600 JAHRE
UNIVERSITÄT LEIPZIG

Evaluation of Query Generators for Entity Search Engines

Stefan Endrullis, Andreas Thor, Erhard Rahm

University of Leipzig

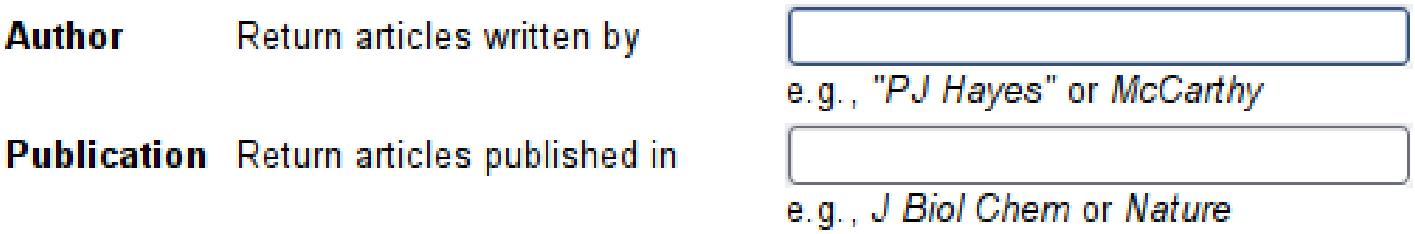
http://dbs.uni-leipzig.de

Aug. 24, 2009

Entity Search Engine (ESE)

Examples: Google Scholar, Amazon Advanced Search for DVDs

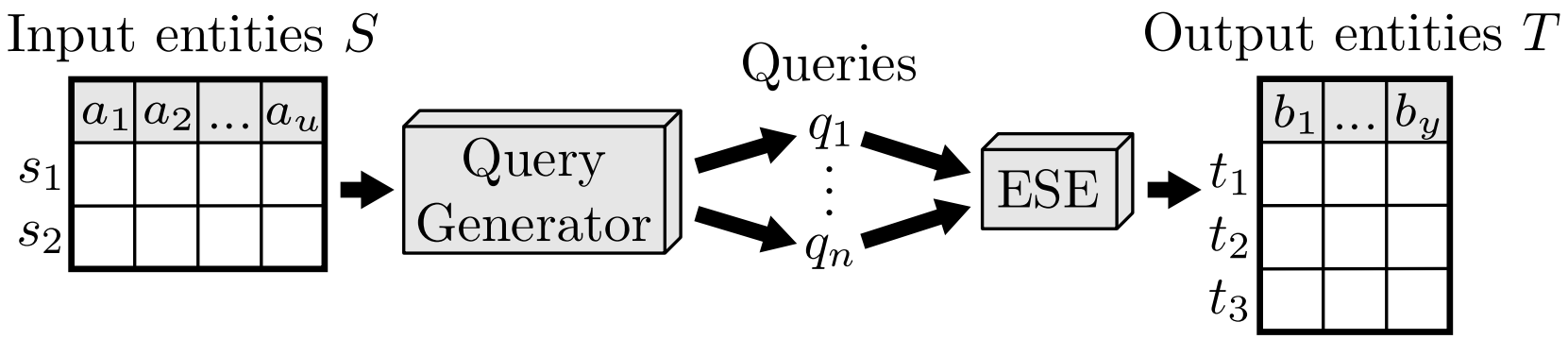
- domain specific search engine
- web interface for querying certain type of entities



- set of predicates $p_1, \ldots, p_m$
- corresponding search values $v_1, \ldots, v_m$
- query interpretation: $q \approx \bigwedge_{1 \leq i \leq m,\ v_i \neq \epsilon} p_i(v_i)$
- result: top-k entities

Motivation

Common data integration task: find specific set of entities at ESE



Relevant for

- offline data integration (e.g. data warehousing)
- online data integration (e.g. mashups)

Examples:

- list of products → find prices
- list of research papers of venue → enrich with citation counts

Outline

- Challenges
- Concept of Query Generators
- Evaluation of Query Generators
- Initial Evaluation for Bibliographic Domain
- Summary and Outlook

Challenges

Problem:

- find given set of entities at an entity search engine

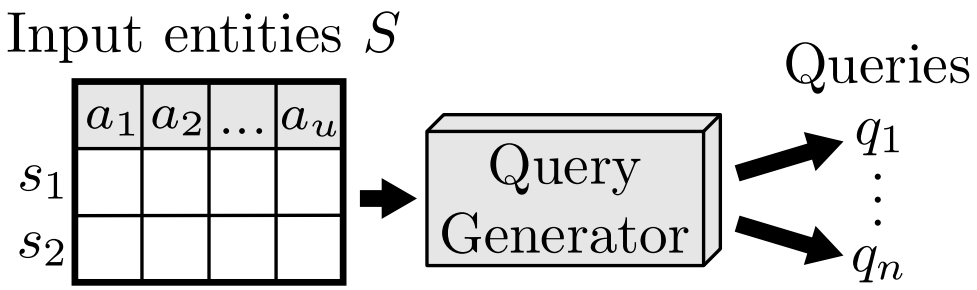
Challenges:

- how to find effective and efficient search queries?
- myriad of different queries (predicate combinations)
- data quality problems (misspelling, missing values, synonyms, ...)
- limited number of requests per time interval
- limited number of entities per result page

Concept of Query Generators

Query Generator:

- function that maps a set of entities (multi-valued relation) to a set of search queries



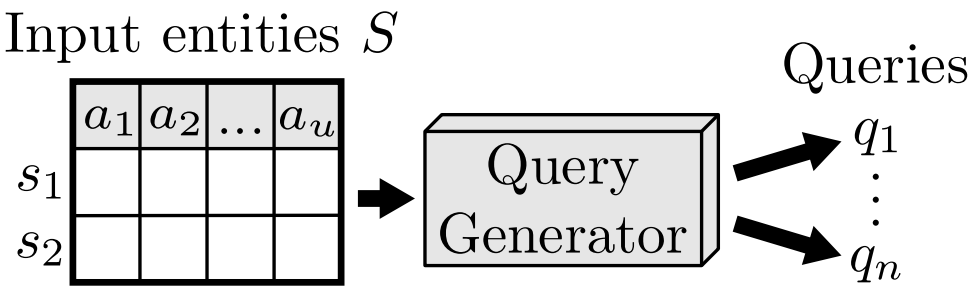
4 general properties based on generic data structures:

- attribute-predicate mapping
- search value generation
- partitioning: naive, frequent value
- aggregation: query combination

Concept of Query Generators

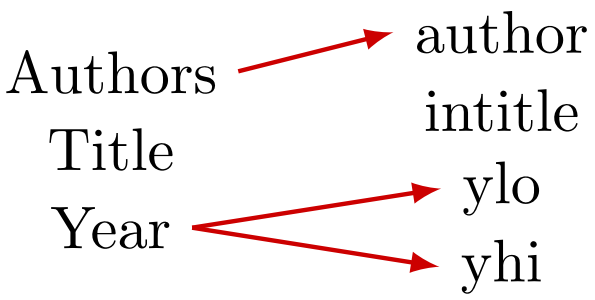
Query Generator:

- function that maps a set of entities (multi-valued relation) to a set of search queries



4 general properties based on generic data structures:

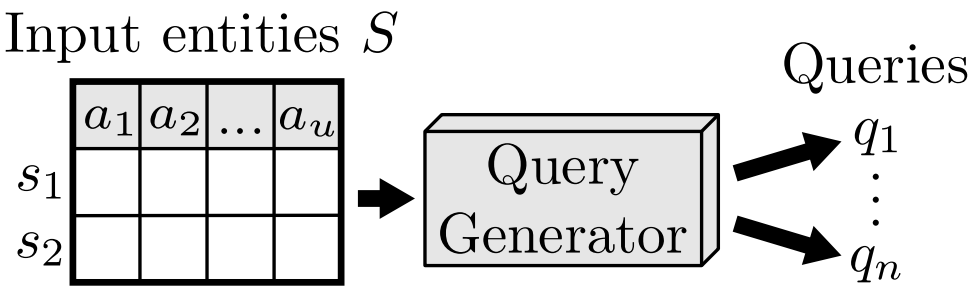
- attribute-predicate mapping
- search value generation
- partitioning: naive, frequent value
- aggregation: query combination



Concept of Query Generators

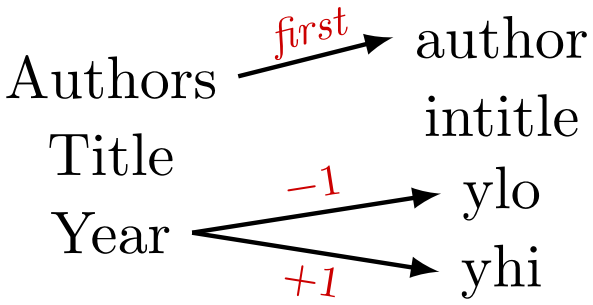
Query Generator:

- function that maps a set of entities (multi-valued relation) to a set of search queries



4 general properties based on generic data structures:

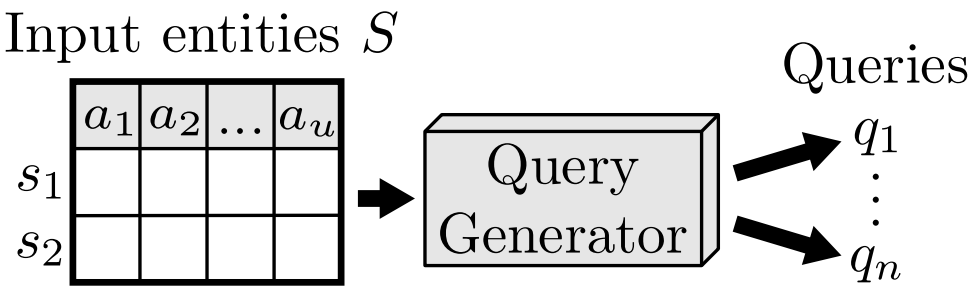
- attribute-predicate mapping
- search value generation
- partitioning: naive, frequent value
- aggregation: query combination



Concept of Query Generators

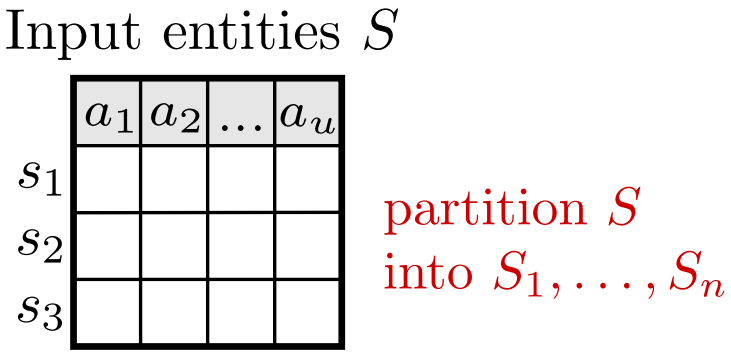
Query Generator:

- function that maps a set of entities (multi-valued relation) to a set of search queries



4 general properties based on generic data structures:

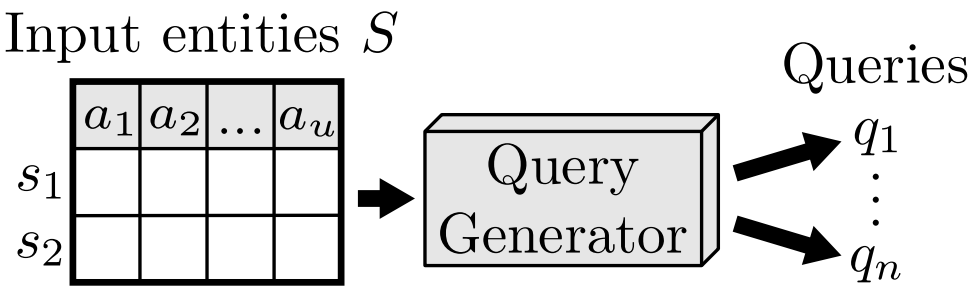
- attribute-predicate mapping
- search value generation
- partitioning: naive, frequent value
- aggregation: query combination



Concept of Query Generators

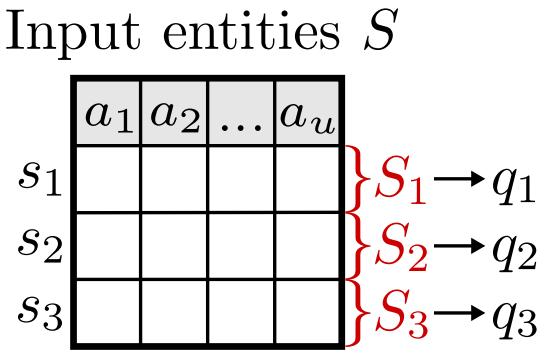
Query Generator:

- function that maps a set of entities (multi-valued relation) to a set of search queries



4 general properties based on generic data structures:

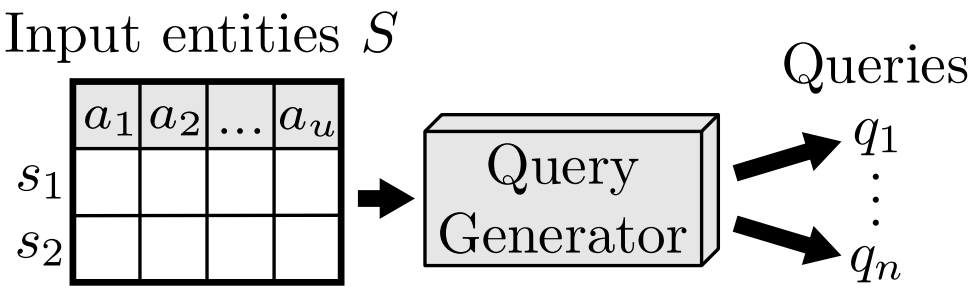
- attribute-predicate mapping
- search value generation
- partitioning: naive, frequent value
- aggregation: query combination



Concept of Query Generators

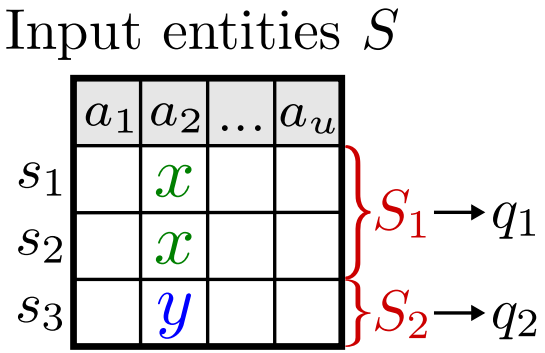
Query Generator:

- function that maps a set of entities (multi-valued relation) to a set of search queries



4 general properties based on generic data structures:

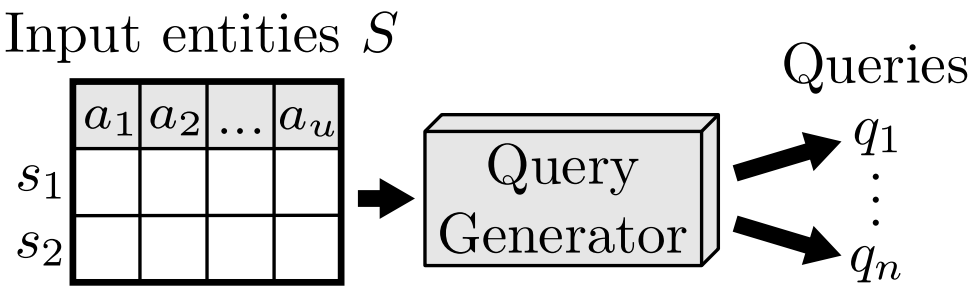
- attribute-predicate mapping
- search value generation
- partitioning: naive, frequent value
- aggregation: query combination



Concept of Query Generators

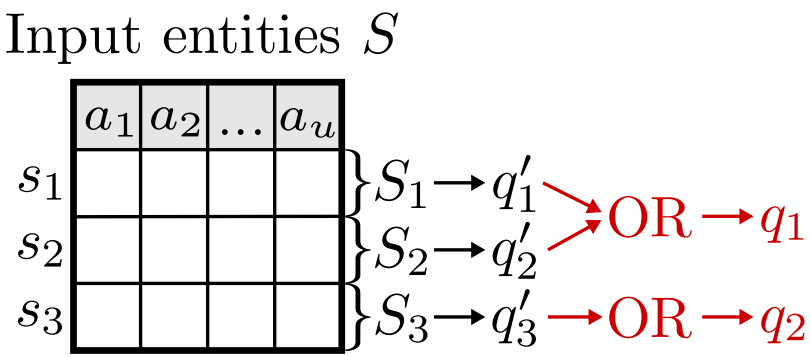
Query Generator:

- function that maps a set of entities (multi-valued relation) to a set of search queries



4 general properties based on generic data structures:

- attribute-predicate mapping
- search value generation
- partitioning: naive, frequent value
- aggregation: query combination



Naive Query Generator

Properties:

- one query per input entity
- OR aggregation to reduce number of queries
  - not applicable to any ESE

Example:

Authors	Title
{Smith, Jones}	The question to 42
{Williams, Smith}	Don't Panic

$$\begin{matrix} \text{Authors} & & \text{author} \\ \text{Title} & \xrightarrow{\textit{keywords}} & \text{intitle} \end{matrix}$$

Queries:

- $q_1 = \text{intitle(question 42)}$
- $q_2 = \text{intitle(panic)}$

Naive Query Generator

Properties:

- one query per input entity
- OR aggregation to reduce number of queries
  - not applicable to any ESE

Example:

Authors	Title
{Smith, Jones}	The question to 42
{Williams, Smith}	Don't Panic

$$\begin{matrix} \text{Authors} & & \text{author} \\ \text{Title} & \xrightarrow{\textit{keywords}} & \text{intitle} \end{matrix}$$

Queries:

- $q_1 = \text{intitle(question 42)}$
- $q_2 = \text{intitle(panic)}$

Naive Query Generator

Properties:

- one query per input entity
- OR aggregation to reduce number of queries
  - not applicable to any ESE

Example:

Authors	Title
{Smith, Jones}	The question to 42
{Williams, Smith}	Don't Panic

$$\begin{matrix}\text{Authors} & & \text{author} \\ \text{Title} & \xrightarrow{\textit{keywords}} & \text{intitle}\end{matrix}$$

Queries:

- $q_1 = \text{intitle(question 42)}$
- $q_2 = \text{intitle(panic)}$

Frequent Value Query Generator

Properties:

- input analysis → identify frequent item sets (FIS) on attribute values
- variation of Apriori algorithm
- partitioning based on FISs with largest coverage → reduce #queries
- applicable to any ESE

Example:

Authors	Title
{Smith, Jones}	The question to 42
{Williams, Smith}	Don't Panic

$$\begin{array}{ccc} \text{Authors} & \xrightarrow{id} & \text{author} \\ \text{Title} & & \text{intitle} \end{array}$$

Queries:

- $q_1 = \text{author}(\text{Smith})$

Frequent Value Query Generator

Properties:

- input analysis → identify frequent item sets (FIS) on attribute values
- variation of Apriori algorithm
- partitioning based on FISs with largest coverage → reduce #queries
- applicable to any ESE

Example:

Authors	Title
{Smith, Jones}	The question to 42
{Williams, Smith}	Don't Panic

$$\begin{matrix}\text{Authors} \\ \text{Title}\end{matrix} \xrightarrow{id} \begin{matrix}\text{author} \\ \text{intitle}\end{matrix}$$

Queries:

- $q_1 = \text{author}(\text{Smith})$

Frequent Value Query Generator

Properties:

- input analysis → identify frequent item sets (FIS) on attribute values
- variation of Apriori algorithm
- partitioning based on FISs with largest coverage → reduce #queries
- applicable to any ESE

Example:

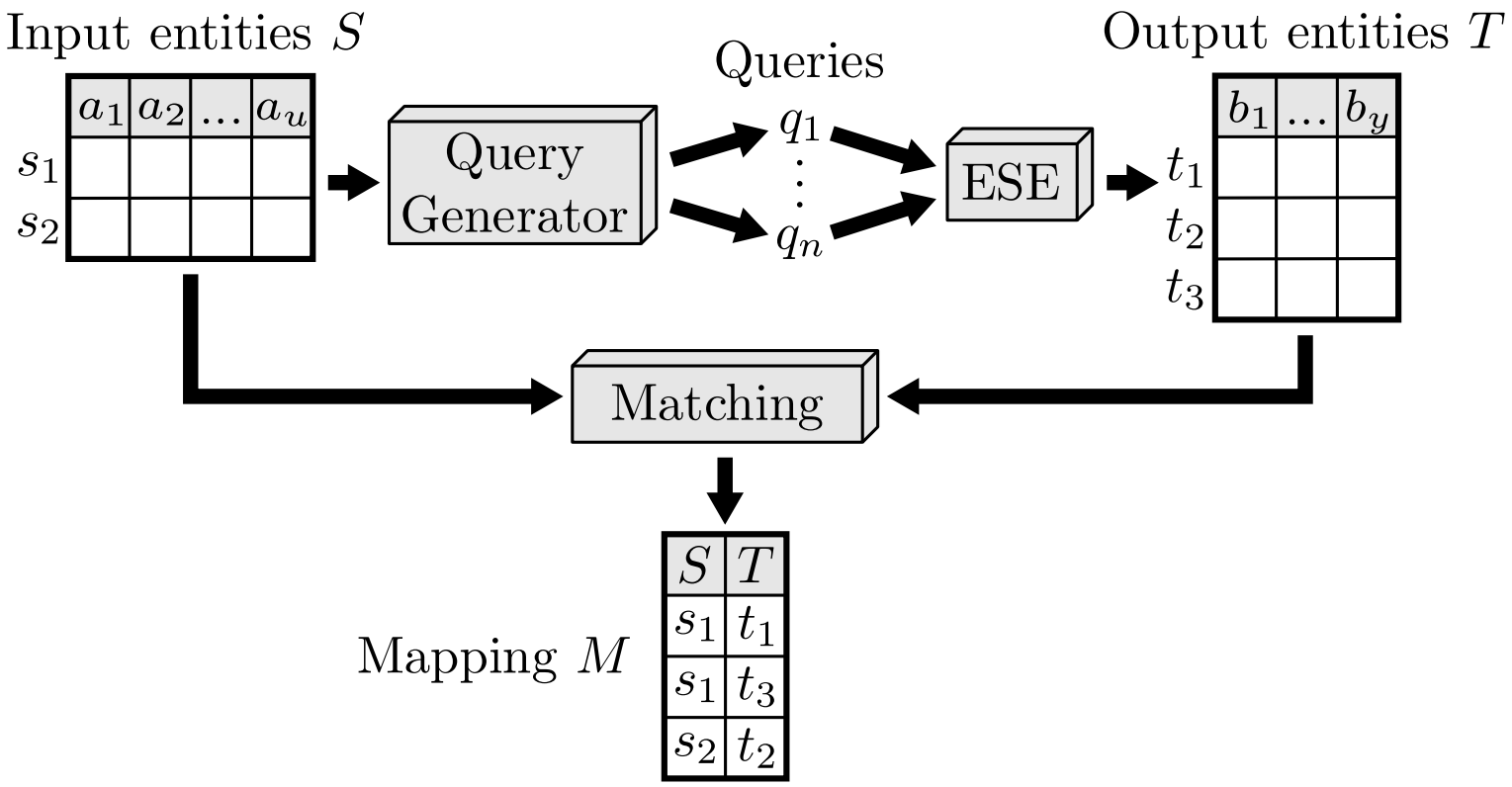
Authors	Title
{Smith, Jones}	The question to 42
{Williams, Smith}	Don't Panic

$$\begin{array}{ccc} \text{Authors} & \xrightarrow{id} & \text{author} \\ \text{Title} & & \text{intitle} \end{array}$$

Queries:

- $q_1 = \text{author}(\text{Smith})$

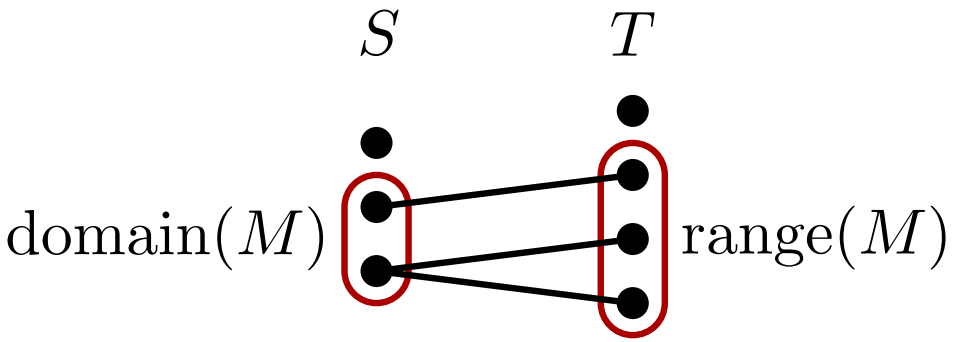
Evaluation of Query Generators



- automatic evaluation of several query generators for different datasets
  - matching between S and T → identify same real world objects
  - query statistics (S, T, M, #requests) stored in data warehouse to perform evaluation afterwards

Evaluation Measures

Mapping example:



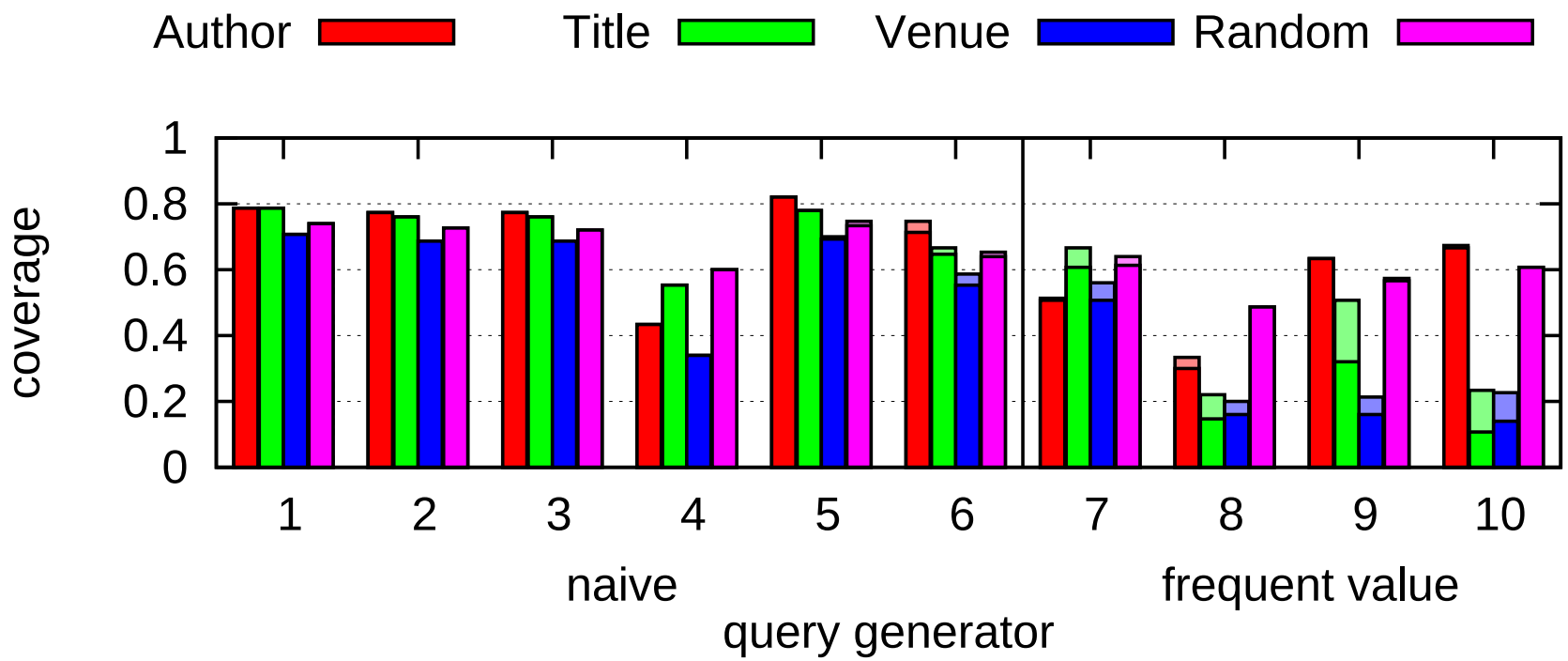
Evaluation measures:

$$\begin{aligned}
\textit{precision} &:= \frac{|\text{range}(M)|}{|T|} &&= \frac{3}{4} \\
\textit{coverage} &:= \frac{|\text{domain}(M)|}{|S|} &&= \frac{2}{3} \\
\textit{efficiency} &:= \frac{|\text{domain}(M)|}{\textit{requests}} &&= 2
\end{aligned}$$

Initial Evaluation for Bibliographic Domain

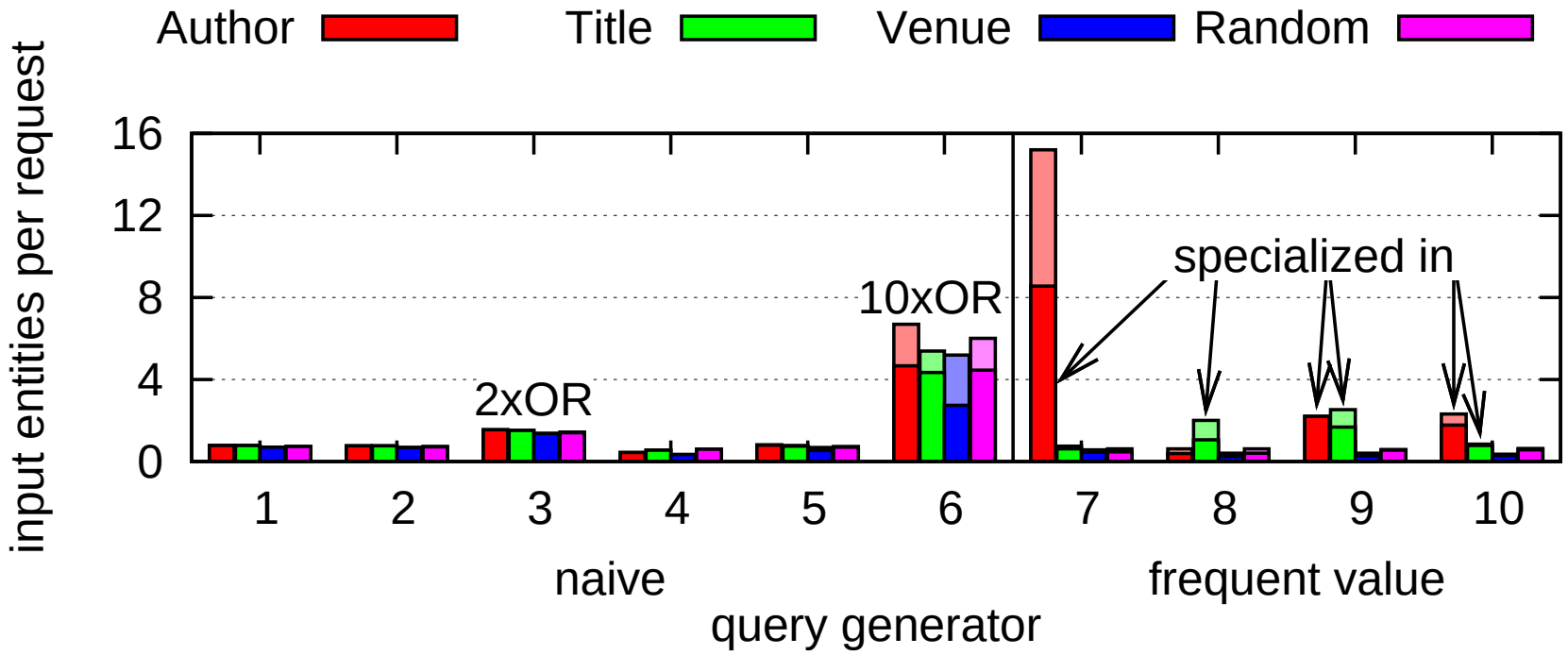
- ESE: Google Scholar, at most 300 results (3×100)
- 10 query generators
- 60 ($= 4 \cdot 3 \cdot 5$) datasets based on DBLP:
  - 4 categories (Author, Title, Venue, Random)
  - 3 dataset sizes (5, 30, 100)
  - 5 different datasets for each combination

Initial Evaluation - Coverage



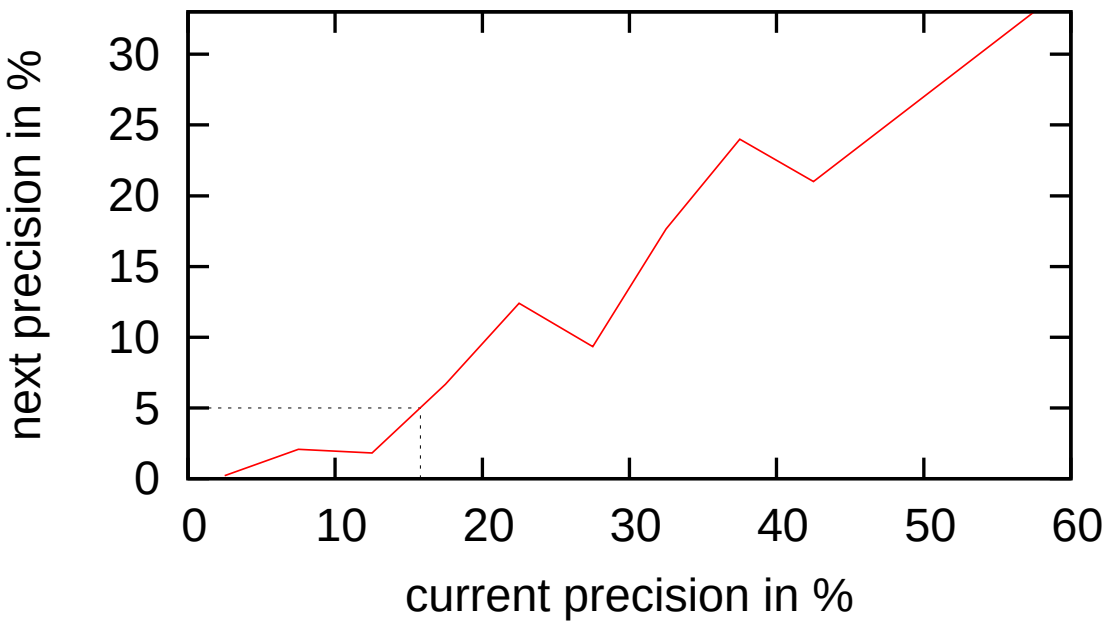
- naive QGs:
  - better coverage than frequent value QGs (except #4 which is too precise)
- frequent value QGs:
  - partially high coverage variations between different categories

Initial Evaluation - Efficiency



- aggregation raises efficiency
- frequent value QGs:
  - efficiency strongly depends on dataset characteristics (category)

Estimated Precision for the next Request



- raise efficiency by following the next link depending on its estimated precision (estimation based on current precision)

Example:

- aim: next link precision $\geq 5\%$
- conclusion: follow next link only iff current precision $\geq 15.8\%$

Summary and Future Work

Summary:

- approach for flexible query generators
- approach for automatic evaluation of query generators
- initial evaluation for bibliographic domain

Future Work:

- adaptive search strategies to automatically select most efficient query generator (combination)
- query generators for additional domains and ESEs