_r = \sum_i \lambda_i$
- Consider conditional probabilities

$$
\begin{aligned}
&P\left[\textstyle\sum_{i \notin E(d)} S_i > v \mid S_i \le high_i \text{ for } i \notin E(d)\right] \\
&= 1 - P\left[\textstyle\sum_{i \notin E(d)} S_i \le v \mid S_i \le high_i \text{ for } i \notin E(d)\right] \\
&= 1 - \frac{P\left[\sum_{i \notin E(d)} S_i \le v \wedge (S_i \le high_i \text{ for } i \notin E(d))\right]}{\prod_{i \notin E(d)} P\left[S_i \le high_i\right]}
\end{aligned}
$$

Guarantees with Histograms

- Capture arbitrary distributions in a compact histogram
  - Consider *n* buckets *(lb,ub]* with *lb[j]=j/n* and *ub[j]=(j+1)/n*
  - Store the frequency *freq[i]* and the cumulative frequency *cum[i]* of scores for each cell
- Consider convolution

$$H.freq[j] = \sum_{(j_1,\dots,j_t)\ with\ j_1+\dots+j_r=i} H_1.freq[j_1]\cdot\ldots\cdot H_r.freq[j_r]$$

$$H.cum[j] = \sum_{l=0..j} H.freq[l]$$

→ in $O(mn^2)$ time using a binary convolution operation

- Consider conditional probabilities

$$P\left[\sum_{i\notin E(d)} S_i > \delta \mid S_i \le high_i\ for\ i \notin E(d)\right]$$

→ by truncating the basic histograms

Queuing Strategies for Probabilistic Candidate Pruning

- Non-negligible prediction overhead
  - 2^m-1 possible convolutions of remainder dimensions per query
  - Frequent predictor updates due to decreasing $high_i$
- Prob. threshold test *embedded* into NRA
  - Periodic candidate pruning (after r sorted accesses)
  - “Reusable” convolutions are temporarily cached
- Queuing as a key role for query evaluation
  - Which candidates are tested?
  - How often is a candidate tested?
  - What actions are taken when a candidate fails the test?

Conservative Queuing

- *Prob-conservative*
  - 2^m-1 queues per query
  - Group candidates by remainder sets *{1..m} – E(d)*
  - Top candidate dominates all candidates within each queue
  - Test top candidate only
  - For all queues *q* :

    Drop queue *q*, if

    *P[**top(q)** can qualify for top-k] < ε*

- *Prob-progressive*
  - 1 queue per query
  - Merge all candidates by their bestscores
  - No dominating candidate in terms of prob. prediction
  - Test all candidates periodically
  - For all candidates *d* in *q* :

    Drop candidate *d*, if

    *P[**d** can qualify for top-k] < ε*

- Stop by safe min-k threshold test or when all queues are empty

Aggressive Queuing

- *Prob-smart*
  - 1 bounded queue per query
  - Merge all candidates by their bestscores
  - No dominating candidate in terms of prob. prediction
  - Update & rebuild entire queue
  - Bound size by *b* top candidates after each rebuild
- *Prob-aggressive*
  - No queue
  - Consider virtual candidate d_v with $E(d_v) = \varnothing$
  - d_v dominates all yet unseen candidates

- Stop heuristically, if
  $P[\ \mathbf{top(q)}\ \textit{can qualify for top-k}] < \varepsilon$
- Stop heuristically, if
  $P[\ \mathbf{d_v}\ \textit{can qualify for top-k}] < \varepsilon$

Guarantees for Top-k Results

- For *Prob-con* and *Prob-pro*
  - Probability p_{miss} of missing a true top-k object equals the probability of erroneously dropping a candidate from the queue
  - For each candidate $p_{miss} \leq \varepsilon$
  - *P[recall = r/k] = P[precision = r/k] =*

$$\binom{k}{r}(1-p_{miss})^r \, p_{miss}{}^{(k-r)} \leq \binom{k}{r}(1-\varepsilon)^r \, \varepsilon^{(k-r)}$$

  - *E[precision] = E[recall] =*

$$\sum_{r=0..k} P[precision = r/k] \cdot r/k \; = \; (1-\varepsilon)$$

  - Lower expected values for *Prob-smart* and *Pro-aggr*

Experimental Setup

- Data collections
  - Gov
    - TREC-12 Web Track's .Gov collection
    - 1.250.000 web documents (html, doc, pdf)
    - 50 keyword queries from the Topic Distillation task, $m \leq 5$

      e.g., "legalization marijuana"
  - XGov
    - Gov with manual query expansion, $m \leq 20$

      e.g., "legalization law marijuana cannabis drug abuse pot …"
  - IMDB
    - Structured (XML) version of the Internet Movie Data Base
    - 375.000 movie files, 1.200.000 person files
    - Mixed text and categorical attributes: Genre, Actor, Description
    - 20 queries of the type:

      "Genre $\supseteq$ {Western} $\wedge$ Actor $\supseteq$ {John Wayne, Katherine Hepburn}
      $\wedge$ Description $\supseteq$ {Sheriff, Marshall}"

Evaluation Metrics

- Efficiency
  - *#sorted accesses*
  - *time:* wall-clock elapsed time
  - *memory:* peak level of working memory for all priority queues

- Quality
  - *Precision & recall* (at macro-average)
  - $\textit{Rank distance} := \frac{1}{k} \sum_{i=1..k} \left| i - truerank(i) \right|$
  - $\textit{Score error} := \frac{1}{k} \sum_{i=1..k} \left| score_i^{(approx)} - score_i^{(exact)} \right|$

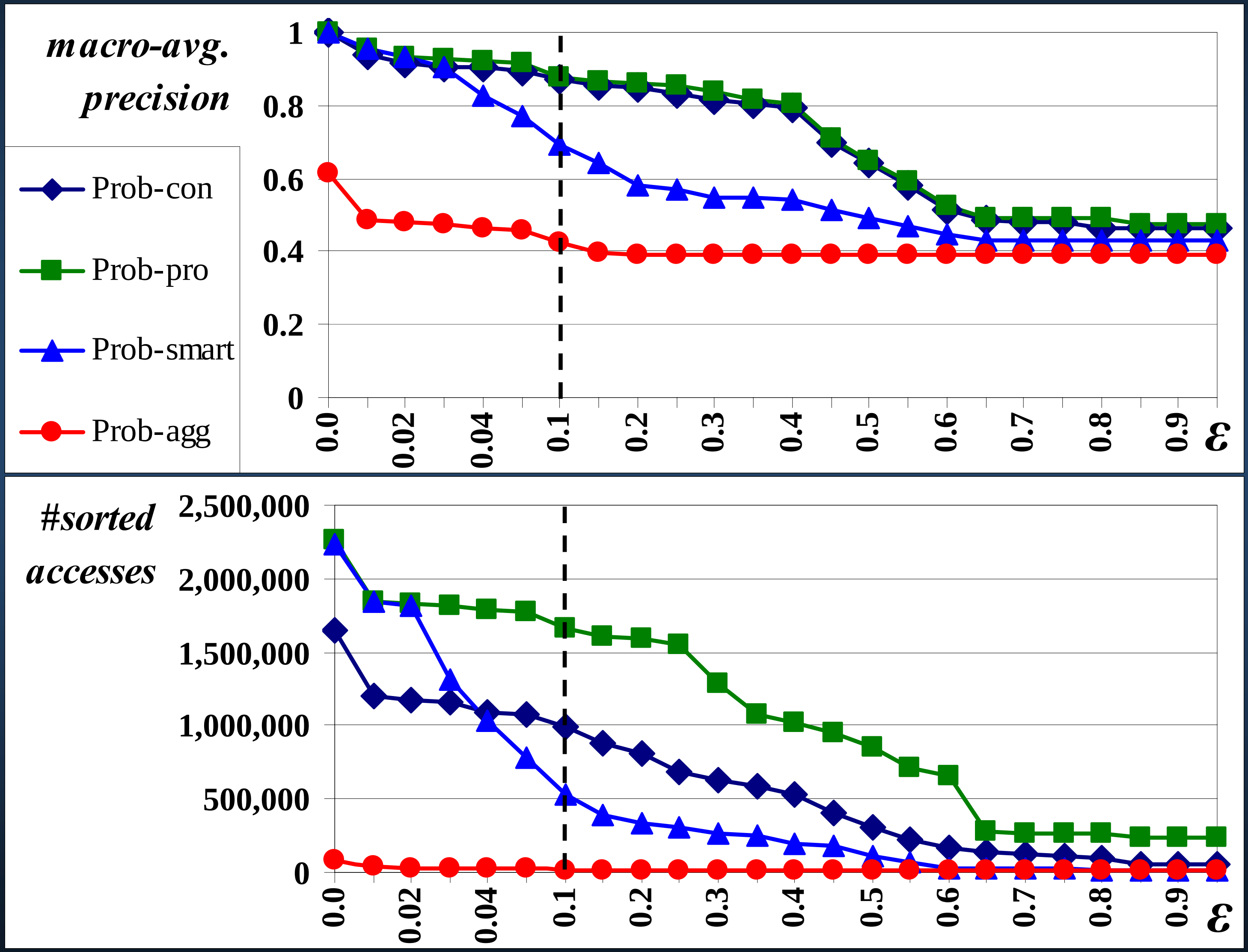
Baseline

k=20, ε=0.1

		# sorted accesses	execution time (sec.)	max. queue size	macro-avg. precision	rank distance	score distance
Gov	*NRA*	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
XGov	*NRA*	22,403,490	7,908	70,896	1.0	0.0	0.0
	Prob-con	10,165,677	6,448	51,893	0.90	10.9	0.038
	Prob-pro	20,006,283	1,791	12,435	0.95	9.3	0.031
	Prob-smart	18,287,636	1,066	400	0.88	14.5	0.035
	Prob-agg	133,745	2	0	0.35	80.7	0.182
IMDB	*NRA*	1,003,650	201.9	12,628	1.0	0.0	0.0
	Prob-con	463,562	17.8	14,990	0.71	119.9	0.18
	Prob-pro	490,041	69.0	9,173	0.75	122.5	0.14
	Prob-smart	403,981	12.7	400	0.54	126.7	0.25
	Prob-agg	41,821	0.7	0	0.18	171.5	0.39

Sensitivity Studies for Gov:

precision vs. sorted accesses for k = 20



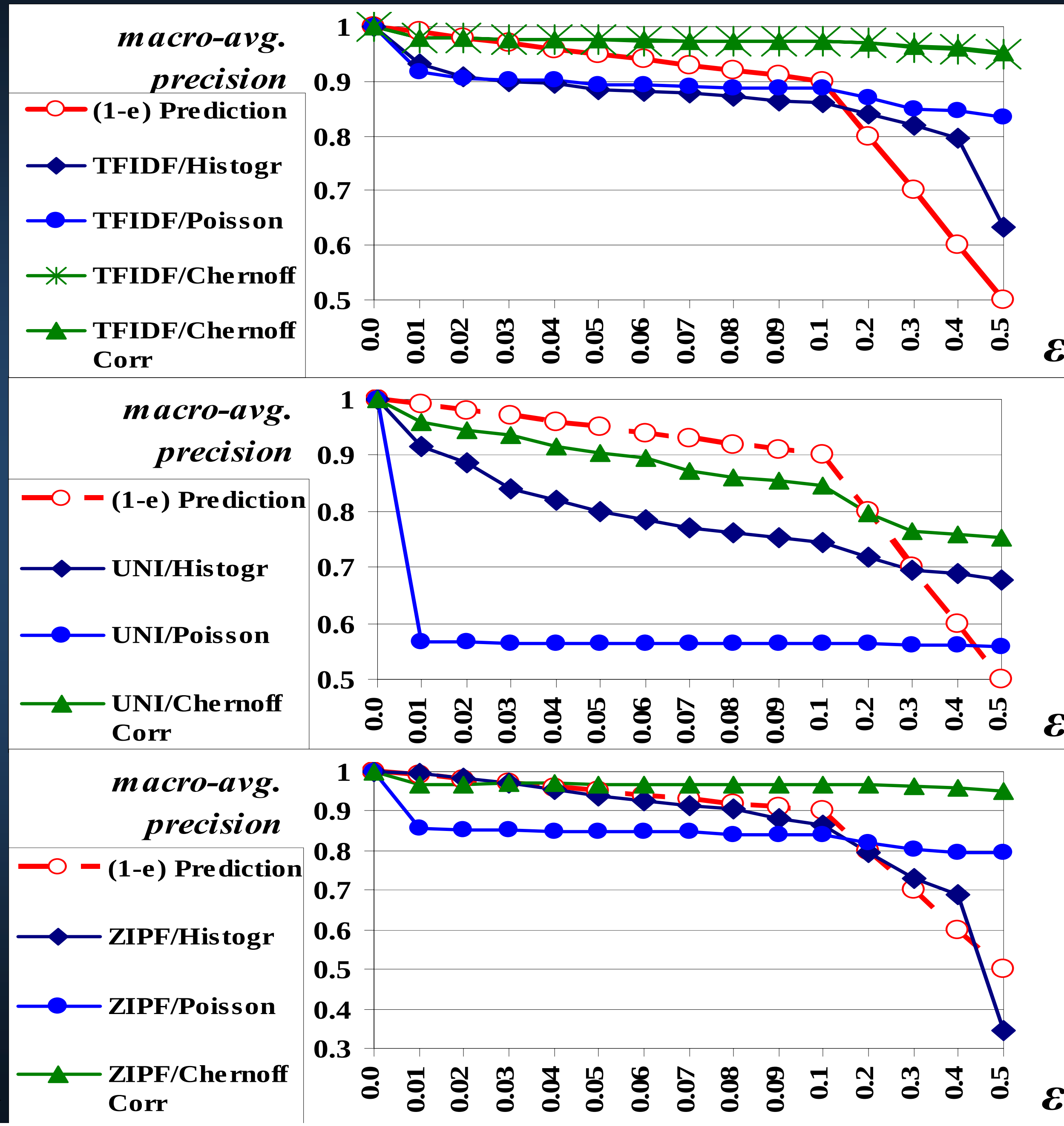
Impact of Probabilistic Predictions for Gov

Original scores

- *tf·idf* weights

Artificially generated score distributions

- Uniform
- Zipf



Conclusions & Ongoing Work

- **“Top-k is a heuristic anyway, so why not do it probabilistically?”**
- Major performance gains of factor of up to 8 at ~80% precision
- **Semantic extensions** for query-specific weights and efficient query expansion
- Applied at this year’s TREC benchmark
  - Robust Track – hard queries
  - Web Track – *tf ·idf* and global *PageRank* scores
  - Terabyte Track – 480 GB web documents
- Further extensions for **XML support** incl. path similarities