Mining Search Engine Query Logs via Suggestion Sampling

Ziv Bar-Yossef
Technion and Google

Maxim Gurevich
Technion
Search Engine Query Logs

- Used by search engines to improve search results
- Contains private information
  - Of users and of the search engine itself
- Not disclosed by search engines
Applications of Query Log Analysis

- Keyword based advertising
- Search quality evaluation
- User modeling
Keyword Based Advertising

- Compare keyword popularity
- Track keyword popularity over time
- Find related keywords
Search Quality Evaluation

- Estimate the amount of undesirable content sent to users
  - Spam
  - Stale results
  - Non-existent results
  - Pornography
  - Hate materials
  - Virus contaminated pages
  - ...

- Estimate search engine bias towards
  - Authoritative sources
  - Certain domains
  - Certain languages
  - ...
User modeling

- Distribution of query types [Broder 02]
  - Navigational
  - Informational
  - Transactional
- Density of commercial queries
- Fraction of geographical queries
- ...

External Query Log Mining

- Sampling (uniform or by popularity)
- Computing aggregate (privacy preserving) functions
Suggest: Trapdoor to Query Logs

- Query auto-completion
  - Suggests query completions
  - More popular first
  - Offered by major search engines
- Backed by a hidden underlying database
  - Assumption: Derived from query logs
Our Contribution

- Algorithm for sampling queries uniformly from the query log using the suggestion service
  - Practical (few suggestion requests)
  - Unbiased
- Algorithm for sampling queries from the query log proportionally to their popularity using the suggestion service
  - Practical (few suggestion requests)
  - Slightly biased
Related Work

- Sampling documents from search engine index
  [BarYossef et al 06,07, Broder et al 07, ...]
  - Different problem and setting
- Sampling from B-trees
  [Wong et al 80, Olken et al 89,95]
  - B-tree specific assumptions
  - Inefficient for query log mining
- Sampling from databases behind web forms
  [Dasgupta et al 07]
  - Different setting, inefficient for query log mining
- Uniform sampling of combinatorial structures
  [Jerrum et al. 86]
  - Theoretical, the basis of our sampling algorithm
Uniform Suggestion Sampling

**Suggestion TRIE**

- **suggestion node**
- **popularity**

**Example:**
- Query prefix: "m"
- Top k (10) suggestions
  - mp3 song
  - mp3
  - mp3 tag

**Goal:** Sample suggestion nodes uniformly
Volume Estimators

- Define: \( \text{volume}(\alpha) = \# \) of suggestions starting with \( \alpha \)

![Diagram showing volume estimators and server connections]

- Example:
  - \( \text{vol}(m) = 3 \)
  - \( \text{vol}(w) = 2 \)

- Illustration of suggestion and volume calculation process.

- Estimate of volume(\( \alpha \))
Random Walk Tree Sampling

- **Assumption**: we have a prefect volume estimator

- **current** = root
- while true
  - If current is a suggestion node
    - Return **current** with probability \(1/\text{volume}(\text{current})\)
  - Go to child \(x\) with probability \(\propto \text{volume}(x)\)

- **Theorem**: If volumes are accurate, the samples are uniform
How To Estimate Volumes

- **Input:** Prefix string $\alpha$
- **Output:** $\approx \#$ of suggestions starting with $\alpha$
- **Naïve estimator:** $\text{volume}(\alpha) \approx \#$ of suggestions the server returns on $\alpha$
- **Popularity based estimator:** $\text{volume}(\alpha) \approx \text{popularity}($most popular suggestion for $\alpha$$)$
  - Rationale: Power Law distribution of popularity
    (procedure for popularity estimation is included in the paper)
- **Sample based estimator:** $\text{volume}(\alpha) \approx \text{normalized number of suggestions for } \alpha \text{ in a previously available query log}$
- **Final estimator** aggregates all the three results
Caveat

- Random Walk Tree Sampler assumed volumes are known perfectly
- We can only approximate volumes (heuristically)

- Suggestion samples are not uniform
Monte Carlo Stochastic Simulation

- \( \pi = \text{target distribution} = \) uniform distribution on suggestions
  - Can deal with other target distributions as well
- \( p = \text{trial distribution} \) on \( S \)
  - Can compute \( p(x) \) for each \( x \)
  - \( p \neq \pi \) but support(\( p \)) should contain support(\( \pi \))
  - \( p \) should be easy-to-sample-from

\[ \pi \rightarrow \text{Sampler} \]
\[ p \rightarrow \text{Sampler} \]
\[ S \]
\[ y_1, \ldots, y_k \text{ from } p \]

Monte Carlo Simulator

- A sample \( x \) from \( \pi \)
Rejection Sampling

[von Neumann 63]

- \textbf{accepted} := false
- while (not \textbf{accepted})
  - Sample suggestion \( q \) from \( p \)
  - Calculate \( p(q) \) and \( \pi(q) \)
  - Toss a coin whose heads probability is \( \frac{\pi(q)}{C \cdot p(q)} \)
  - if coin comes up heads, \textbf{accepted} := true
- return \( q \)

\[
\Pr(q \text{ accepted}) = p(q) \cdot \frac{\pi(q)}{C \cdot p(q)} \propto \pi(q)
\]
Recap – Uniform Sampling

- Suggestion Server
- Tree Sampler
- Volume Estimator
- Monte Carlo Simulator ($\pi = \text{uniform}$)

Samples $y_1, \ldots, y_k$ from $p$

- ~6000 suggestion server requests per uniform sample
Popularity based suggestion sampling

- Assumptions
  - Popularity is distributed according to Power Law
  - The Power Law exponent is known apriori
- Basic building block: Popularity Estimator (PE)

\[ \pi(y) = \text{PE}(y) \]

Basic building blocks:
- Monte Carlo Simulator
- Volume Estimator
- Tree Sampler
- Suggestion Server
Sampling Bias

![Bar chart showing sampling bias]

The bar chart above illustrates the percentage of suggestions in a sample, comparing uniform sampling with sampling based on popularity. The x-axis represents the deciles of suggestions ordered by popularity, while the y-axis shows the percentage of suggestions in the sample. The chart highlights the discrepancy between uniform sampling and popularity-based sampling, indicating a significant bias towards more popular suggestions in the latter.
Coverage of sampled queries by Wikipedia

![Bar chart showing coverage of queries by Wikipedia for SE1 and SE2 categories. The chart compares uniform and popularity-based queries. SE2 shows higher coverage compared to SE1.](chart.png)
Percent of non-existent search results

![Graph showing percent of dead pages for SE1 and SE2.](image)
Conclusions

- Algorithms for sampling queries randomly from a search engine query log
  - Uniformly or by popularity
  - Useful for keyword based advertising, search engine evaluation, user behavior studies
  - Via public suggestion interface only
  - Practical (accurate and efficient)
Thank You