Processing Keyword Queries under Access Limitations

Andrea Calì, Thomas Lynch, Davide Martinenghi, Riccardo Torlone
What is the Deep Web?

• Web pages (HTML mostly) have been indexed and searched for many years
• Such pages constitute the so-called Surface Web
  • huge, valuable amount of information
• The web has also continuously “deepened”
  • searchable databases, accessible usually through forms
• The Deep Web (aka Hidden Web or Invisible Web) is not effectively crawlable nor indexeable
  • it is largely unexplored, apart from manual queries issued by users
The Deep Web

The Public Web

Only 4% of Web content (~8 billion pages) is available via search engines like Google.

7.9 Zettabytes

The Deep Web

Approximately 96% of the digital universe is on Deep Web sites protected by passwords.

Source: The Deep Web: Semantic Search Takes Innovation to New Depths
Conceptual view of the Deep Web
[He et al. 2007]
Modeling the deep Web

• Each source is modeled as a relational table with access limitations
• Access limitations: input vs output attributes
  • We can only access a table if we can provide a value for every input attribute
  • Access pattern: maps attributes into an access mode: input (i) or output.

People(FirstName, LastName, State)
Keyword Search in the Deep Web

• Accessing the deep Web:
  • Traditionally, conjunctive queries over data sources with access limitations

• Goal:
  • Provide a high-level access to Deep Web
  • Free the user from the knowledge of:
    • Query languages
    • Structure of data sources

• Approach:
  • Keyword-based queries
Join graph

\[ r_1 = \begin{array}{cc}
A_1 & A_2 \\
c_0 & c_1 \\
c_2 & c_3 \\
\end{array} \begin{array}{c}
t_{11} \\
t_{12} \\
\end{array} \quad r_2 = \begin{array}{cc}
A_2 & A_1 \\
c_1 & c_2 \\
c_4 & c_2 \\
c_1 & c_6 \\
\end{array} \begin{array}{c}
t_{21} \\
t_{22} \\
t_{23} \\
\end{array} \quad r_3 = \begin{array}{ccc}
A_1 & A_2 & A_3 \\
c_2 & c_1 & c_9 \\
c_5 & c_4 & c_9 \\
c_6 & c_7 & c_9 \\
\end{array} \begin{array}{c}
t_{31} \\
t_{32} \\
t_{33} \\
\end{array} \]
Answers to keyword queries

• A **keyword query** is a set of constants called keywords
• An **answer** to a keyword query $q$ against a database instance $r$ over a schema $R$ with access limitations is a set of tuples $A$ in the reachable instance such that:
  1. Each keyword in $q$ occurs in at least one tuple $t$ in $A$;
  2. The join graph of $A$ is connected;
  3. for every subset $A'$ of $A$ such that $A'$ enjoys Condition 1, the join graph of $A'$ is not connected.
• An answer is **optimal** if it has minimum size.
Computing an optimal answer

\[ r_1 = \begin{bmatrix} A_1^t & A_2 \\ k_1 & c_1 \end{bmatrix} t_{11} \begin{bmatrix} k_1 & c_1 \end{bmatrix} t_{12} \]

\[ r_2 = \begin{bmatrix} A_2^t & A_1 \\ c_1 & c_2 \end{bmatrix} t_{21} \begin{bmatrix} c_1 & c_2 \end{bmatrix} t_{22} \]

\[ r_3 = \begin{bmatrix} A_1^t & A_2 & A_3 \\ c_2 & c_1 & k_2 \end{bmatrix} t_{31} \begin{bmatrix} c_2 & c_1 & k_2 \end{bmatrix} t_{32} \]

\[ q = \{k_1, k_2\} \]
A method for computing an answer

A brute-force approach:
1. Extract the reachable portion
2. Find an optimal (or at least minimal) answer in the reachable instance
Data complexity

1. Extraction of the reachable instance
   • It can be implemented by a Datalog program $P$ over the input database $d$,
   • $P$ can be evaluated in polynomial time in the size of $d$ [Vardi 82].

2. Determining an optimal answer from the reachable instance
   • It corresponds to finding a Steiner Tree (ST) of its join graph, i.e., a minimal-weight subtree of this graph involving a subset of its nodes.
   • STs can be enumerated in ranked-order with polynomial delay, i.e., the time for printing the next optimal answer is polynomial in the size of $d$ [Kimelfeld and Sagiv 2006].

An optimal answer to a keyword query against a database instance with access limitations can be efficiently computed under data complexity.
Conclusions

• Formalization of keyword-based query answering in the Deep Web
• Preliminary insights on possible methods for computing optimal answers
• It turns out that:
  • The problem it is not easy to solve even over a few data sources
  • Traditional techniques for query answering in the Deep Web need to be revised
  • Even in the worst case the problem remains tractable
Current and Future work

• Optimization strategies for query answering
  • conditions under which an optimal answer can be derived without extracting the whole reachable instance;
• Implementation
  • based on the Dataplex framework
• Adoption of schema-based techniques
  • e.g., when the domains of the keywords are known in advance
• Take into account source availability and proximity
  • they can be modeled as weights on nodes and arcs, respectively