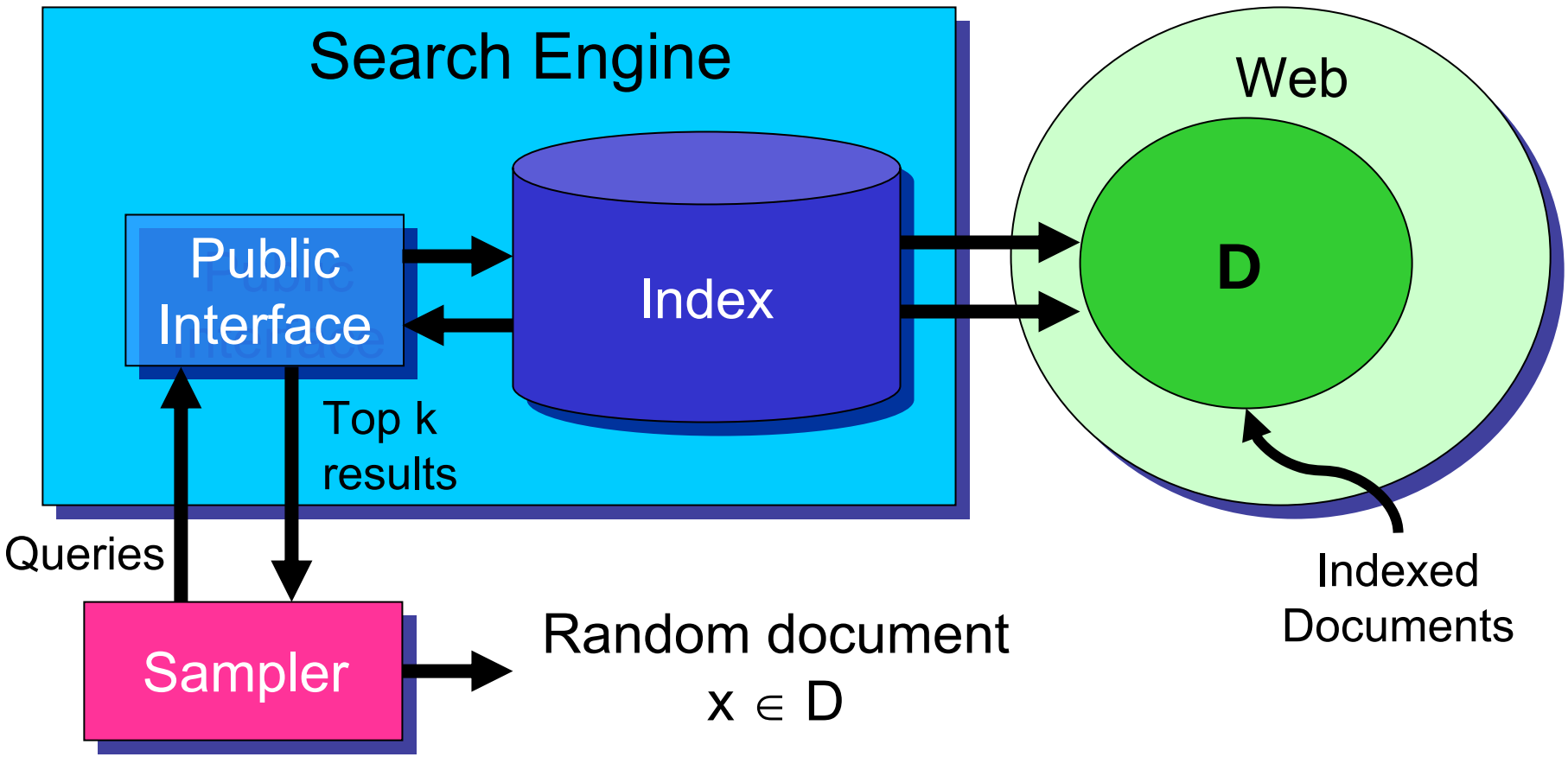
Random Sampling from a Search Engine‘s Index

Ziv Bar-Yossef
Maxim Gurevich

Department of Electrical Engineering
Technion

Search Engine Samplers



Motivation

- Useful tool for search engine evaluation:
  - Freshness
    - Fraction of up-to-date pages in the index
  - Topical bias
    - Identification of overrepresented/underrepresented topics
  - Spam
    - Fraction of spam pages in the index
  - Security
    - Fraction of pages in index infected by viruses/worms/trojans
  - Relative Size
    - Number of documents indexed compared with other search engines

Size Wars

August 2005

YAHOO! : We index **20 billion documents**.

September 2005

Google : We index **8 billion documents,** but our index is **3 times larger** than our competition's.

So, who's right?

Related Work

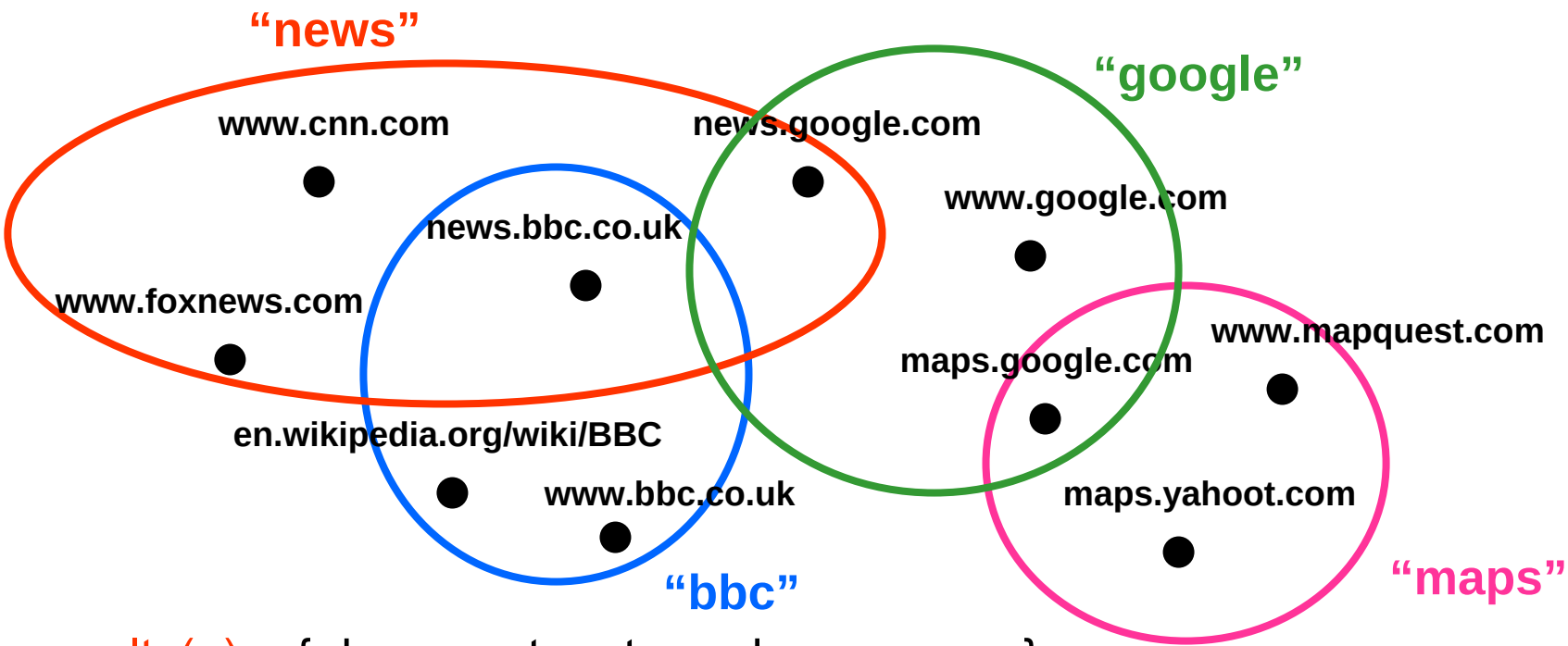
- Random Sampling from a Search Engine's Index
  [BharatBroder98, CheneyPerry05, GulliSignorni05]
- Anecdotal queries
  [SearchEngineWatch, Google, BradlowSchmittlein00]
- Queries from user query logs
  [LawrenceGiles98, DobraFeinberg04]
- Random sampling from the whole web
  [Henzinger et al 00, Bar-Yossef et al 00, Rusmevichientong et al 01]

Our Contributions

- A pool-based sampler
  - Guaranteed to produce near-uniform samples
- A random walk sampler
  - After sufficiently many steps, guaranteed to produce near-uniform samples
  - Does not need an explicit lexicon/pool at all!

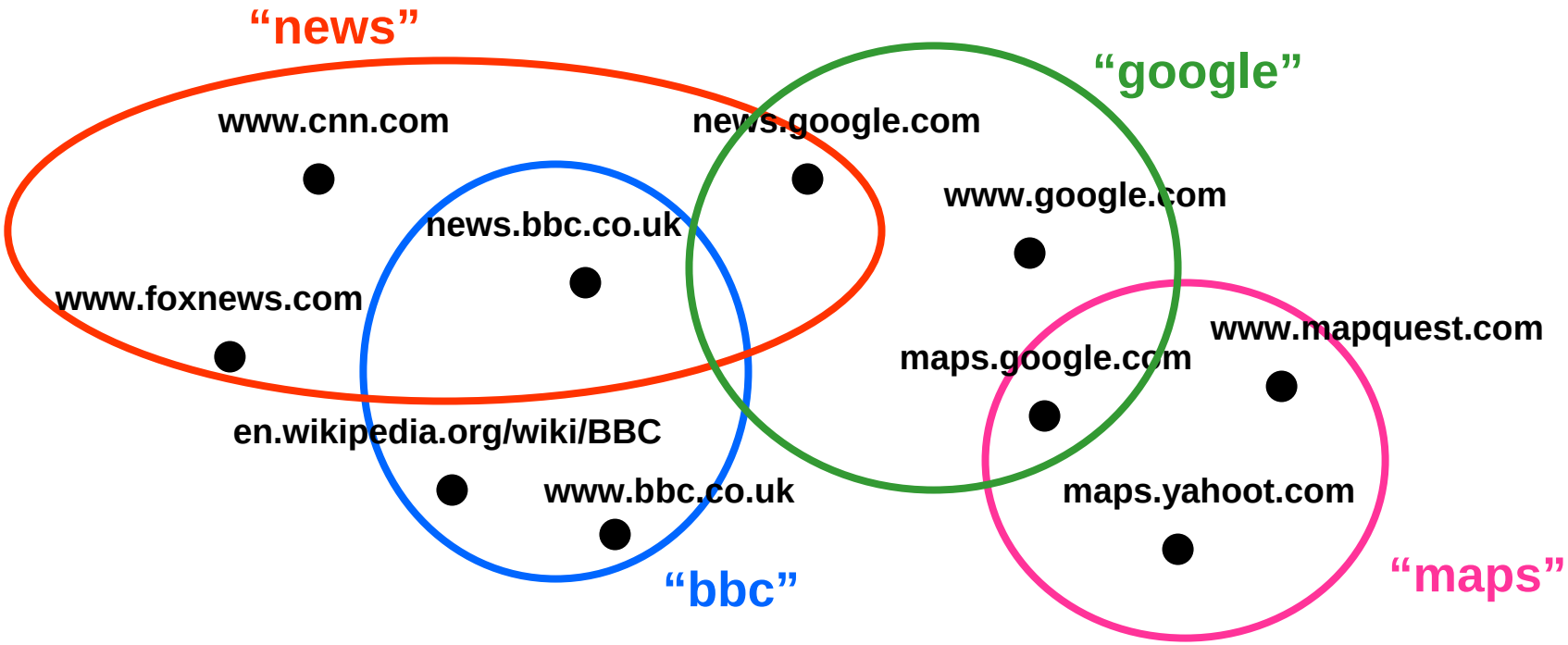
} Focus of this talk

Search Engines as Hypergraphs



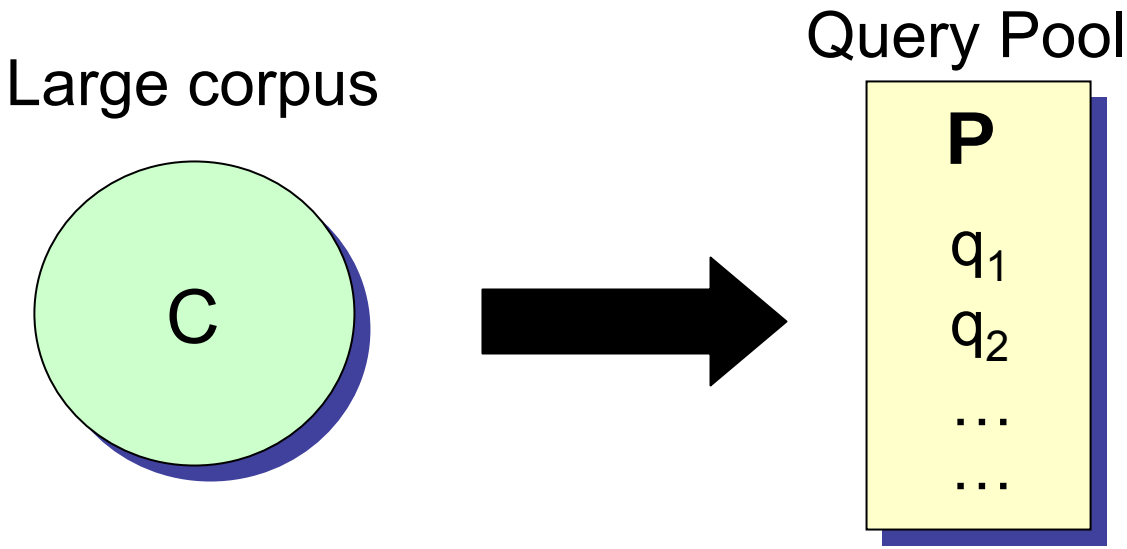
- results(q) = { documents returned on query q }
- queries(x) = { queries that return x as a result }
- P = query pool = a set of queries
- Query pool hypergraph:
  - **Vertices:** Indexed documents
  - **Hyperedges:** { result(q) | q ∈ P }

Query Cardinalities and Document Degrees



- Query cardinality: card(q) = |results(q)|
- Document degree: deg(x) = |queries(x)|
- Examples:
  - card(“news”) = 4, card(“bbc”) = 3
  - deg(www.cnn.com) = 1, deg(news.bbc.co.uk) = 2

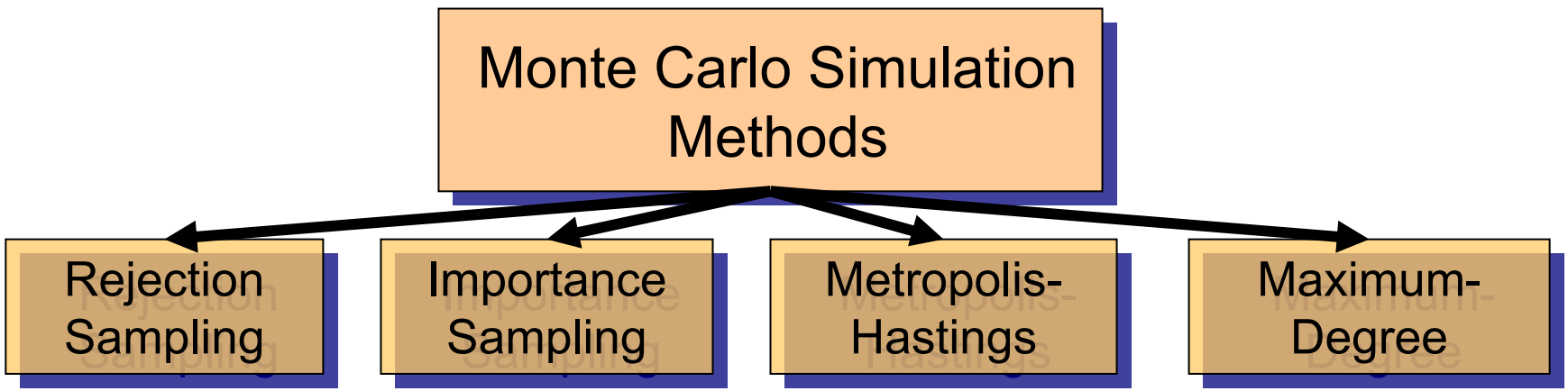
The Pool-Based Sampler:
Preprocessing Step



- Example: P = all 3-word phrases that occur in C
  - If "**to be or not to be**" occurs in C, P contains:
    - "**to be or**", "**be or not**", "**or not to**", "**not to be**"
- Choose P that "covers" most documents in D

Monte Carlo Simulation

- We don't know how to generate uniform samples from D directly
- How can we use biased samples to generate uniform samples?
- Samples with weights that represent their bias can be used to simulate uniform samples



Document Degree Distribution

- We are able to generate biased samples from the "document degree distribution"

$$p(x) = \frac{\deg(x)}{\sum_{x' \in D} \deg(x')}$$

- Advantage: Can compute weights representing the bias of p:

$$w_p(x) = \frac{1}{\deg(x)}$$

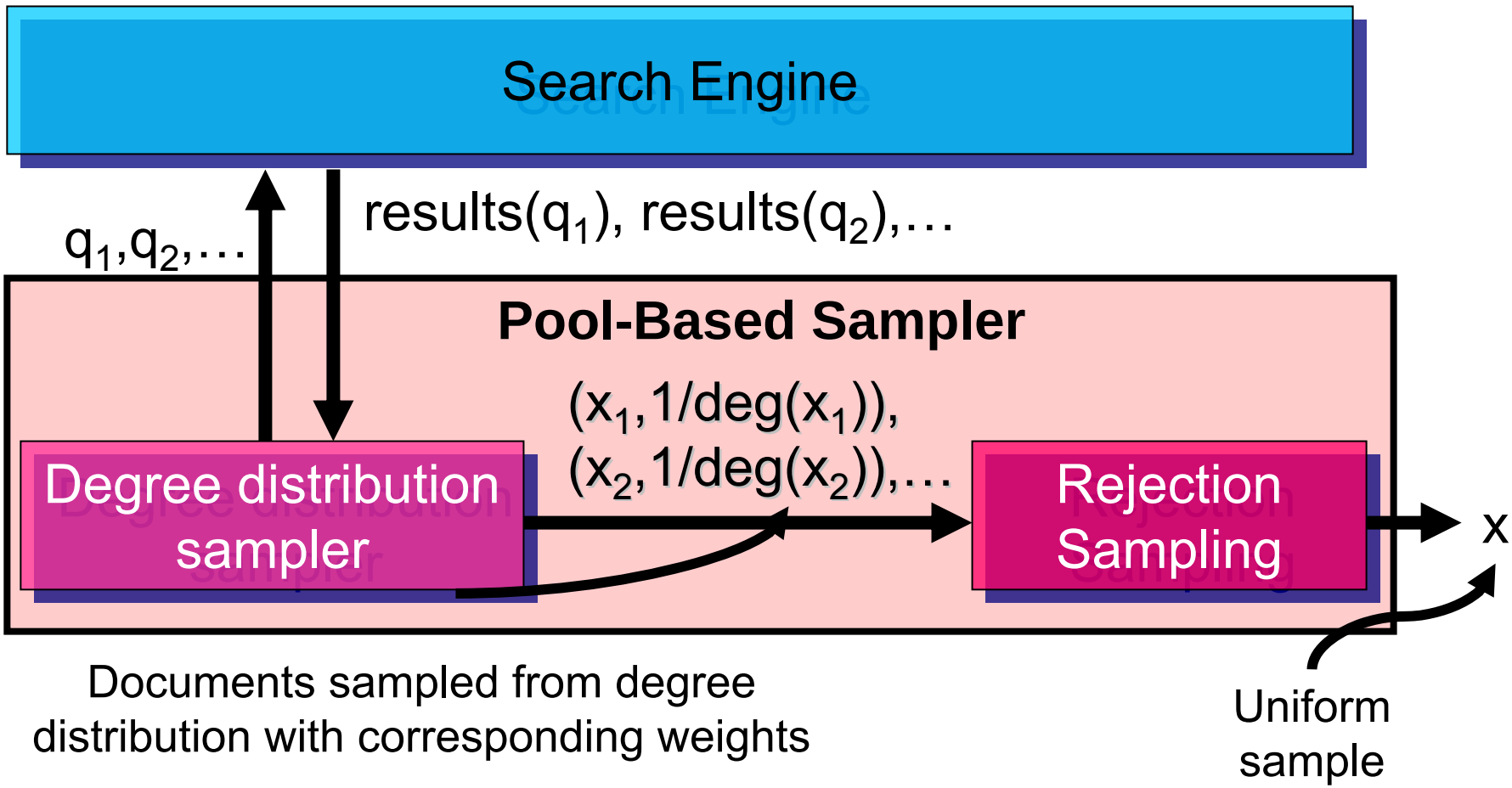
Rejection Sampling [von Neumann]

$$p(x) = \frac{\deg(x)}{\sum_{x' \in D} \deg(x')}, \quad w_p(x) = \frac{1}{\deg(x)}$$

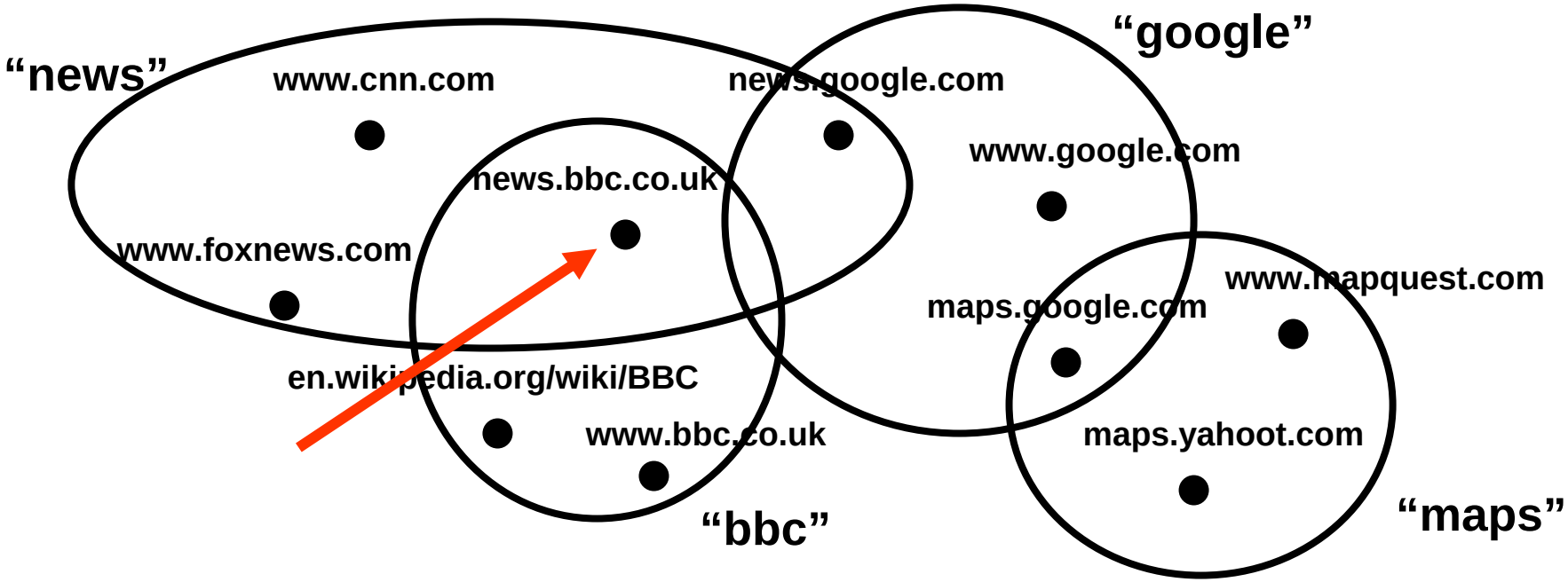
- accept := false
- while (not accept)
  - generate a sample x from p
  - toss a coin whose heads probability is $w_p(x)$
  - if coin comes up heads,
    accept := true
- return x

Pool-Based Sampler

- Degree distribution: $p(x) = \deg(x) / \Sigma_{x'} \deg(x')$



Sampling documents by degree



- Select a random q $\in$ P
- Select a random x $\in$ results(q)
- Documents with high degree are more likely to be sampled
- If we sample q uniformly → “oversample” documents that belong to narrow queries
- We need to sample q proportionally to its cardinality

Sampling queries by cardinality

- Sampling queries from pool uniformly: Easy
- Sampling queries from pool by cardinality: Hard
  - Requires knowing cardinalities of all queries in the search engine

- Use Monte Carlo methods to simulate biased sampling via uniform sampling:
  - Sample queries uniformly from P
  - Compute "cardinality weight" for each sample:

$$w(q) = \frac{\mathrm{card}(q)}{k}$$

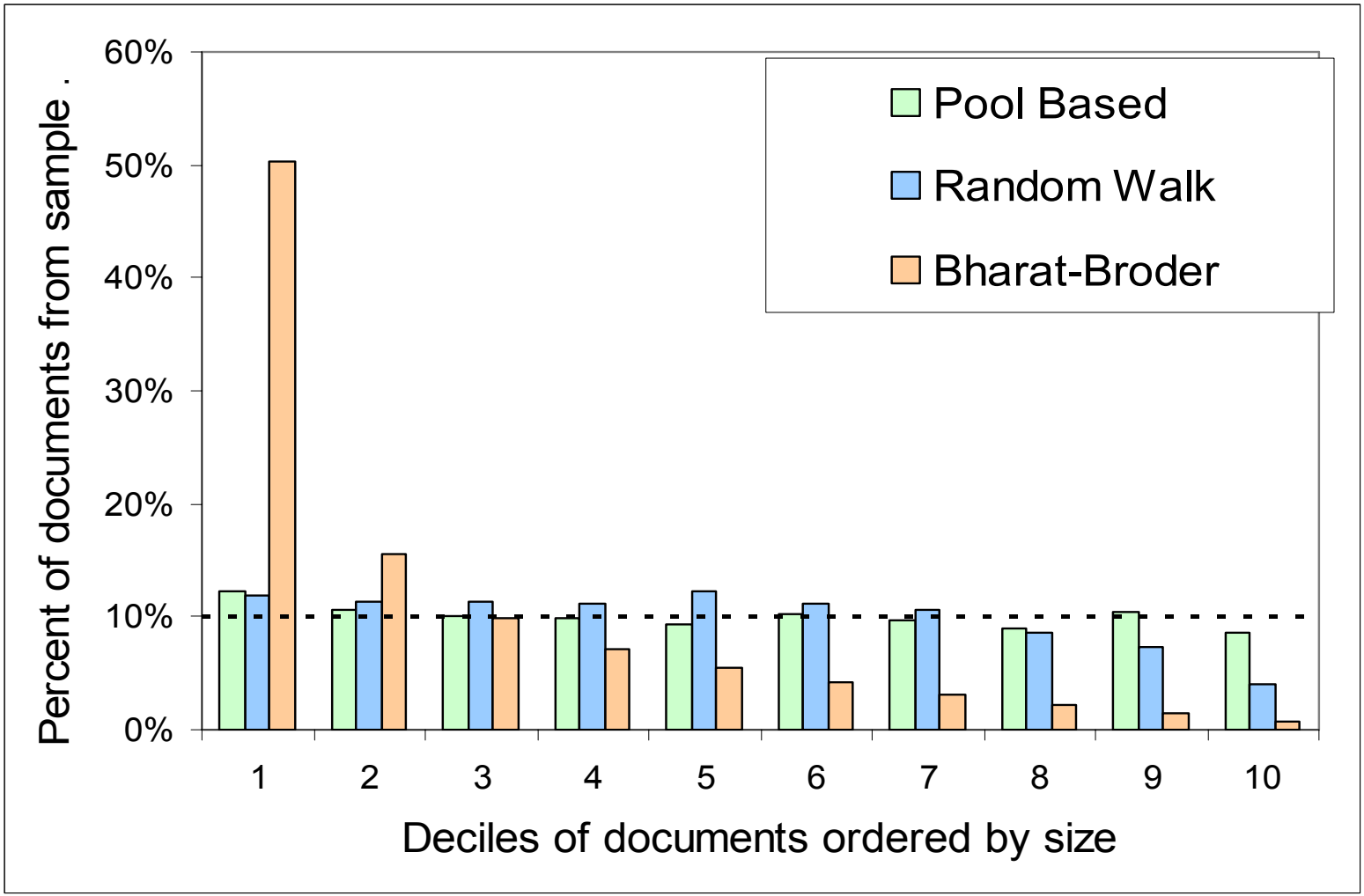
  - Obtain queries sampled by their cardinality

Dealing with Overflowing Queries

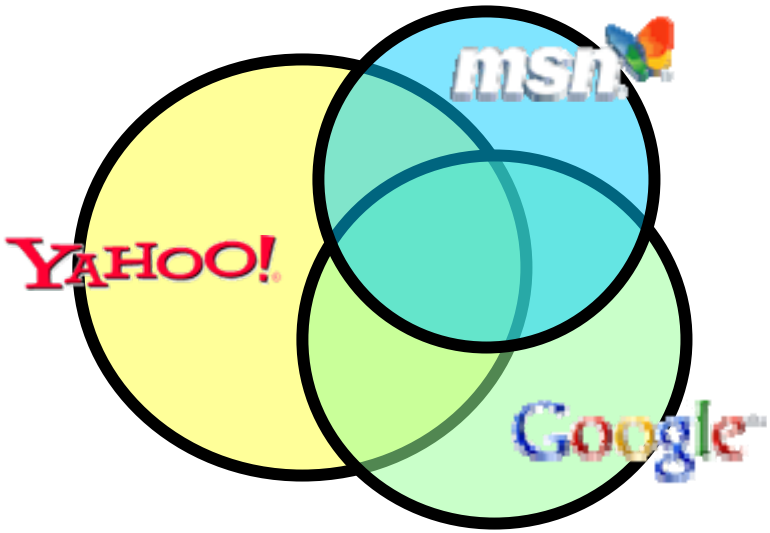
- Problem: Some queries may overflow (card(q) > k)
  - Bias towards highly ranked documents
- Solutions:
  - Select a pool P in which overflowing queries are rare (e.g., phrase queries)
  - Skip overflowing queries
  - Adapt rejection sampling to deal with approximate weights

Theorem:
Samples of PB sampler are at most β-away from uniform. (β = overflow probability of P)

Bias towards Long Documents



Relative Sizes of Google, MSN and Yahoo!



Google = 1

Yahoo! = 1.28

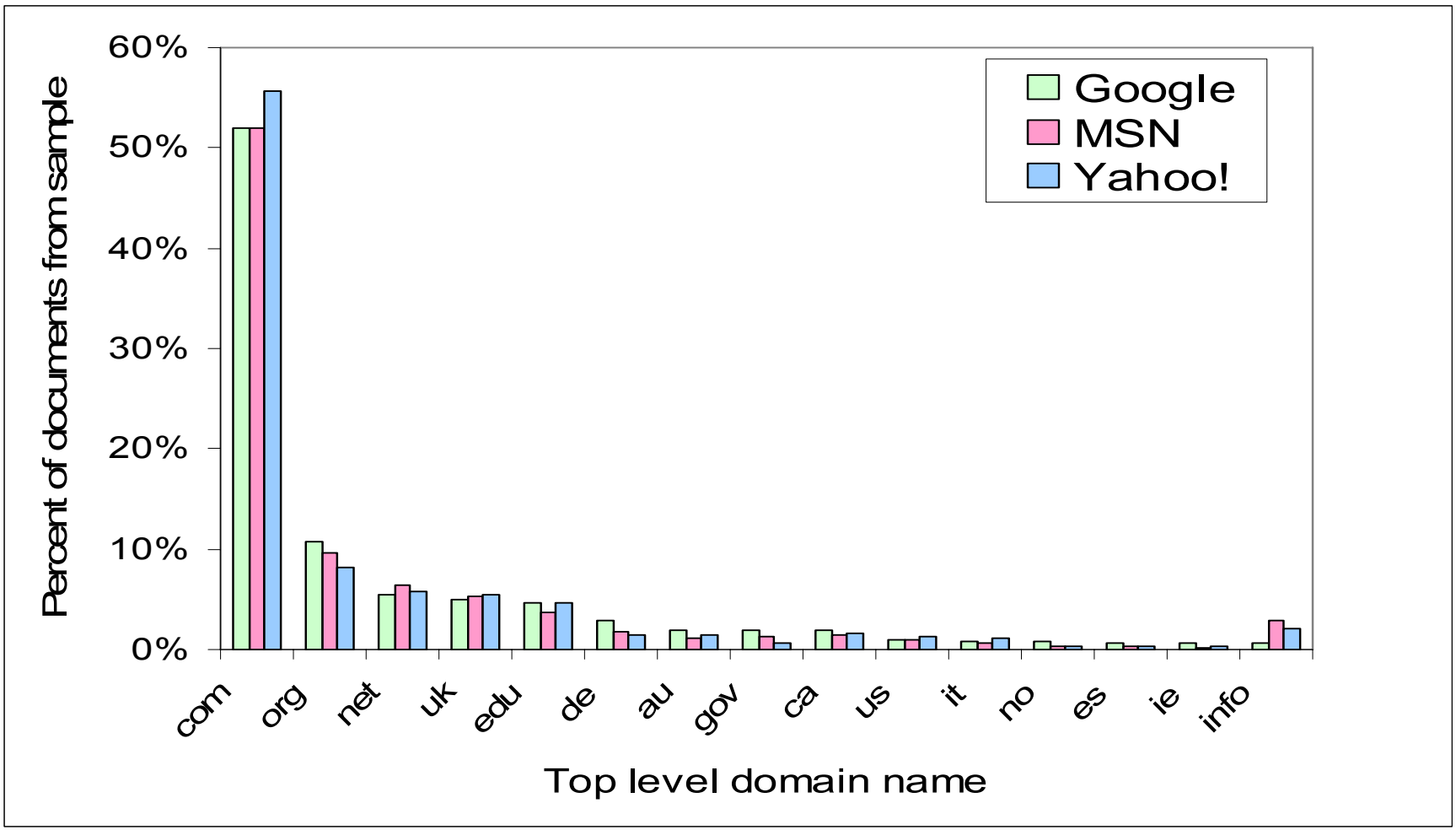
MSN Search = 0.73

Conclusions

- Two new search engine samplers
  - Pool-based sampler
  - Random walk sampler
- Samplers are guaranteed to produce near-uniform samples, under plausible assumptions.
- Samplers show no or little bias in experiments.

Thank You

Top-Level Domains in Google, MSN and Yahoo!



Query Cardinality Distribution

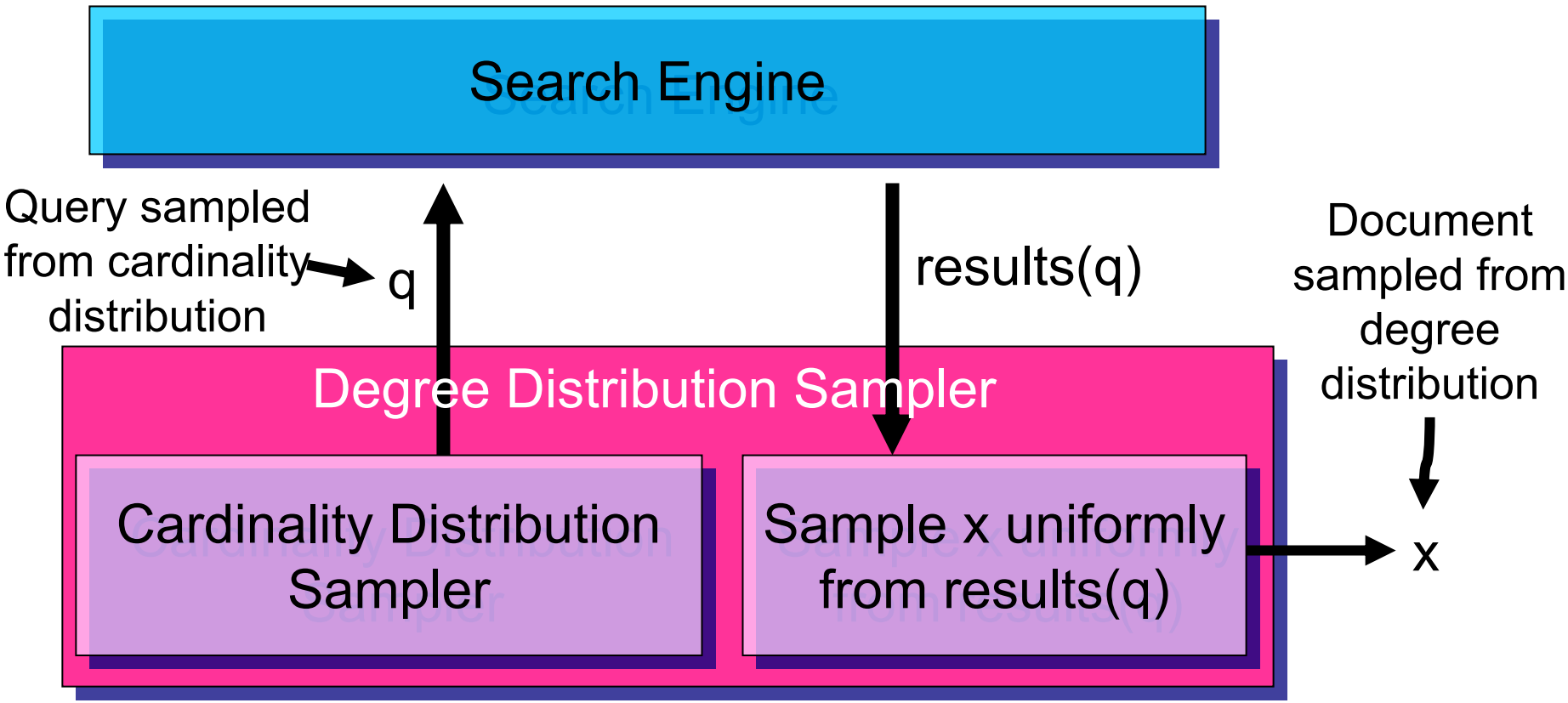
- results(q) = { documents returned on query q }
- card(q) = |results(q)|
- Cardinality distribution:

$$\mu(q) = \frac{\text{card}(q)}{\sum_{q' \in P} \text{card}(q')}$$

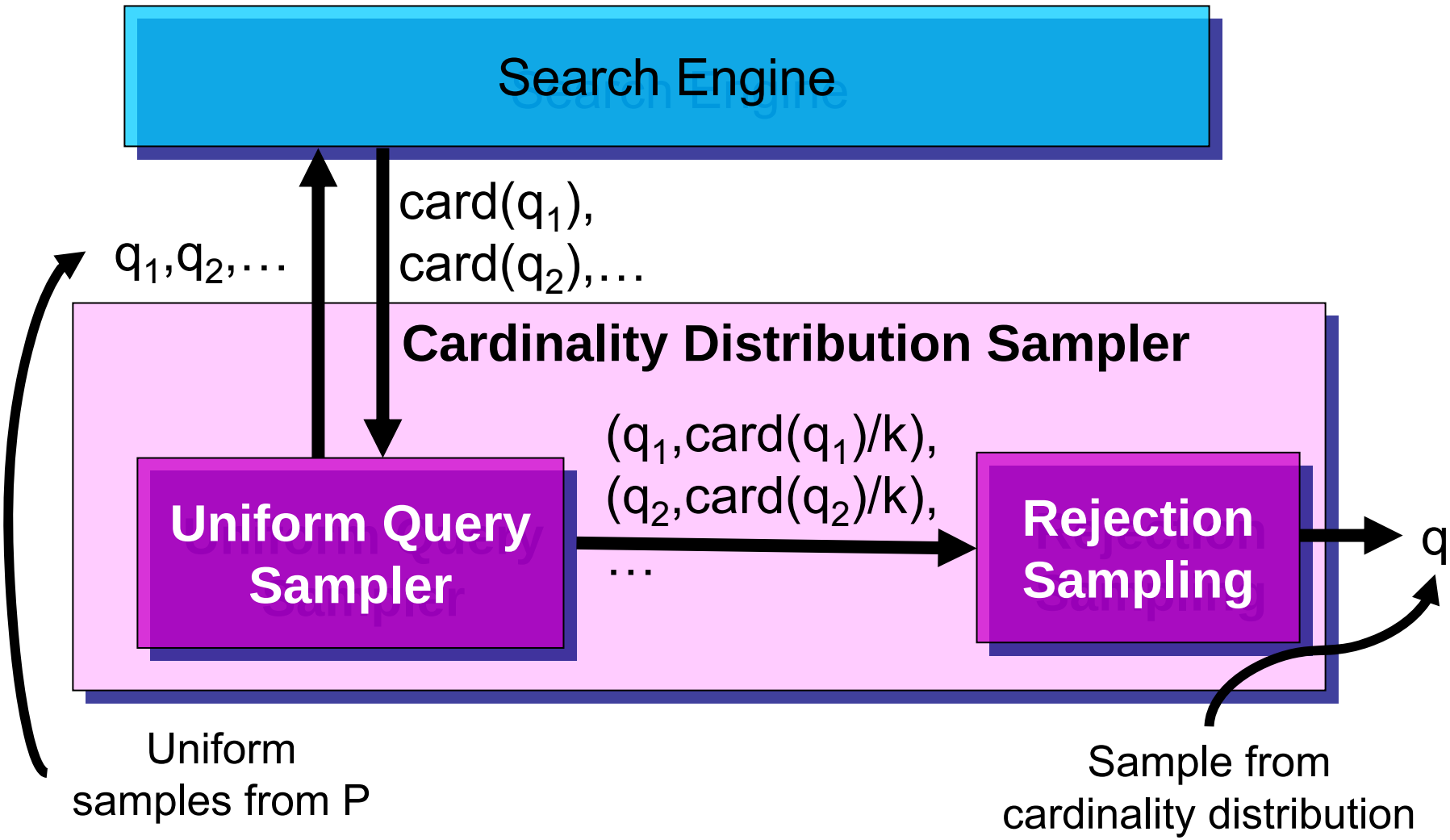
Unrealistic assumptions:

- Can sample queries from the cardinality distribution
  - In practice, don't know a priori card(q) for all q ∈ P
- ∀q ∈ P, 1 ≤ card(q) ≤ k
  - In practice, some queries underflow (card(q) = 0) or overflow (card(q) > k)

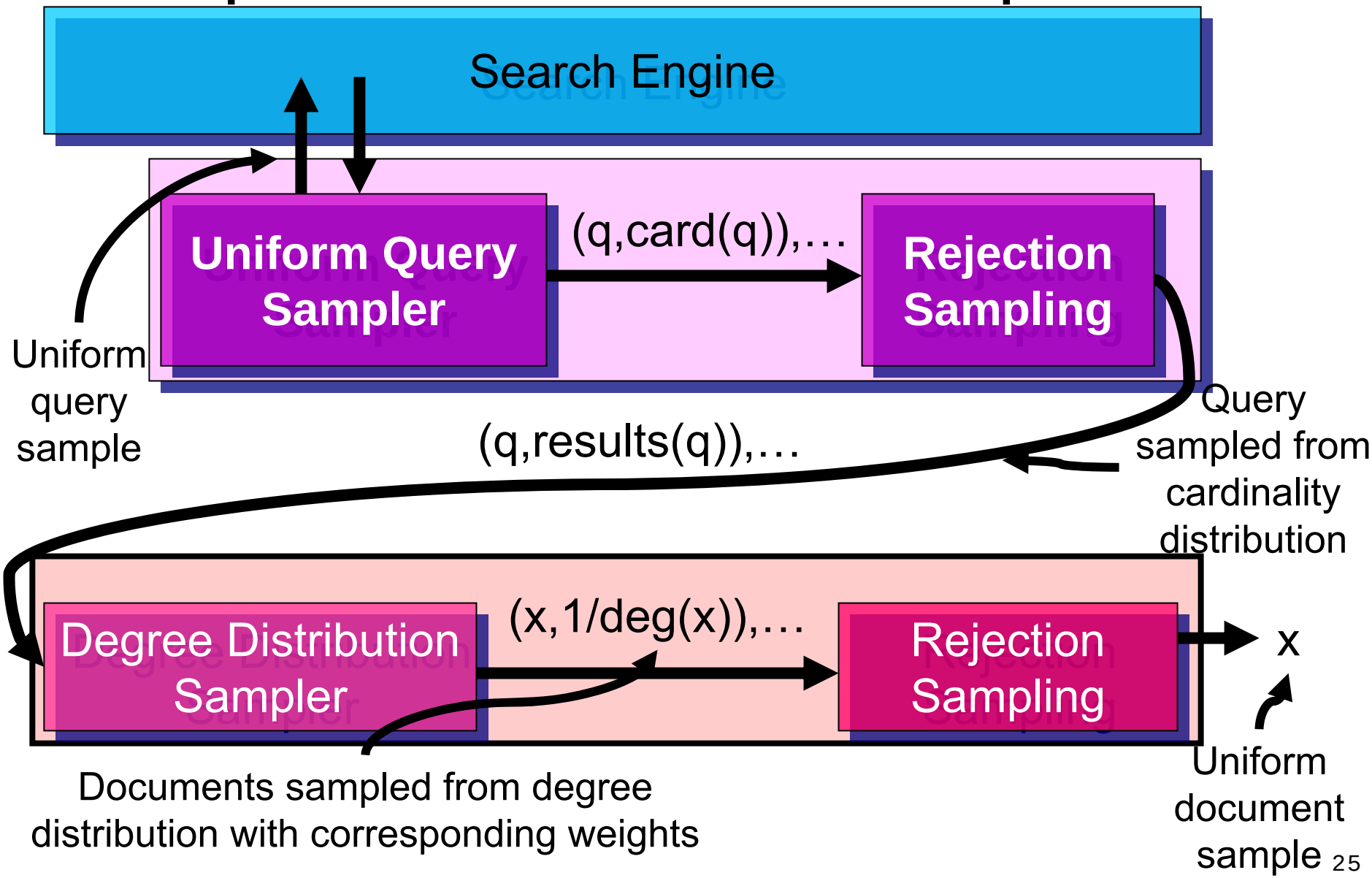
Degree Distribution Sampler



Cardinality Distribution Sampler



Complete Pool-Based Sampler



A random walk sampler

- Define a graph G over the indexed documents
  - (x,y) ∈ E iff results(x) ∩ results(y) ≠ ∅
  - $$w(x,y) = \sum_{q \in \text{results}(x) \cap \text{results}(y)} \frac{1}{\text{card(q)}}$$
- Run a random walk on G
  - Limit distribution = degree distribution
  - Use MCMC methods to make limit distribution uniform.
    - Metropolis-Hastings
    - Maximum-Degree
- Does not need a preprocessing step
- Less efficient than the pool-based sampler