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
Motivating example

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
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”

1. Director \(\rightarrow\) Department.Director Watson \(\rightarrow\) \(\text{dom(Department.Director)}\) Address \(\rightarrow\) Department.Address
2. Director \(\rightarrow\) Department.Director Watson \(\rightarrow\) \(\text{dom(Department.Director)}\) Address \(\rightarrow\) Person.Address
3. Director \(\rightarrow\) Department.Director Watson \(\rightarrow\) \(\text{dom(Department.Address)}\) Address \(\rightarrow\) Person.Address
Motivating example (3)

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

<table>
<thead>
<tr>
<th>Department</th>
<th>id</th>
<th>DName</th>
<th>Address</th>
<th>Director</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watson</td>
<td>x123</td>
<td>CS</td>
<td>15 Tribeca</td>
<td>Hunt</td>
</tr>
<tr>
<td>Lenzerini</td>
<td>cs34</td>
<td>IE</td>
<td>5 Charles</td>
<td>Date</td>
</tr>
<tr>
<td>Hunt</td>
<td>m111</td>
<td>ME</td>
<td>2 Cottle</td>
<td>Hunt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Watson</td>
</tr>
<tr>
<td>Lenzerini</td>
</tr>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Hunt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Affiliated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professor</td>
</tr>
<tr>
<td>Watson</td>
</tr>
<tr>
<td>Lenzerini</td>
</tr>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Hunt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Lenzerini</td>
</tr>
<tr>
<td>Date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
</tr>
<tr>
<td>Data Integration</td>
</tr>
<tr>
<td>Foundation Matters</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>DBLP</td>
</tr>
<tr>
<td>ACM DL</td>
</tr>
</tbody>
</table>
From Keywords to Queries

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

<table>
<thead>
<tr>
<th></th>
<th>$R_1$</th>
<th>$...$</th>
<th>$R_n$</th>
<th>$A_1^R$</th>
<th>$...$</th>
<th>$A_{n1}^R$</th>
<th>$...$</th>
<th>$A_{1n}^R$</th>
<th>$...$</th>
<th>$A_{nn}^R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyword$_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>keyword$_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>keyword$_k$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The problem is known as Bipartite Weighted Assignations, 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 relativity 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 of Schema Database Terms

Step 2: Selection of Best Mappings to Schema Terms

Step 3: Contextualization of VW based on $M^*_i$

Step 4: Selection of Best Mappings to Value Terms

Step 5: Generation of the Configurations

Query keywords + Schema Information

SW

VW

Mappings $M^*_i$, with $i = 1..n$

Pairs $< M^*_i, VW_i >$, with $i = 1..n$

Pairs $< M^*_i, M^*_{ik} >$, with $i = 1..n$ and $k = 1..m$,

..... Configurations $C_k$ with $k = 1..m$

..... Interpretations

Query, ... Query
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:

<table>
<thead>
<tr>
<th></th>
<th>Schema Weights</th>
<th>Value Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntactic tech.</td>
<td>similarity measures based on edit distances, ...</td>
<td>Regular expressions</td>
</tr>
<tr>
<td>Semantic tech.</td>
<td>Lexical analysis</td>
<td>Data types, google similarity, ...</td>
</tr>
</tbody>
</table>
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

- 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

<table>
<thead>
<tr>
<th></th>
<th>P.N</th>
<th>P.A</th>
<th>P.P</th>
<th>P.Ad</th>
<th>P.E</th>
<th>D.I</th>
<th>D.D</th>
<th>D.A</th>
<th>D.Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>30</td>
<td>18</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>50</td>
<td>75</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Hopkins</td>
<td>45</td>
<td>30</td>
<td>0</td>
<td>35</td>
<td>10</td>
<td>32</td>
<td>40</td>
<td>35</td>
<td>39</td>
</tr>
<tr>
<td>Mary</td>
<td>46</td>
<td>20</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>Summerhill</td>
<td>10</td>
<td>7</td>
<td>0</td>
<td>34</td>
<td>22</td>
<td>20</td>
<td>10</td>
<td>32</td>
<td>15</td>
</tr>
</tbody>
</table>

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

- **1st position**: University > IMDB
- **not in 1st position**: University < IMDB
- **not found**: University < IMDB
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»
FIRST ISSUE: to decide which part of the database models the intended keyword meaning.
Motivating example

User query «Garcia-Molina Journal 2011»

Journals where Garcia Molina was an author in 2011

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

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

SECOND ISSUE: to decide how the retrieved structures relate to each other
A HMM can be used to address these problems:

- **Prediction**: Given a Hidden Markov Model $\lambda$ 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 $\lambda$ that maximizes the probability of generating $O$.

Heuristic Rules
HMMs are **parametric models**, and the parameters are:

- $N =$ number of states
- $\Pi =$ initial state probabilities
- $A =$ transition probability matrix $N \times N$
- $B =$ 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 $| |$
- $\Pi \rightarrow$ HITS algorithm
- $A \rightarrow$ heuristic rules
- $B \rightarrow$ similarity measures
Number of states

- \( N = || \text{database vocabulary} || \)
  - 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 = $P(\text{schema element} | \text{keyword})$
    - the Bayes theorem to calculate $P(\text{keyword} | \text{schema element})$, 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

![Graph showing effectiveness for top-1, top-5, top-10, top-50, and top-100 for UNIV, DBLP, and IMDB]
Evaluation - Efficiency

![Chart showing efficiency comparison with different datasets and keyword counts]
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