Promise Winter School
Bridging between Information Retrieval and Databases

Bressanone, Italy 4 - 8 February 2013

The Keymantic and Keyry approaches for querying relational databases

Francesco Guerra

Dipartimento di Ingegneria Enzo Ferrari
University of Modena and Reggio Emilia, Italy

Keyword based searching over RDBs – recap

- Keyword-based searching is **an attractive alternative** to traditional SQL queries

- The challenges are
  - to discover the database structures that contain the keywords
  - to explore how these structures are inter-connected to form an answer

- The discovered structures and their inter-connections represent in relational terms the semantic interpretation of the keyword query

- Existing techniques typically suffer from two main limitations:
  1. They are **based on indexes** on the data
  2. No enough attention has been paid to the **inter-dependencies among the keywords**

Keymantic

Keymantic

- We designed and implemented a keyword-based search engine that **does not rely on the knowledge of the database instance**
  - Keyword queries are translated into SQL queries

- Keymantic approach is based on
  - **weights** measuring the likelihood that keywords are mapped into database terms
  - an **extension** of the **Hungarian algorithm** for computing ranked mappings of keywords and database terms

- Reference papers
  - S. Bergamaschi, E. Domnori, F.Guerra, R. Trillo Lado, Y. Velegrakis: Keyword search over relational databases: a metadata approach. SIGMOD Conference 2011: 565-576
  - S. Bergamaschi, E. Domnori, F. Guerra, M. Orsini, R. Trillo Lado, Y. Velegrakis: Keymantic: Semantic Keyword-based Searching in Data Integration Systems. PVLDB 3(2): 1637-1640 (2010)

Motivating example

Person

Name	Area	Phone	Address	Email
Watson	Database	(320) 4631234	30 Bloor	watson@aaa.bb
Lenzerini	Database	(390) 6987654	Ariosto 25	lenzerini@bbb.cc
Date	Database	(817) 1937842	107 GACB	date@ccc.dd
Hunt	Inf. Systems	(343) 2920812	17 Helix	Hunt@ddd.ee

Affiliated

Professor	Department
Watson	x123
Lenzerini	cs34
Date	cs34
Hunt	m111

Department

id	DName	Address	Director
x123	CS	25 Blicker	Watson
cs34	IE	15 Tribeca	Hunt
ee67	EE	5 Charles	Date
m111	ME	2 Cottle	Hunt

Author

Name	Publication
Lenzerini	Data Integration
Date	Foundation Matters

Publication

Title	Year	Resource
Data Integration	2002	ACM DL
Foundation Matters	2002	DBLP

Database

Name	Address
DBLP	http://www.informatik.uni-trier.de
ACM DL	http://portal.acm.org/dl.cfm

- The first problem is to decide **what** role each keyword plays in the query
  - Is it a value?
  - Does it describe some meta-information?

- Let "`Date Database`" be a keyword query posed over this database

- Both keywords are values:
  - `Date` → dom(Person.Name) and `Database` → dom(Person.Area)
- One keyword is a value, the other is meta-information
  - Date → Author.Name and Database → Database

Motivating example (2)

Person

Name	Area	Phone	Address	Email
Watson	Database	(320) 4631234	30 Bloor	watson@aaa.bb
Lenzerini	Database	(390) 6987654	Ariosto 25	lenzerini@bbb.cc
Date	Database	(817) 1937842	107 GACB	date@ccc.dd
Hunt	Inf. Systems	(343) 2920812	17 Helix	Hunt@ddd.ee

Affiliated

Professor	Department
Watson	x123
Lenzerini	cs34
Date	cs34
Hunt	m111

Department

id	DName	Address	Director
x123	CS	25 Blicker	Watson
cs34	IE	15 Tribeca	Hunt
ee67	EE	5 Charles	Date
m111	ME	2 Cottle	Hunt

Author

Name	Publication
Lenzerini	Data Integration
Date	Foundation Matters

Publication

Title	Year	Resource
Data Integration	2002	ACM DL
Foundation Matters	2002	DBLP

Database

Name	Address
DBLP	http://www.informatik.uni-trier.de
ACM DL	http://portal.acm.org/dl.cfm

- The second problem is to decide **which** part of the database actually models the intended keyword meaning
  - *Configuration*: an injective mapping from the keywords into the database terms, i.e. relation and attributes names, attribute domains
- Consider the keyword query “`Director Watson Address`”

`Director Watson Address`

1. `Director` → Department.Director `Watson` → dom(Department.Director) `Address` → Department.Address
2. `Director` → Department.Director `Watson` → dom(Department.Director) `Address` → Person.Address
3. `Director` → Department.Director `Watson` → dom(Department.Address) `Address` → Person.Address

Motivating example (3)

Person

Name	Area	Phone	Address	Email
Watson	Database	(320) 4631234	30 Bloor	watson@aaa.bb
Lenzerini	Database	(390) 6987654	Ariosto 25	lenzerini@bbb.cc
Date	Database	(817) 1937842	107 GACB	date@ccc.dd
Hunt	Inf. Systems	(343) 2920812	17 Helix	Hunt@ddd.ee

Affiliated

Professor	Department
Watson	x123
Lenzerini	cs34
Date	cs34
Hunt	m111

Department

id	DName	Address	Director
x123	CS	25 Blicker	Watson
cs34	IE	15 Tribeca	Hunt
ee67	EE	5 Charles	Date
m111	ME	2 Cottle	Hunt

Author

Name	Publication
Lenzerini	Data Integration
Date	Foundation Matters

Publication

Title	Year	Resource
Data Integration	2002	ACM DL
Foundation Matters	2002	DBLP

Database

Name	Address
DBLP	http://www.informatik.uni-trier.de
ACM DL	http://portal.acm.org/dl.cfm

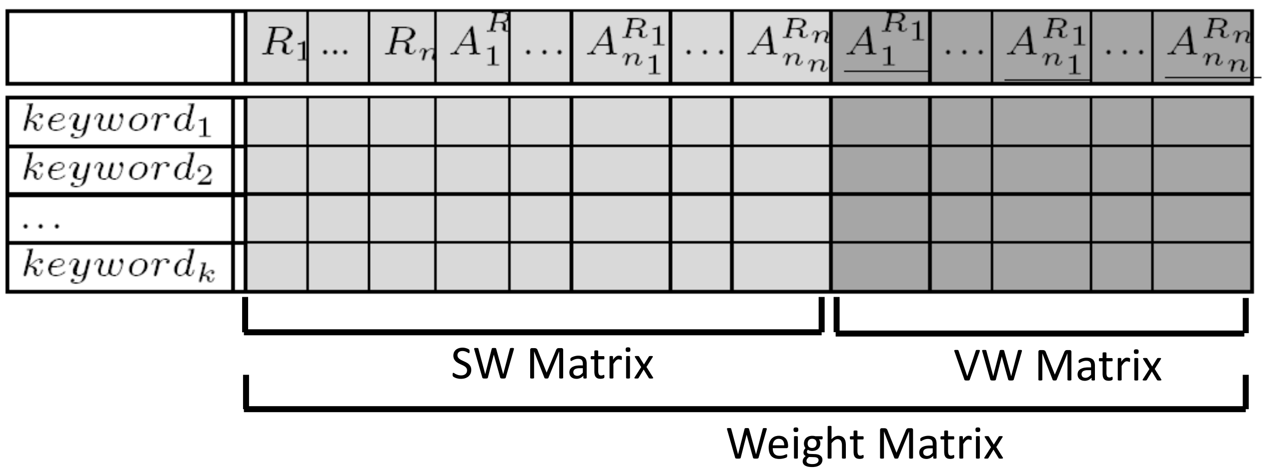
- Answering a query requires to decide **how** the selected database terms relate to each other
  - Two database terms may be connected by multiple join paths, thus leading to different interpretations
- *Interpretation* of a keyword query using a configuration is an SQL query where the select, from, where clauses are formulated with the configuration

`Email CS`

1. Department, Person
   - to find the email of the CS department Director
2. Department, Affiliated, Person
   - to find emails of the CS affiliated persons

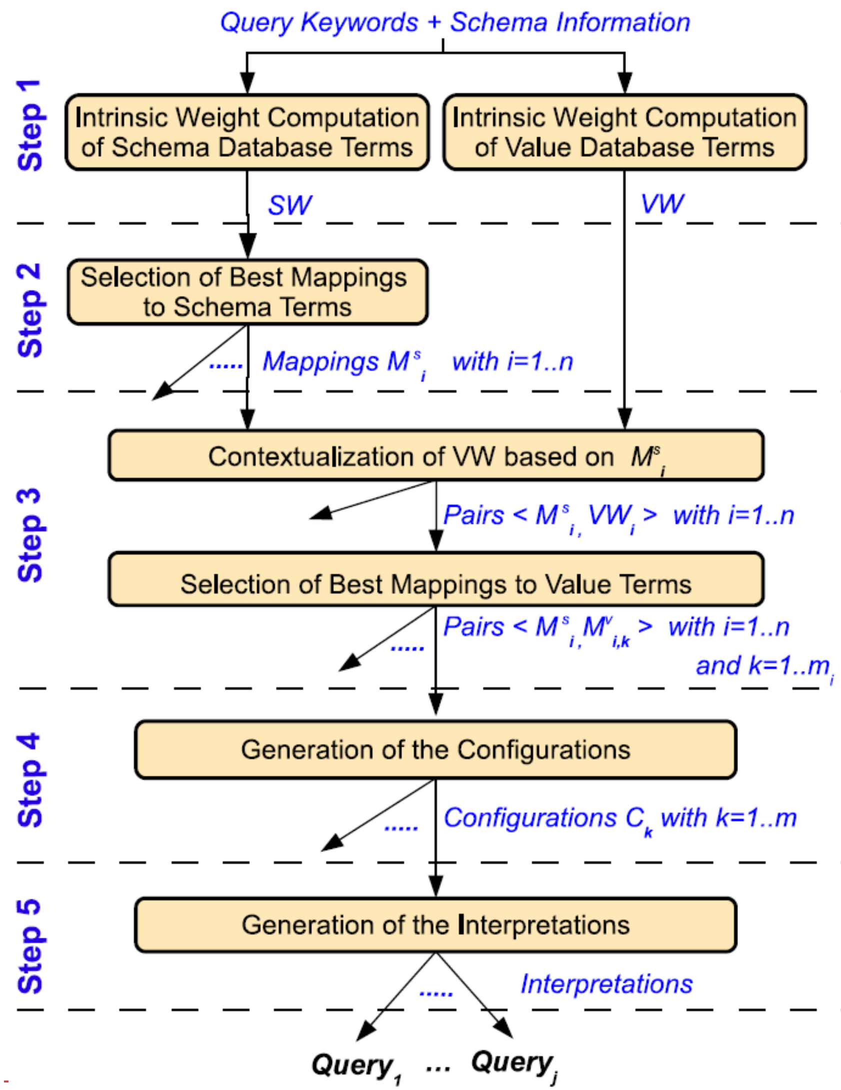
From Keywords to Queries

- Weights may measure the relativeness of a keyword to a database term:

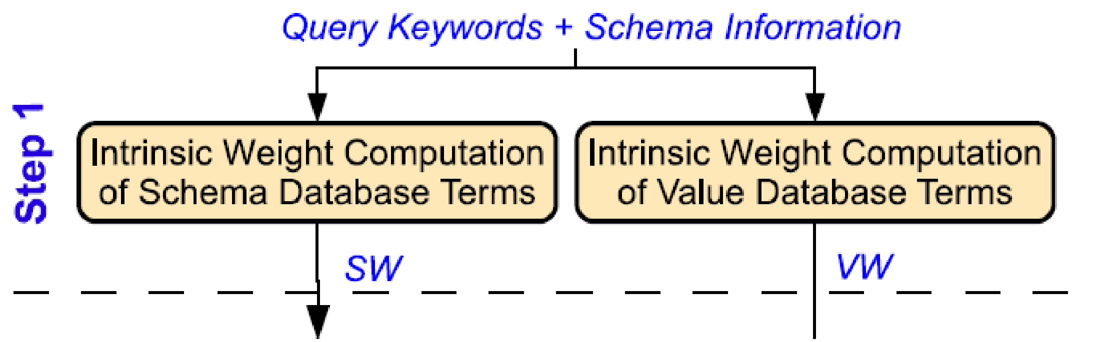


- The problem is known as Bipartite Weighted Assignements, but usual solutions
  1. do not consider any interdependencies between the partial associations
     - **Intrinsic**: measures the likelihood that a keyword should be mapped into a database term in isolation
     - **Contextual**: measures the relativeness of a keyword to a database term by taking into account the mappings of other keywords into database terms
  2. provide only the best mapping, instead of a ranked list based on the scores
     - we have extended the Hungarian algorithm

From keywords to queries, the process



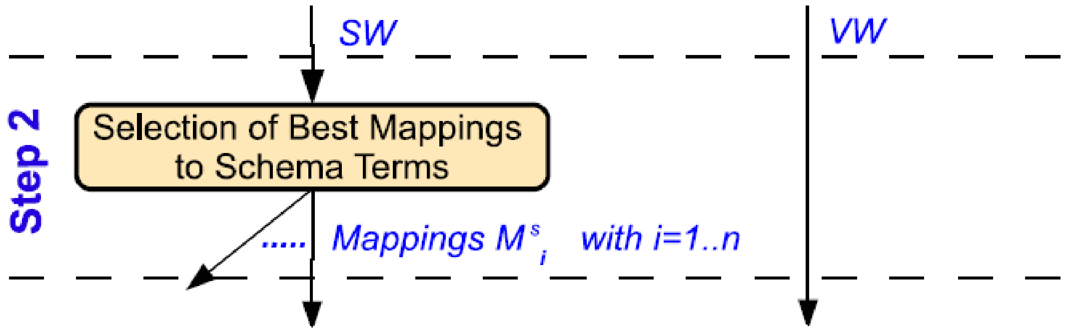
Step 1 – Intrinsic Weight Computation



- We exploit similarity techniques based on structural and lexical knowledge
  - extracted from the data source,
  - based on external knowledge, e.g., ontologies, vocabularies, domain, …
- The techniques include:

	Schema Weights	Value Weights
Syntactic techniques	similarity measures based on edit distances, …	Regular expressions
Semantic techniques	Lexical analysis	Data types, google similarity, …

Step 2
Selection of the Best Mappings to schema terms



- We consider first the prominent mappings of keywords to schema terms.
  - A series of mappings , M_1^S , M_2^S ,…, M_n^S, of keywords to schema terms are generated
  - The mappings are partial, i.e., not all the keywords are mapped to some schema term
  - The unmapped keywords are considered for mapping to value database terms.

How to find configurations: the extended Hungarian algorithm

- The execution of the algorithm consists of a series of iterative steps that generate a mapping with a maximum score.
- In our approach
  - Once a keyword is associated to a database term, the weight in the matrix are modified in order to take into account the mapping
  - Once a complete configuration is computed
    - the weight matrix is modified to exclude the computation of the same mapping again
    - the process continues to compute the mapping with the second largest score, etc.

How to find configurations: the extended Hungarian algorithm

	P.N	P.A	P.P	P.Ad	P.E	D.I	D.D	D.A	D.Di
CS	30	18	0	18	0	50	75	50	50
Hopkins	45	30	0	35	10	32	40	35	39
Mary	46	20	0	5	7	20	25	30	40
Summerhill	10	7	0	34	22	20	10	32	15

- The maximum weight of each row is first identified and characterized as *maximum*
  - If the maximum weights are all located in different columns, then a mapping is generated
  - if there is a column containing more than one weight characterized as maximum, all maximums in the column except the one with the maximum value loose their characterization as maximum.

How to find configurations: the extended Hungarian algorithm

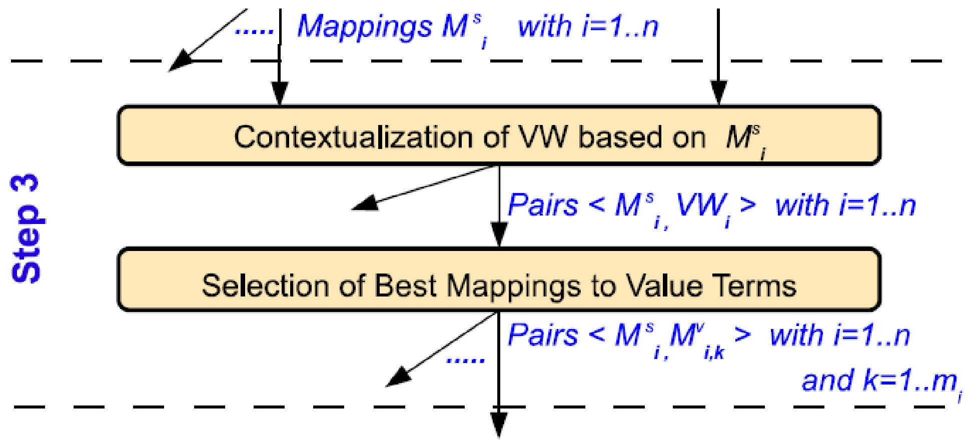
	P.N	P.A	P.P	P.Ad	P.E	D.I	D.D	D.A	D.Di
CS	30	18	0	18	0	50	75	50	50
$Hopkins$	45	30	0	35	10	32	40	35	39
$Mary$	46	20	0	5	7	20	25	30	40
$Summerhill$	10	7	0	34	22	20	10	32	15

- The values of the weights in these rows are then updated according to a number of contextual weights

	P.N	P.A	P.P	P.Ad	P.E	D.I	D.D	D.A	D.Di
CS	30	18	0	18	0	50	75	50	50
$Hopkins$	~~49~~	34	0	39	14	34	~~42~~	37	41
$Mary$	50	24	0	9	11	22	27	32	42
$Summerhill$	14	11	0	38	26	22	12	34	17

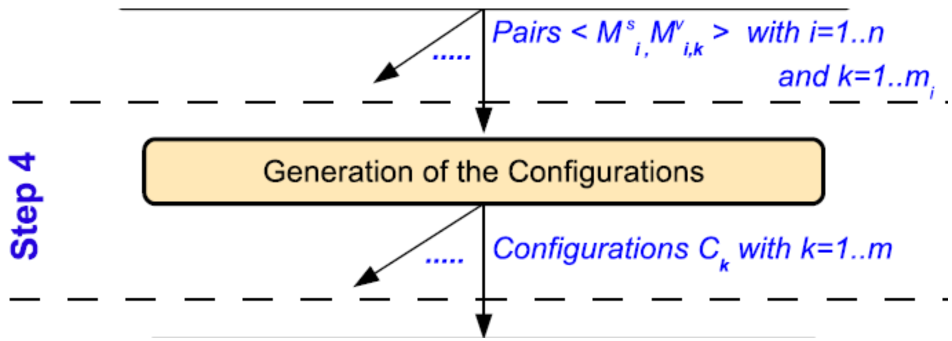
	P.N	P.A	P.P	P.Ad	P.E	D.I	D.D	D.A	D.Di
CS	30	18	0	18	0	50	75	50	50
$Hopkins$	49	34	0	39	14	34	42	37	41
$Mary$	50	24	0	9	11	22	27	32	42
$Summerhill$	14	11	0	38	26	22	12	34	17

Step 3
Selection of the Best Mappings to Value terms



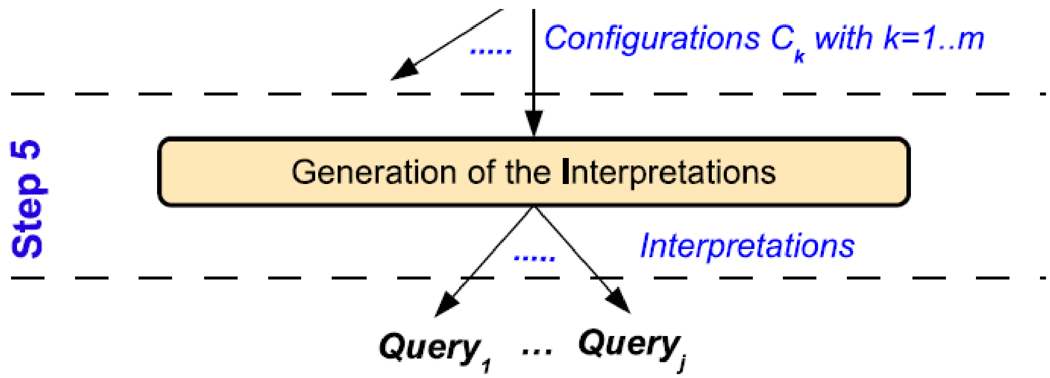
- For each partial mapping M_i^S, the mappings of the remaining unmapped keywords to value terms needs to be decided.
  1. **Contextualization of the VW sub-matrix:** the VW submatrix is updated to reflect the added value provided by the mappings in M_i^S
  2. **Selection of the Best Mappings** by using the adapted Hungarian algorithm
- The result is a series of partial mappings $M_{i,k}^V$

Step 4 – Generation of the Configurations



- A configuration C_{ik} is formed for each pair of a mapping $M^{V}_{i,k}$ together with its associated mapping M^{S}_{i}
- The score of the configuration is the sum of the scores of the two mappings

Step 5 – Generation of the interpretations



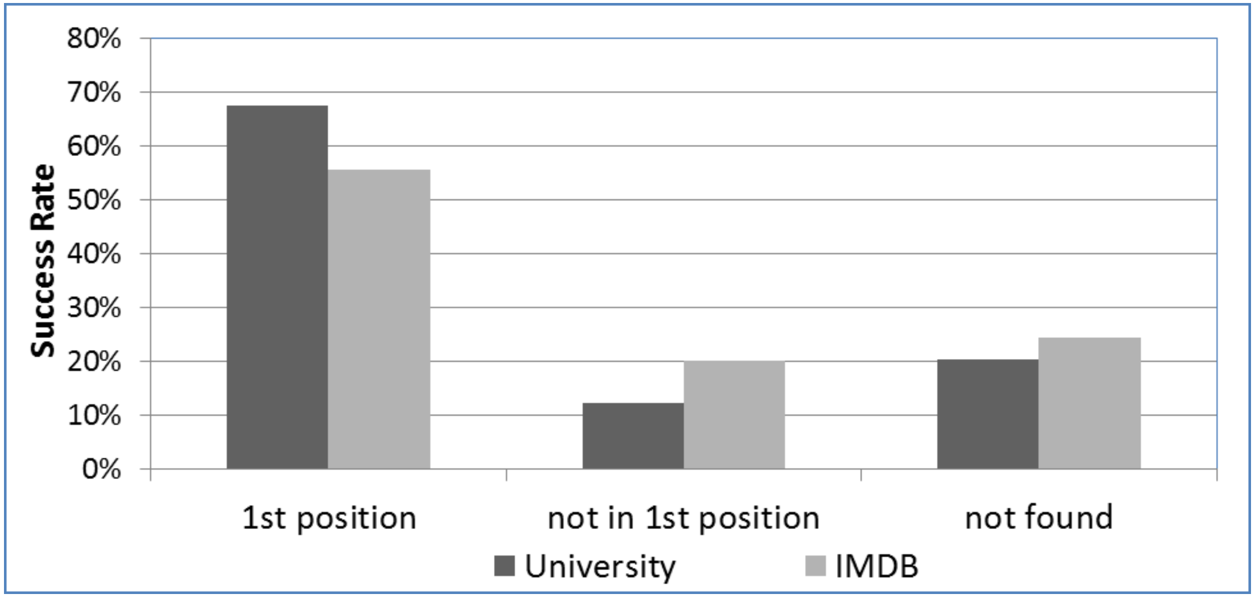
- Different join paths among these terms results in multiple interpretations
  - Several strategies can be used to further rank the selections
  - Length of the join path, …

- We compute a query for every alternative join path

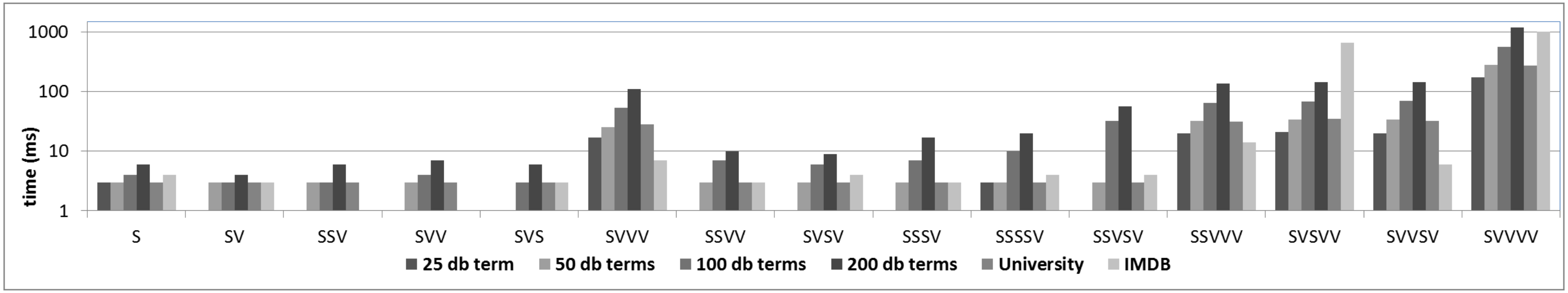
Evaluation

- We selected for our experiments two real data sets.
  - a university database
  - a fraction of the IMDB database
- 29 real users were asked to provide a set of keyword queries
- A database expert translated each keyword query into SQL.
- We used a total of 99 and 44 queries for the university and the IMDB database, respectively.

Evaluation - Effectiveness



Evaluation - Efficiency



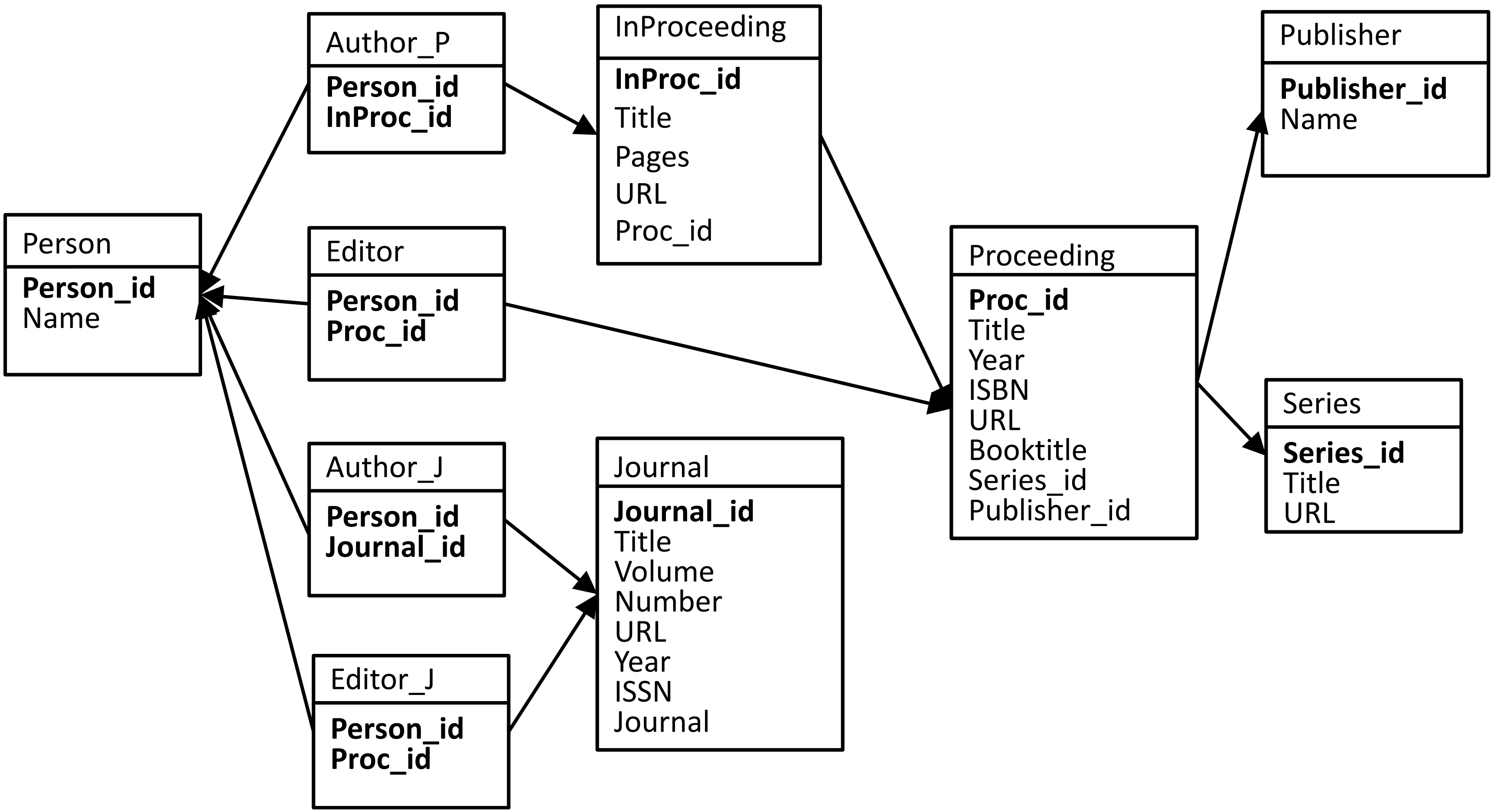
- The response time increases with the number of keywords
  - when there is a prevalence of keywords mapped to schema terms this increase is not dramatic.
- We did not report the time needed to actually evaluate the interpretations to avoid having the query engine performance blurring the results

Keyry

Our approach

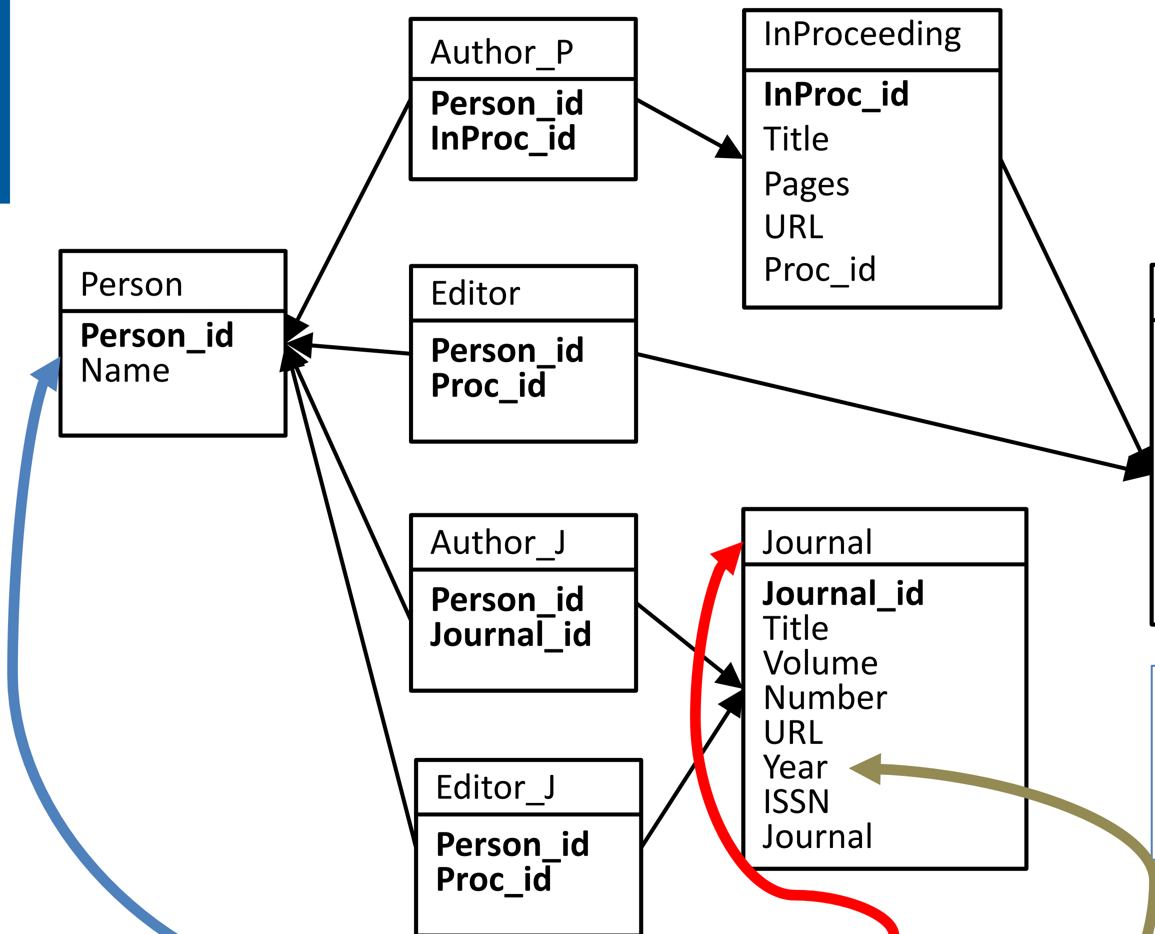
- KEYRY is based on
  - a Hidden Markov Model for mapping the user keywords into database terms
  - a method for providing a parameter setting not relying on any training data
    - **heuristics** rules, **similarity** measures and a variation of the **HITS algorithm**
- Reference papers
  - Sonia Bergamaschi, Francesco Guerra, Silvia Rota, Yannis Velegrakis: A Hidden Markov Model Approach to Keyword-Based Search over Relational Databases. ER 2011: 411-420S.
  - Silvia Rota, Sonia Bergamaschi, Francesco Guerra: The list Viterbi training algorithm and its application to keyword search over databases. CIKM 2011: 1601-1606

Motivating example



- User query «`Garcia-Molina Journal 2011`»

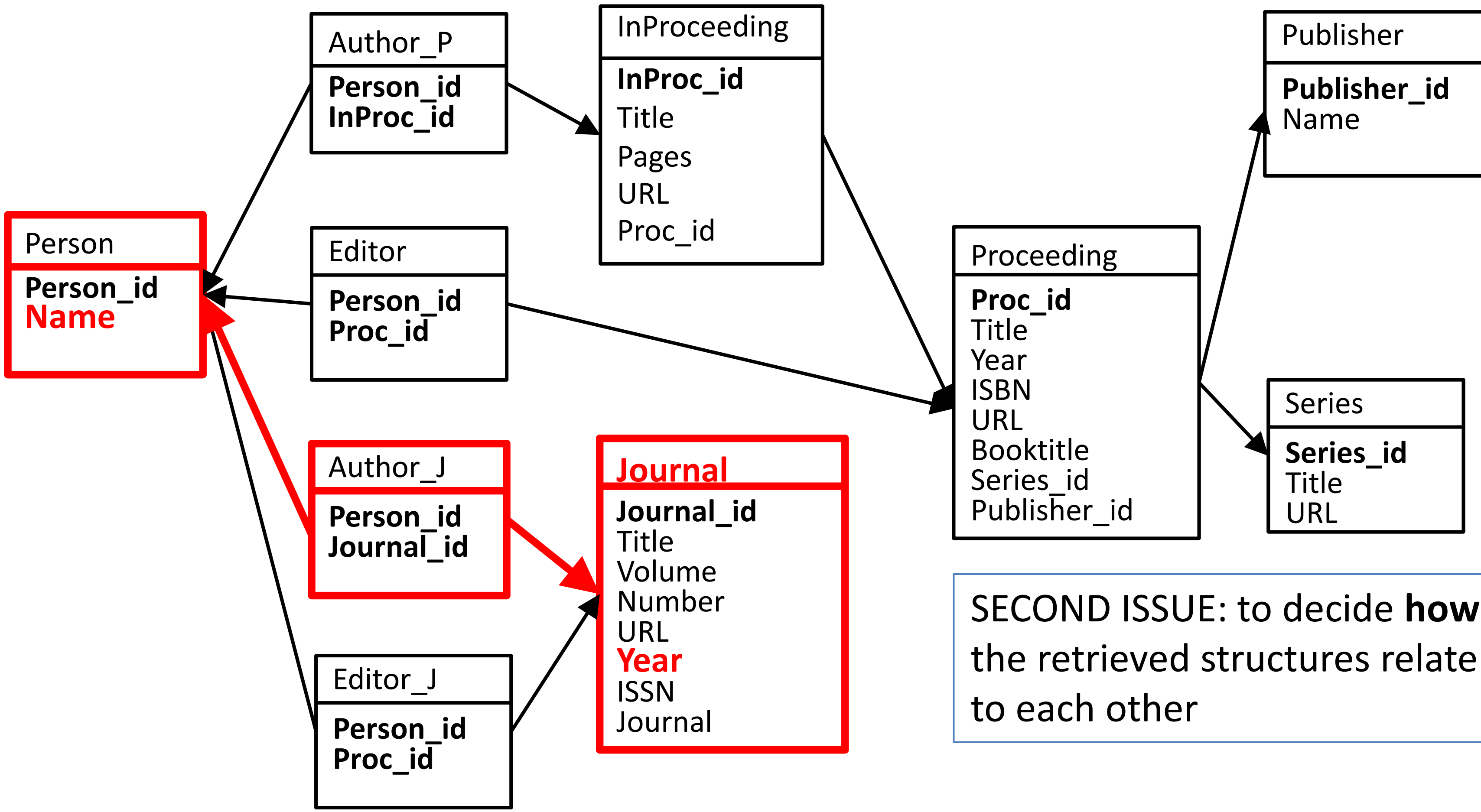
Motivating example



FIRST ISSUE: to decide **which** part of the database models the intended keyword meaning

- User query <Garcia-Molina Journal 2011>

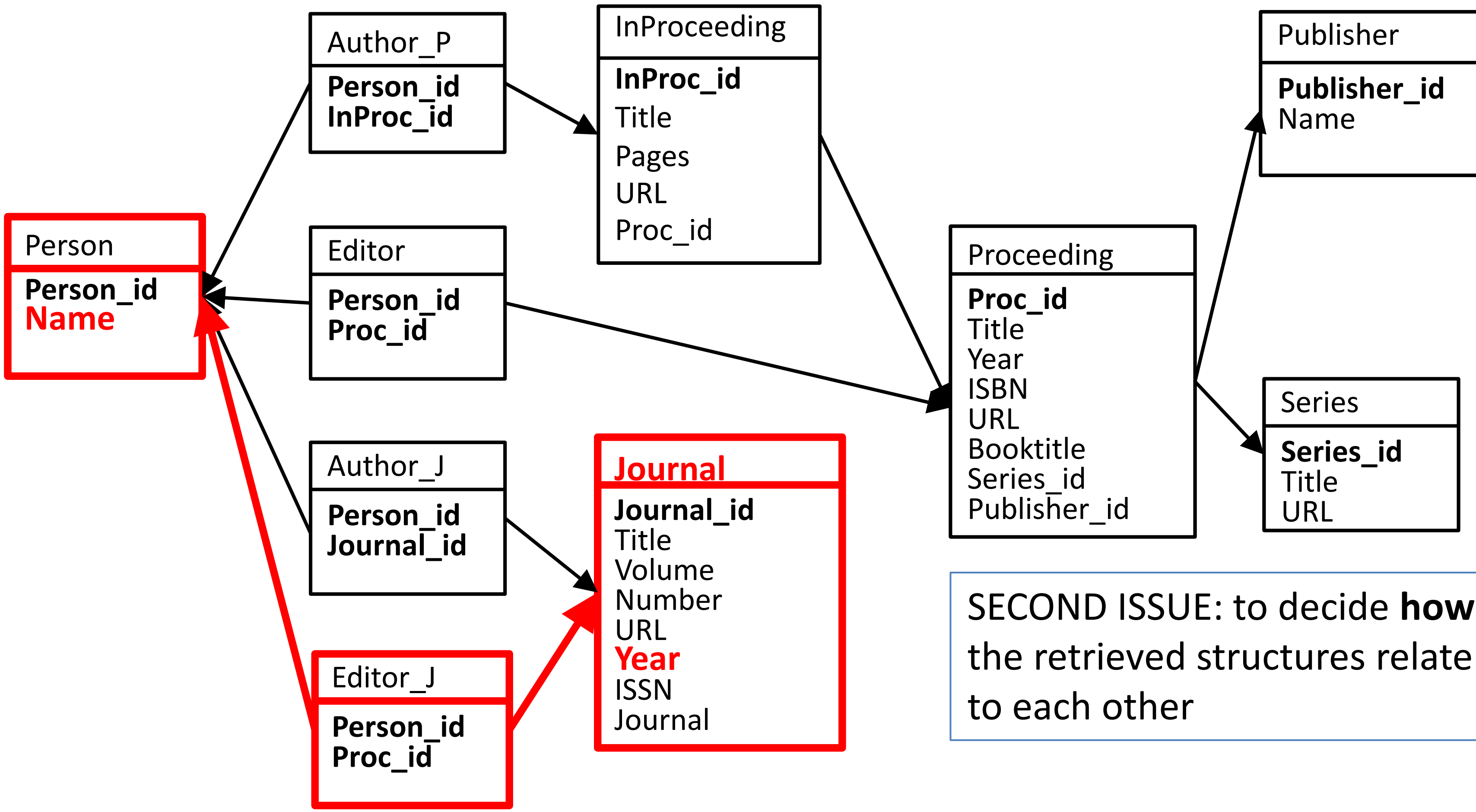
Motivating example



SECOND ISSUE: to decide **how** the retrieved structures relate to each other

- User query «`Garcia-Molina Journal 2011`»
  - Journals where Garcia Molina was an author in 2011

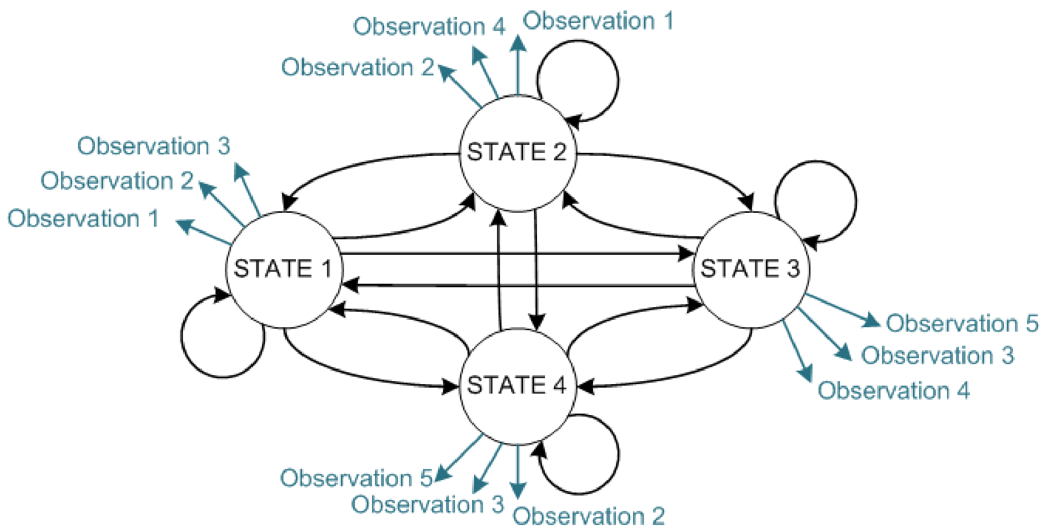
Motivating example



SECOND ISSUE: to decide **how** the retrieved structures relate to each other

- User query «`Garcia-Molina Journal 2011`»
  - Journals where Garcia Molina was an editor in 2011

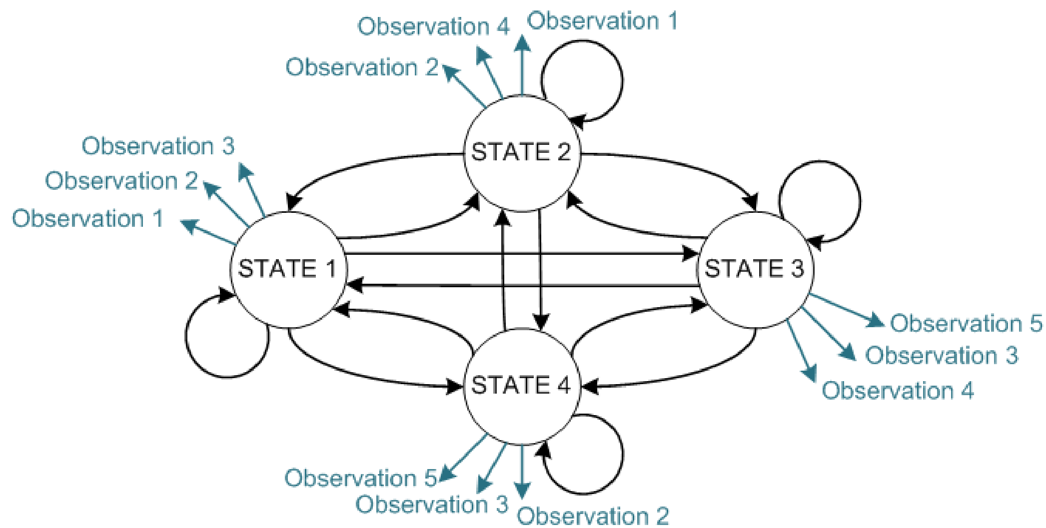
Hidden Markov Models



- A HMM can be used to address these problems:
  - **Prediction:** Given a Hidden Markov Model λ and a sequence of observations O, we would like to find out the state sequence which has the highest probability of generating O
  - ~~**Training:** Given a training set of observation sequences O, we would like to learn the model λ that maximizes the probability of generating O~~

Heuristic Rules

Modeling keyword search with a HMM



- HMMs are **parametric models**, and the parameters are:
  - N = number of states
  - Π = initial state probabilities
  - A = transition probability matrix N x N
  - B = emission probability for each state
- Mapping keywords to database elements:
  - keywords are the observable sequence O
  - the database elements are the hidden states to be inferred
- How do we calculate the HMM parameters?
  - N = || database vocabulary ||
  - Π → HITS algorithm
  - A → heuristic rules
  - B → similarity measures

Number of states

- N = || database vocabulary ||
  - Database vocabulary: the set of all the names of the tables, the attributes, and all the domains of the database.

Transition probability matrix A

- **Heuristic rules** based on the semantic relationships existing between the database terms (aggregation and generalization inferred by foreign key constraints)
  - the transition probability values decreases with the distance of the states.
  - Higher probabilities are associated to:
    - transitions to elements inside the same table
    - transitions between elements in tables connected through foreign keys

Emission probabilities

- The database vocabulary is composed of schema and domain elements.
- For schema elements:
  - Similarity measures
    - similarity value = P(schema element | keyword)
    - the Bayes theorem to calculate P(keyword | schema element), which is the emission probability
- For domain elements:
  - data types/ regular expressions/Google similarity to calculate the similarity of keywords and domains

Initial state probabilities

- We use an adaptation of the **HITS algorithm**.
  - the HITS algorithm is a link analysis algorithm used to rank web pages.
  - the algorithm calculates two rank values for each state: **hub** and **authority**.
    - a state is a good hub if it links to many good authority states, and a good authority is a state linked by many good hubs.
  - we take into account the number of attributes minus the number of foreign keys in a table

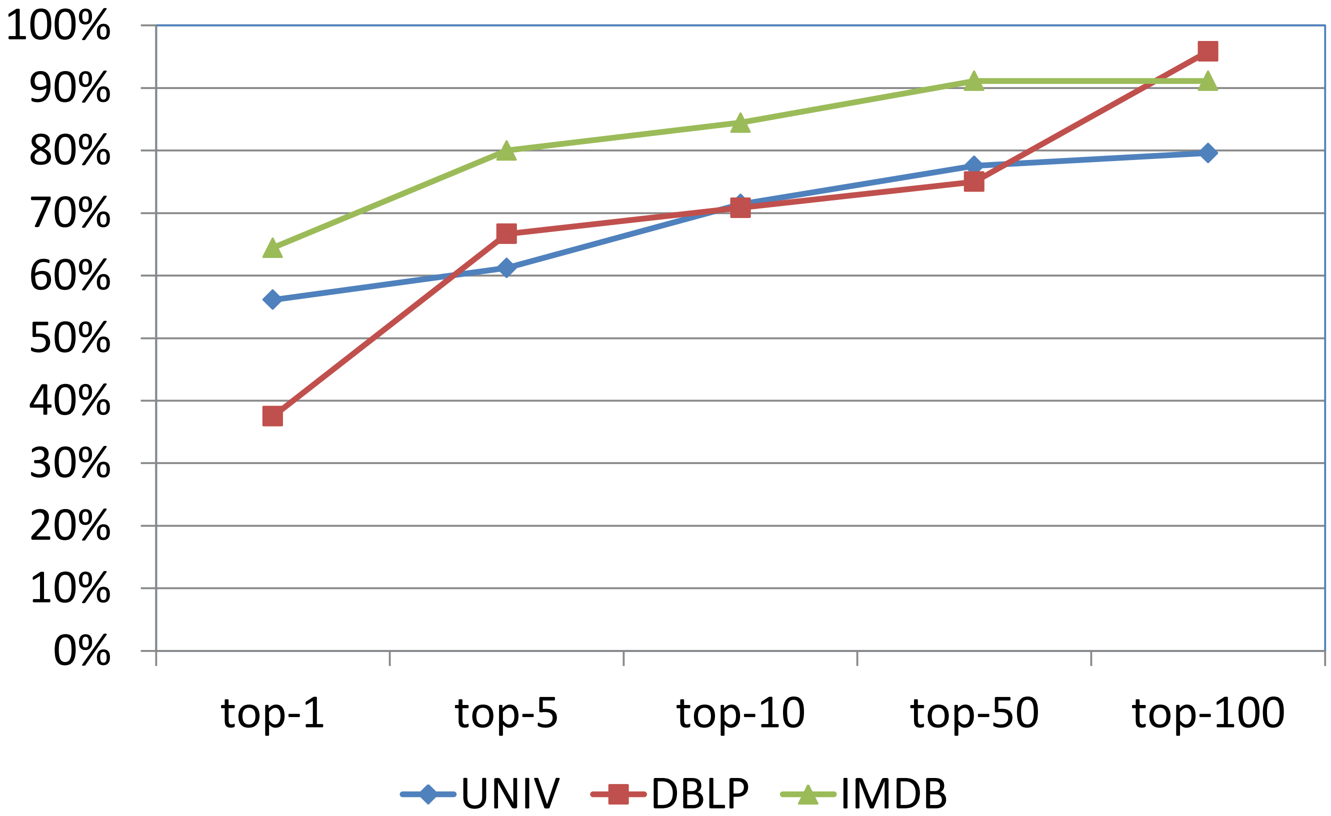
Prediction

- Given a HMM and a sequence of observations we use the **List Viterbi algorithm** to predict which are the best correspondent top-k state sequences, i.e. the database terms.
- The algorithm, which is a dynamic programming procedure, makes the following assumptions:
  - both the observations and the states must be in a sequence
  - a single element in the observation needs to correspond to exactly one state
  - computing the most likely state sequence up to a certain point *t* must depend only on the observed element at point *t*, and the most likely state at point $t - 1$

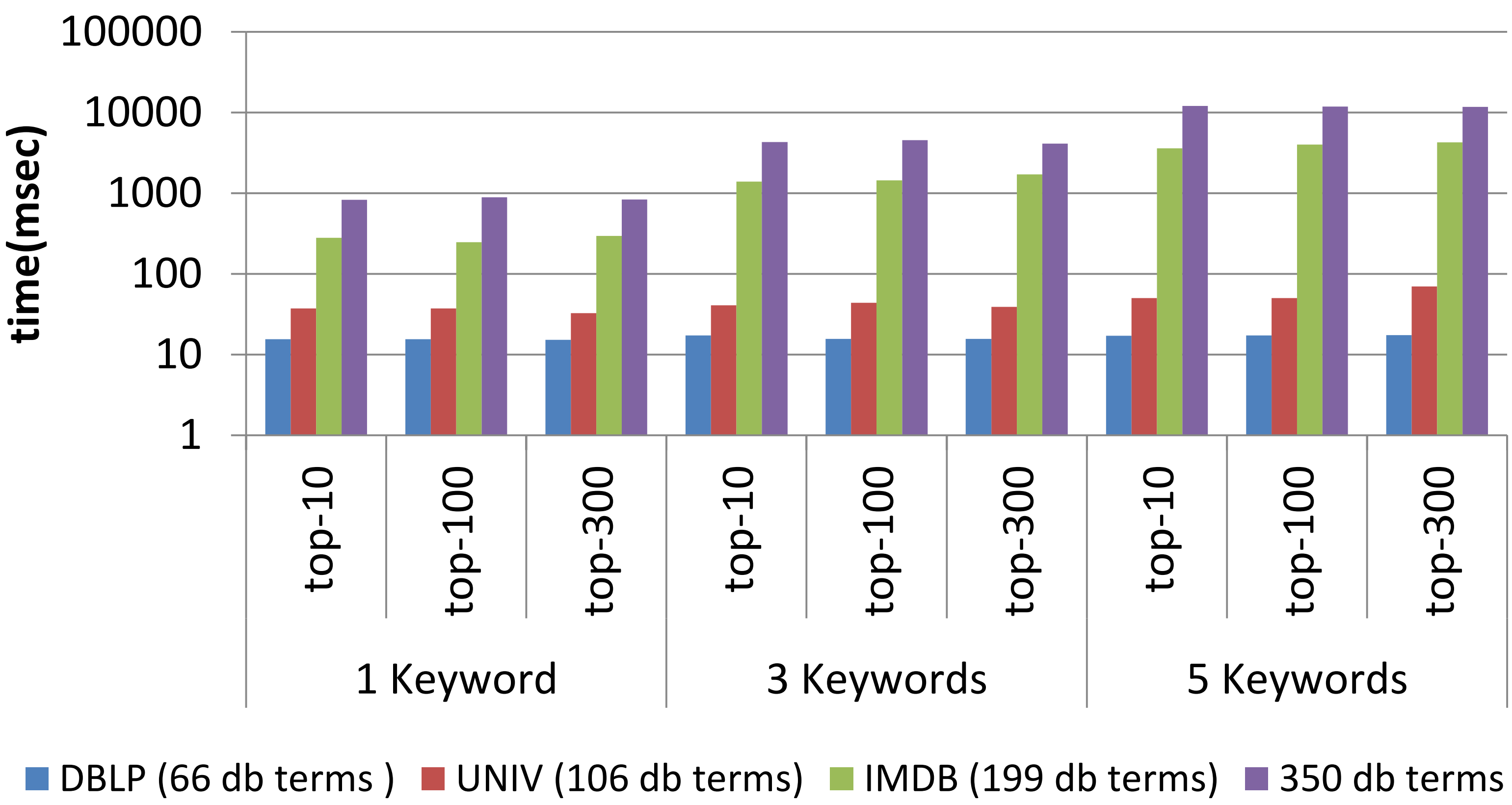
Evaluation

- We selected for our experiments three real data sets.
  - a university database
  - a fraction of the IMDB database
  - a fraction of the DBLP database
- 29 real users were asked to provide a set of keyword queries.
- A database expert translated each keyword query into a configuration.
- We used a total of 99, 44 and 30 queries for the university, the IMDB, and the DBLP database.

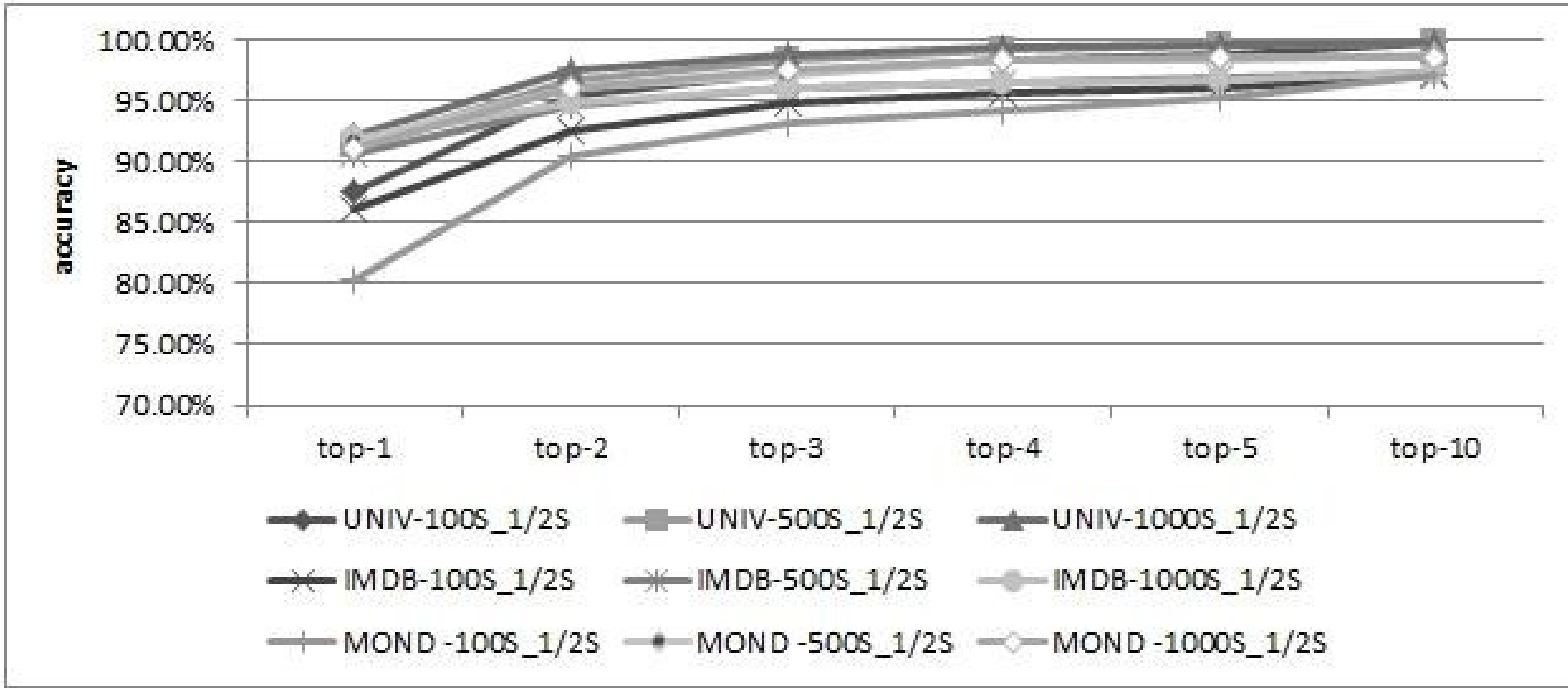
Evaluation - Effectiveness



Evaluation - Efficiency



Effectiveness with a training dataset



- Typical Expectation-Maximization approach extended
  - It bases the expectation step on the List Viterbi algorithm
- Silvia Rota, Sonia Bergamaschi, Francesco Guerra: The list Viterbi training algorithm and its application to keyword search over databases. CIKM 2011: 1601-1606

Convergence of the training algorithm