Accessing the Deep Web with Keywords: A Foundational Approach

Andrea Calí[1,2] and Martín Ugarte[3]

[1]Dept of Comp. Sci. and Inf. Syst.
Birkbeck, Univ. of London, UK

[2]Oxford-Man Inst. of Quantitative Finance
University of Oxford, UK

[3]Laboratory of Web and Information Technologies
Université Libre de Bruxelles

andrea@dcs.bbk.ac.uk, mugartec@ulb.ac.be

Abstract. The Deep Web is constituted by data that are generated dynamically as the result of interactions with Web pages. The problem of accessing Deep Web data presents many challenges: it has been shown that answering even simple queries on such data requires the execution of recursive query plans. There is a gap between the theoretical understanding of this problem and the practical approaches to it. The main reason behind this is that the problem is to be studied by considering the database as part of the input, but queries can be processed by accessing data according to *limitations*, expressed as so-called access patterns. In this paper we embark on the task of closing the above gap by giving a precise definition that reflects the practical nature of accessing Deep Web data sources. In particular, we define the problem of querying Deep Web sources with keywords. We describe two scenarios: in the first, called *unrestricted*, there query answering algorithm has full access to the data; in the second, called *restricted*, the algorithm can access the data only according to the access patterns. We formalise the associated decision problem associated to that of query answering in the Deep Web, explaining its relevance in both the aforementioned scenarios. We then present some complexity results.

1 Introduction

The *Deep Web* (also called *Hidden Web*) [9, 3, 6] refers to the data content that is created dynamically as the result of interactions with the Web. For example, when we search for a person in a White Pages website, the generated output consists of one or more pages containing the result of a query posed on an underlying database; these pages cannot be indexed by search engines and the underlying database cannot be freely queried by users. When we search in whitepages.com through a form, we are forced to fill in certain fields of the form, for instance the Name field; the result is then structured as a table. A Deep Web source can be naturally modeled as a relational table (or a set of relational tables) that can be queried only according to so-called *access patterns*, each of which enforces the selection on some of the attributes (i.e. the filling of input fields on a form), which

are called *input* attributes. Relational tables accessible through access patterns then are said to have *access limitations*.

Obtaining data dynamically from Deep Web sources is the key problem in the integration of such sources. Interestingly, when Deep Web sources are modelled as relations with access limitations, answering a simple query on such sources require, in the worst case, the evaluation of a *recursive* Datalog query plan [7, 3]. In such plans, values obtained as output from one search are used as input for other sources.

In this paper we study the problem of *accessing Deep Web sources via keywords*, that is, given a query, a set of keywords with which to access the sources and a database, computing the answers to the query. Interestingly, the keywords in this context are not used to select the answers or the sources, but to *retrieve* data from the sources. This problem is related to conjunctive query (CQ) answering, an extensively studied topic in the literature [1, 7, 3]; however, in the case of sources with access limitations a thorough theoretical study of the central problems and their computational complexity is surprisingly still lacking. In this paper we present the keyword querying problem in a formal way, distinguishing two variants of it. We then provide results on the computational complexity of the boolean case of both variants.

2 Preliminaries

We consider the relational setting extended with *access limitations* and *abstract domains*. We assume the reader is familiar with the well-known notions of relations, attributes, variables, constants and ground atoms (a.k.a. facts); and the relational setting in general. For formal definitions we refer the reader to [1].

Access limitations on a relation are constraints imposing that certain attributes must be *selected* (that is, bound to a constant) for the relation to be accessed. More formally, a schema with access limitations is a pair $\langle \mathcal{R}, \Lambda \rangle$, where $\mathcal{R}$ is a relational schema (a set of relations) and Λ is a set of access limitations that specifies, for every attribute of every relational predicate, whether it is an *input* or an *output* attribute; in order to access a relation, all input attributes must be selected[1]. We indicate the access limitations of each relation as a sequence, of 'i' and 'o' symbols written as a superscript in the signature of the relation; an 'i' (resp., 'o') indicates that the corresponding argument is an input (resp., output) argument. A signature has therefore the form r^{Λ_r}, where Λ_r represents the access limitation on r. In our setting (see also [3]) some general domains, called *abstract domains*, are associated to attributes; these attributes are used to distinguish, for instance, strings representing names from strings representing addresses. To avoid notational clutter, we assume that attribute names are assigned so that attributes having the same abstract domain also have the

[1] In general, there could be more than one annotation for each predicate, that is, more than one way of accessing the corresponding relation. However, in this paper we assume there is exactly one access limitation (or pattern) per predicate. Our results can be extended to the general case.

same name. The problem that we study in this paper consist of two parts: a database is first accessed through access limitations following the known abstract domains, and the *obtainable* part of the database is then queried by using the most common class of queries, namely *conjunctive queries* (CQs) [1].

In the presence of access limitations on the sources, queries cannot be evaluated as in the traditional case. As we don't have direct access to the database, we need a set I of initial keywords to start *scraping* the database. This has been previously noted in [8], where the authors present an algorithm that extracts all *obtainable* tuples in the answer to the query. This algorithm compiles the evaluation strategy into a suitable Datalog program, which encodes both the access limitations on the sources and the query itself, and is evaluated as follows: starting from a set of initial keywords (that must include those appearing as constants in the query), we access all the relations we can according to the access limitations. With the new facts (if any), we obtain new keywords with which we can repeat the process and access the relations again, until we have no way of making new accesses. The program extracts all facts obtainable while respecting the access limitations, but there may be facts in the sources that cannot be retrieved.

Example 1. Consider the relations $r_1^{ioo}(N, D, C)$ and $r_2^{ioo}(C, S, N)$ depicted in Figure 2. The tuples in r_1 are conformed by a *Nation*, a typical *Dish* of that nation, and a famous *Chef* that cooks it; the tuples in r_2 contain a *Chef*, the amount of Michelin *Stars* he has obtained and his *Nationality*. The access limitations only allow for searching typical dishes (r_1) by nation, and searching Chefs by last name (we assume C contains the last name of the Chef). Assume now that we want to obtain the set of dishes that are prepared by Chefs with three Michelin stars. This is naturally expressed by the conjunctive query $q(D) \leftarrow r_1(N_1, D, C), r_2(C, 3, N_2)$. However, because of the access limitations we cannot directly pose this query to the database. Instead, we need to recursively inspect the database starting from a set of keywords known in advance. For example, assume we know that this database contains information about dishes in Italy. Then, we can search r_1 using *Italy*, which returns tuple t_1. Now we have the last name of chef Heinz Beck, so we can search r_2 with input *Beck*. This will return tuple s_1, indicating that *risotto* is a dish prepared by a chef with three Michelin stars (and thus part of our answer). But s_1 also contains the value *Germany*, which we can use as input to query r_1 again; this time we get t_2, which contains the last name of a new chef, Ducasse. We can now query r_2 with input *Ducasse*, obtaining tuple s_2 and realizing that *Magenbrot* is also part of our answer. Tuple s_2 also contains a new country, France. However, when we query r_1 with *France* we get an empty result, and therefore there are no more tuples we can obtain from the database. Note that *Onigiri* would also be part of the answer if we had complete access to the data. However, because of the access limitations, we cannot retrieve this value unless Japan is one of our initial keywords.

The recursive procedure illustrated above is formalised by evaluating the Datalog program depicted in Figure 1. In this program, relations $\hat{r}_1$ and $\hat{r}_2$

ρ_1 : $q(D) \leftarrow \hat{r}_1(N, D, C), r_2(C, 3, N)$
ρ_2 : $\hat{r}_1(N, D, C) \leftarrow r_1(N, D, C), dom_N(N)$
ρ_3 : $\hat{r}_2(C, S, N) \leftarrow r_2(C, S, N), dom_C(C)$
ρ_4 : $dom_C(C) \leftarrow \hat{r}_1(N, D, C)$
ρ_5 : $dom_A(N) \leftarrow \hat{r}_2(C, S, N)$
ρ_6 : $dom_N(Italy)$

Fig. 1. Datalog program of Example 1.

r_1^{ioo}

	N	D	C
t_1	Italy	Risotto	Beck
t_2	Germany	Magenbrot	Ducasse
t_3	Japan	Onigiri	Robuchon

r_2^{ioo}

	N	S	C
s_1	Beck	3	Germany
s_2	Ducasse	3	France
s_3	Robuchon	3	France

Fig. 2. Database of Example 1.

represent the *obtainable* parts of r_1 and r_2, respectively (assuming we start only from the keyword $Italy$). Rule ρ_1 represents the original query (over the *obtainable* versions of r_1 and r_2), rules ρ_2 to ρ_5 encode the recursive access to the sources, and ρ_6 simply *initializes* the constant $Italy$ by adding it to the abstract domain of nations. ∎

The previous example shows the typical way in which Deep Web sources are accessed an queried over the web. We now define the notion of answer and the *obtainable* portion of a database under access limitations; such a portion is determined by the initial keywords. Given a CQ q posed over a schema $\mathcal{S} = \langle \mathcal{R}, \Lambda \rangle$, a set of initial keywords $I \subseteq \Delta$, and a database D over schema $\mathcal{R}$, $\rho_{\Lambda,I}(D)$ denotes the set of facts of D that can be recursively obtained under Λ starting from I. The set of answers to q over D with access limitations Λ and initial set of keywords I is denoted by $\text{ans}(q, \Lambda, D, I)$ and is defined as the set of answers, in the classic sense, to q over $\rho_{\Lambda,I}(D)$. If q is a Boolean CQ (i.e., with zero-arity head), we write $D \models_{\Lambda,I} q$ when q is true on $\rho_{\Lambda,I}(D)$ (denoted $\rho_{\Lambda,I}(D) \models q$).

3 The complexity of querying under access limitations

In this section we study the complexity of answering queries on Deep Web datasets, where data are to be extracted from an initial set of keywords. Surprisingly, the notions of complexity present in the literature do not seem to fully capture the correct difficulty of the problem. To clarify this problem, we present two variants of the Boolean query answering problem (the extension to the non-Boolean case is straightforward).

Definition 1. *Given a database D, a set of initial keywords $I \subseteq \Delta$, a set Λ of access limitations and a BCQ q, the problem of query answering with initial keywords I and query q, on database D and under Λ, is to determine whether $D \models_{\Lambda,I} q$. This is defined in two variants:*

(i) unrestricted *case: this is the problem of determining whether $D \models_{\Lambda,I} q$, while having* arbitrary *access to the relations of D.*
(ii) restricted *case: this is the problem of determining whether $D \models_{\Lambda,I} q$, while having access to the relations of D* only *according to Λ.*

Notice that the problem in the restricted case is the "classic" case [3, 2], where we are computing the answers to a CQ having only limited access to the data, according to Λ. The CQ answering problem in the unrestricted case is also relevant in real-world scenarios. Assume for example that access limitations are enforced by an organisation in order to limit access to data by external users (e.g., those outside the organisation). The organisation has arbitrary access to the data, and it is interested in determining what external users, who probably know certain initial keywords and whose access to the data is limited by Λ, can retrieve from the database. In order to determine this, of course, an algorithm will have the advantage of freely accessing the data, regardless of access limitations.

We need to point out that in the restricted case, if we are to tackle the search problem formally, we need to understand the relations with access limitations as *oracles*; each access (that consists in the processing of an atomic query on a single relation) is a call to an oracle corresponding to the same relation. The execution of such a query takes evidently (at most) linear time in the size of the instance of the relation, therefore the oracle does not really serve to determine a complexity class, as is done, for example, when we have a class $\mathcal{C}_1^{C_2}$ of problems that can be decided by an algorithm of class $\mathcal{C}_1$ that can call an oracle solving problems in class $\mathcal{C}_2$ (each costing 1, given the nature of the oracle). The oracles in our case do not add computational power; instead they *limit* the access to the data rather than allowing the solution to a problem instance in constant time. In this case, the algorithm cannot receive the instance D as (fully accessible) input; yet we want to measure its complexity considering the size of D. Interestingly, this is why the classical notion of complexity does not capture the actual difficulty of the problem. The following example shows that there are instances in which the two variants of the problem actually present different complexity.

Example 2. Consider the schema $\mathcal{R} = \{r^{i\cdots i}\}$, constituted by a single relation with k input arguments and no output arguments, the single-tuple database $D = \{r(a_1, \ldots, a_k)\}$, and the keyword set $I = \{c_1, \ldots, c_m\} \supseteq \{a_1, \ldots, a_k\}$, and the atomic Boolean CQ q defined as $q() \leftarrow r(X_1, \ldots, X_k)$. To determine whether $r(a_1, \ldots, a_k) \in \text{ans}(q, \Lambda, D, I)$, according to the access restrictions we need to try accessing the relation $r(D)$ with all possible k-tuples of constants of I. This corresponds to the restricted case and, in the worst case, this will necessarily require m^k searches on r. On the contrary, if we had direct access to the database the query would be answered trivially, as we can directly see that $r(a_1, \ldots, a_k)$ is part of the database. ∎

The case of the example, which uses very simple CQs (atomic Boolean CQs), is trivial in the unrestricted case, but in the restricted case every deterministic keyword search algorithm requires at least exponential time.

We now briefly discuss a result, stated in previous works [4, 5], on CQ answering in our setting. The result states that CQ answering is NP-complete both in the restricted and the unrestricted case. In the light of Example 2, this is somewhat counterintuitive, as the restricted case appears to be computationally more difficult than the unrestricted case. Regarding upper bounds, the

most interesting technique is the one used to prove that the problem is in NP in the restricted case: a non-deterministic algorithm is exhibited, whose maximum number of steps, in the worst case, is surprisingly bounded by the number of atoms in the database D. The lower bounds are given instead by the obvious NP lower bound of CQ answering in the case *without* access limitations. However, the lower bound in the restricted case does not constitute a fully satisfactory study of the complexity as it does not make use of the restrictions given by the presence of the oracles; indeed it still remains to *define*: *(1)* what kind of computational model we need to model the oracles; *(2)* what kind of reduction would imply a complexity lower bound in this setting. In addition, it remains to understand whether simpler classes of CQs (e.g. atomic, acyclic or bounded-treewidth CQs) enjoy lower complexity. This will be the subject of future investigation.

4 Discussion

In this paper we have introduced two variants of the problem of querying Deep Web sources with a set of initial keywords, namely the *restricted* and an *unrestricted* case. We have shown that the two variants can differ by an exponential factor in very simple cases. However, the problem of CQ answering with keywords under access limitations has been shown to be NP-complete in both the restricted and unrestricted case. As future work we plan to carry out a formal definition of the associated decision problem for the restricted case, and the characterization of those classes of queries for which the complexity of the two variants differ. For instance, we will investigate whether the NP lower bound holds for atomic queries in the restricted case (which is the case of Example 2), or for other restricted classes of CQs such as acyclic or bounded-treewidth CQs.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. Andrea Calì and Davide Martinenghi. Conjunctive Query Containment under Access Limitations. In *Proc. of ER*, 2008.
3. Andrea Calì and Davide Martinenghi. Querying data under access limitations. In *Proc. of ICDE*, 2008.
4. Andrea Calì, Davide Martinenghi, Igor Razgon, and Martín Ugarte. Querying the deep web: Back to the foundations. In *Proc. of AMW*, 2017. To appear.
5. Andrea Calì and Igor Razgon. Complexity of conjunctive query answering under access limitations (preliminary report). In *Proc. of SEBD*, 2014.
6. Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *Proc. of CIDR*, 2005.
7. Chen Li. Computing complete answers to queries in the presence of limited access patterns. *Very Large Database Journal*, 12(3):211–227, 2003.
8. Chen Li and Edward Chang. Query planning with limited source capabilities. In *Proc. of ICDE*, 2000.
9. Jayant Madhavan, Loredana Afanasiev, Lyublena Antova, and Alon Y. Halevy. Harnessing the deep web: Present and future. In *Proc. of CIDR*, 2009.