# A Framework for Conjunctive Query Answering over Distributed Deep Web Information Resources

Andrea Calì[1,3] and Umberto Straccia[2]

[1] Dept of Computer Science and Information Systems
Birkbeck University of London, UK
andrea@dcs.bbk.ac.uk
[2] Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche, Pisa, Italy
straccia@isti.cnr.it
[3] Oxford-Man Institute of Quantitative Finance
University of Oxford, UK

**Abstract.** Deep Web Information Resources (DWIRs) are data that are accessible through web forms but are not indexable by search engines. We propose a novel framework to tackle the problem of conjunctive query (CQ) answering over a mediated schema in which the local resources are DWIR. To this aim, we propose to use techniques from the field of Distributed Information Retrieval (DIR). We discuss a novel approach to automated DWIR sampling, size estimation and selection, as well as an approach to result list merging.

## 1 Introduction

The *Deep Web* (a.k.a. *Hidden Web*) is the set of data that are accessible on the Internet, usually though HTML forms, but are not normally indexable by search engines, as they are returned in dynamic pages. By Deep Web Information Resources (DWIRs) we denote sources in the aforementioned Deep Web. It is immediately seen that accessing data through a DWIR through a form is logically equivalent to querying a relational table, where a selection is specified by the fields of the form that are filled in with values. Typically, certain fields are required to be filled in by the user in order to obtain a result; for instance, any weather information web site will require at least a location name as input. An analogous situation occurs in some legacy systems where data, stored in legacy structures, are wrapped as relational tables.

In this paper we propose a framework for the integration of DWIRs in a global-as-view approach [9], where queries as posed on a virtual mediated schema where elements (relational entities) are defined by *mappings* that are views on the data sources. Limitations on how sources can be accessed (fields to be necessarily filled in) significantly complicate query processing: as shown, e.g., in [10], query answering in the presence of access limitations in general requires the evaluation of a *recursive* query plan.

We tackle the CQ answering problem from a novel point of view. We assume a setting where each relational entity in the mediated schema can be associate to several local databases (DWIRs). A CQ $q$ can therefore be rewritten into a query over the local databases in several ways, thus generating a large number of possible *rewritings*. Our goal is therefore to select small and meaningful rewriting sets, assuming we have a suitable criterion for ordering rewritings, and then merge the results obtained for each rewriting so as to get the "best" answers. Notice that this approach necessarily returns a partial answer to $q$.

Our contribution is the following.

1. We propose a technique for sampling DWIRs, similar to those adopted for textual databases; the samples are stored in a suitable database.
2. We illustrate how to automatically select resources by using a scoring function. We employ techniques adapted from textual distributed information retrieval so as to select a small number of rewritings to be actually processed on the source databases.
3. We propose a solution to the problem of ranked list merging, that is, to rank the results of the evaluation of the different rewritings processed on the sources.

## 2 Preliminaries

### 2.1 Integration of DWIRs

The query answering setting we are going to consider is *Global As View* (GAV) [9] over distributed web databases, *i.e.*, in which a query over the mediator schema is resolved via queries over distributed local web databases, called the local DBs (see Figure 1). Informally (see, *e.g.*, [21]), when a user query $q$ over the global schema $\mathcal{G}$ is issued to the *Mediator*, the *Mediator Engine* performs the following tasks.

1. It rewrites the query $q$ into queries $q_i$ over of the local schemas $\mathcal{S}$.
2. It asks the *Resource Selector*, which of the $q_i$ are the top-$k$ queries that likely will provide relevant results back (we assume here that it is too costly to submit all the rewritings to the underlying local databases).
3. It submits the selected queries $q_i$ to the wrappers that interface with the distributed web resources.
4. Finally, it merges the $k$ ranked lists into one final ranking, via the *Ranked List Merger*, and provides the result back to the user.

### 2.2 Answering Conjunctive Queries by Rewriting

To start with, we introduce next the basic notions of our framework.

1. Let $\mathcal{G} = \{R_1, \ldots, R_n\}$ $(n \geqslant 1)$ be the relational entities of the Mediator's global schema through which we are going to formulate our queries. Generally, the entities in $\mathcal{G}$ may be part of an *ontology*, *e.g.*, a Datalog program, OWL ontology or RDFs graph. However we do not yet consider this case.

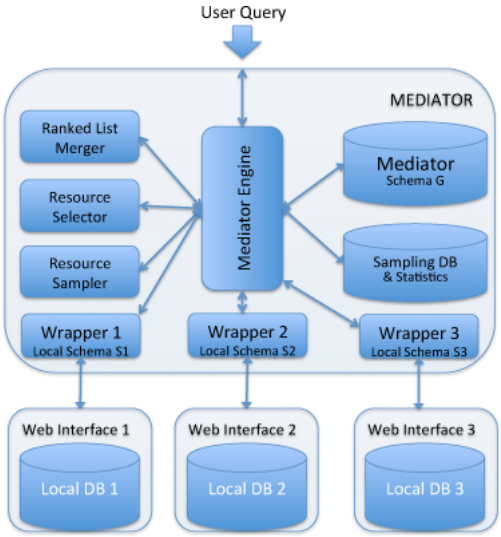


**Fig. 1.** Architecture of a Deep Web Mediator.

2. Let $\mathcal{DB} = \{DB_1, \ldots, DB_m\}$ $(m \geqslant 1)$ be the distributed, local web databases from which we would like to gather information by means of the mediator;
3. Let $\mathcal{S} = \{S_1, ..., S_m\}$ be the local relational entities through which we access the local databases $DB_i \in \mathcal{DB}$. Specifically, we have exactly one relation $S_i$ through which we access $DB_i$. For $S_i \in \mathcal{S}$, $DB_i \in \mathcal{DB}$, vector of variables and constants $\bar{\mathbf{z}}$, with $ans(S_i(\bar{\mathbf{z}}), DB)$ we denote the *answer set* to the local query $S_i(\bar{\mathbf{z}})$ over database $DB$, *i.e.*,

$$ans(S_i(\bar{\mathbf{z}}), DB) = \{\bar{\mathbf{t}} \mid DB \models S_i(\bar{\mathbf{t}}) \text{ s.t. } \bar{\mathbf{t}} \text{ agrees with } \bar{\mathbf{z}} \text{ w.r.t. the constants in } \bar{\mathbf{z}}\} \ .$$

We assume that the tuples in $ans(S_i(\bar{\mathbf{z}}), DB)$ are ordered and with $ans_k(S_i(\bar{\mathbf{z}}), DB)$ we denote the top $k \geqslant 1$ retrieved tuples in $ans(S_i(\bar{\mathbf{z}}), DB)$.
4. We further assume that more than one database may be used to instantiate a relation $R \in \mathcal{G}$.[4] We model this scenario by means of a set of *mapping* of rules. That is, for each $R \in \mathcal{G}$, let $\mathcal{M}_R$ be the set of $k_R$ *mapping rules* w.r.t. $R$ defined as

$$\begin{aligned} R(\bar{\mathbf{x}}) &\leftarrow S_{1_R}(\bar{\mathbf{x}}) \\ &\vdots \\ R(\bar{\mathbf{x}}) &\leftarrow S_{k_R}(\bar{\mathbf{x}}) \ , \end{aligned}$$

where $S_{i_R} \in \mathcal{S}$. These mappings may also have been computed automatically and, thus, there is an inherent uncertainty whether a mapping may be relevant to $R$ or not. More involved mappings rules may be considered as well, such as conjunction in the body, or graded rules to reflect the uncertainty of the rule. The automated generation of mappings rules falls under the research area of *schema matching* (see *e.g.*, [12,13,21]). With $\mathcal{M}$ we denote the set of all

[4] For instance, for $R(\texttt{carModel}, \texttt{year}, \texttt{partNumber}) \in \mathcal{G}$, there may be various web DBs through which the Mediator may find car spare parts.

*mapping rules* w.r.t. $R \in \mathcal{G}$, *i.e.*, $\mathcal{M} = \bigcup_{R \in \mathcal{G}} \mathcal{M}_R$. We assume that each $S \in \mathcal{S}$ is *typed* in the sense that each attribute $x_j$ of the relation $S(...., x_j, ...) \in \mathcal{S}$ has a type, such as number, date, string, etc.. There may be some other constraints by accessing local DBs such as *mandatory* attributes, access restrictions, costs, etc., but we do not yet address them here (see, *e.g.*, [3,5]);

5. Let a *conjunctive query* $q$ over the global schema $\mathcal{G}$ be a rule $r$ of the form

$$q(\bar{\mathbf{x}}) \leftarrow \exists \bar{\mathbf{y}}.\varphi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \ ,$$

where $\varphi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a conjunction of relations in $\mathcal{G}$. Variables in $\bar{\mathbf{x}}$ are called the *distinguished variables*, while those in $\bar{\mathbf{y}}$ are called the *non-distinguished variables*. We assume that each variable in $\bar{\mathbf{x}}$ occurs in at least one relation occurring in $\varphi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. The existential $\exists \bar{\mathbf{y}}$ may be omitted. The *answer set* of a conjunctive query $q$ expressed via rule $r$ is defined as the set $ans(q, \mathcal{DB}, \mathcal{M})$ of *answers* $\bar{\mathbf{t}}$ (vector of constants)

$$ans(q, \mathcal{DB}, \mathcal{M}) = \{\bar{\mathbf{t}} \mid \mathcal{DB} \cup \mathcal{M} \cup \{r\} \models q(\bar{\mathbf{t}})\} \ ,$$

*i.e.*, the query body of rule $r$ evaluates to true, given the set of databases and the mappings. As before, we will assume that $ans(q, \mathcal{DB}, \mathcal{M})$ is an ordered set. With $ans_k(q, \mathcal{DB}, \mathcal{M})$ we denote the top $k \geqslant 1$ retrieved tuples in $ans(q, \mathcal{DB}, \mathcal{M})$;

6. Let a *complex conjunction query* $q$ be a rule $r$ of the form (see [19,25])

$$\begin{aligned} q(\bar{\mathbf{x}}, \alpha) \leftarrow\ & \exists \bar{\mathbf{y}}.\varphi(\bar{\mathbf{x}}, \bar{\mathbf{y}}), \\ & \mathsf{GroupedBy}(\bar{\mathbf{w}}), \\ & \alpha := @[f(\bar{\mathbf{z}})] \ , \end{aligned}$$

where

(a) $\bar{\mathbf{w}}$ are variables in $\bar{\mathbf{x}}$ or $\bar{\mathbf{y}}$, each variable in $\bar{\mathbf{x}}$ occurs in $\bar{\mathbf{w}}$ and any variable in $\bar{\mathbf{z}}$ occurs in $\bar{\mathbf{y}}$

(b) $@$ is an aggregate function with $@ \in \{\mathsf{SUM}, \mathsf{AVG}, \mathsf{MAX}, \mathsf{MIN}, \mathsf{COUNT}\}$.

(c) $f$ is a built-in function, called *scoring function* with range the real numbers. The assignment $\alpha := @[f(\bar{\mathbf{z}})]$ is called *scoring atom*, $\alpha$ is called *scoring variable*

(d) The grouping, the aggregation operator, or the scoring atom may be omitted.

The answer set of a complex conjunctive query is defined in a similar way as for conjunctive queries, except that now answers have a score determined by the scoring atom. That is, the *answer set* of a complex conjunctive query $q$ expressed via a rule $r$ is defined as the set $ans(q, \mathcal{DB}, \mathcal{M})$ of *answers*

$$ans(q, \mathcal{DB}, \mathcal{M}) = \{\langle \bar{\mathbf{t}}, s \rangle \mid \mathcal{DB} \cup \mathcal{M} \cup \{r\} \models q(\bar{\mathbf{t}}, s)\} \ ,$$

where we assume that it can not be the case that both $\langle \bar{\mathbf{t}}, s \rangle$ and $\langle \bar{\mathbf{t}}, s' \rangle$ are in $ans(q, \mathcal{DB}, \mathcal{M})$ with $s \neq s'$ (therefore, each tuple has an unique score[5]). Tuples in $ans(q, \mathcal{DB})$ are assumed to be ordered in decreasing order of the score and the *top-k* answering problem ($k \geqslant 1$) consists in retrieving the top-$k$ answers of a query, possibly without computing the whole answer set (see, *e.g.*, [18,19,20]). With $ans_k(q, \mathcal{DB}, \mathcal{M})$ we denote the top $k \geqslant 1$ retrieved tuples in $ans(q, \mathcal{DB}, \mathcal{M})$.

[5] Otherwise, just consider the tuple with maximal score only and remove the other one.

An immediate solution to compute the answer set of a conjunctive query would consist in determining the *rewriting set* $r(q, \mathcal{M})$ of a query $q$ ($R'_i \in \mathcal{G}$) of the form

$$q(\bar{\mathbf{x}}) \leftarrow R'_1(\bar{\mathbf{z}}_1), \ldots, R'_k(\bar{\mathbf{z}}_k) \ .$$

That is, we compute the set of *rewritings* of $q$ having the form ($S'_i \in \mathcal{G}$)

$$q(\bar{\mathbf{x}}) \leftarrow S'_1(\bar{\mathbf{z}}_1), \ldots, S'_k(\bar{\mathbf{z}}_k) \ ,$$

where each $R'_i \in \mathcal{G}$ has been replaced with some $S'_i \in \mathcal{S}$ occurring in the set of *mapping rules* w.r.t. $R'_i$.

Notice that, as there may be as many as

$$rw = \prod_{R'_i} k_{R'_i} \tag{1}$$

rewritings for $q$ (recall that $k_{R'_i}$ is the number of mapping rules w.r.t. $R'_i$), this approach is likely going to fail in terms of response time for large $rw$ (*e.g.*, $rw \geqslant 100$).

## 3 CQ Answering over Distributed Resource

**Goal:** Our objective consists instead in applying techniques developed in the context of *Distributed Information Retrieval* (DIR) [22] to the conjunctive query answering problem over distributed DBs. That is, given a query $q$, how can we select the top-$s$ best rewritings $q' \in r(q, \mathcal{M})$ (with $s \ll |r(q, \mathcal{M})|$, *e.g.*, $s = 10$ and $|r(q, \mathcal{M})| = 100$) such that a suitable, objective criteria is met? [6]

We recall here that informally the procedure for DIR consists of three steps (for a recent overview, see *e.g.*, [15,22]). Given a query (list of keywords)

1. Compute automatically via sampling a meaningful sample of each information resource (called *resource sampling*, see *e.g.*, [6,7]) and store the data into the *Sampling database* (see Figure 1).
2. Using the samples, determine which are the top $k$ most relevant resources to answer the query (called *resource selection*, see *e.g.*, [17,23]).
3. Submit the query to the resources and merge the results (called *ranked list merging*, see *e.g.*, [11,14,16,24]). Notice that this step is not trivial as each resource returns a ranked list and the goal is here to merge all returned lists into an unique ranked list preserving relative orders among the items.

Our goal is now to adapt the ideas developed in DIRs to our setting (see also [21]).

1. Compute automatically a meaningful sample for each $DB \in \mathcal{DB}$ and store the data into the *Sampling database* (see Figure 1).
2. Using the DB samples, determine which are the top $s$ best query rewritings $q' \in r(q, \mathcal{M})$ according to some objective criteria. Notice that in our setting mapping rules may have been determined automatically, and therefore they are not all necessarily relevant to the corresponding global relation.
3. Submit the selected queries to the DBs and merge the results.

[6] Notice that in general we are computing a subset of $ans(q, \mathcal{DB}, \mathcal{M})$.

*Discussion.* We are going now to discuss briefly the above-mentioned three steps.

**Resource Sampling.** In order to select suitable DBs for a query, we need to know about the contents of each DB as well as other important information (*e.g.*, the size of the DB). We keep a representation set for each DB. The representation set of each DB contains information about the tuples that are indexed by that DB. Specifically, tuple statistics of the DB are approximated by using a number of tuples sampled from that DB. In order to do so, we plan to use a *query-based sampling* (QBS) approach in a similar way to what is done for textual databases [6,7]. Essentially, for each DB, we do the following.

1. We start with a random query;
2. We submit the query to the local DB and store the retrieved tuples in the sample DB.
3. We build a new query from the sample data.
4. We iterate steps 2 and 3 until a stopping condition holds; such a condition expresses the fact that the sample changed less than a certain amount in the last iteration.

**Automated Resource Selection.** Given a conjunctive query $q$ ($R'_i \in \mathcal{G}$)

$$q(\bar{\mathbf{x}}) \leftarrow R'_1(\bar{\mathbf{z}}_1), \ldots, R'_k(\bar{\mathbf{z}}_k) \tag{2}$$

consider its rewriting set $r(q, \mathcal{M})$, that is, the set of rewritings of $q$ the form ($S'_i \in \mathcal{G}$)

$$q(\bar{\mathbf{x}}) \leftarrow S'_1(\bar{\mathbf{z}}_1), \ldots, S'_k(\bar{\mathbf{z}}_k)$$

where each $R'_i \in \mathcal{G}$ has been replaced with all possible $S'_i \in \mathcal{S}$ occurring in the set of mapping rules w.r.t. $R'_i$. We indicate the rewritings as $q_1, \ldots, q_{rw}$ (see Eq. 1), where $rw = |r(q, \mathcal{M})|$. The goal in this task is to determine which of the $q_i$ are most likely to return relevant answers to $q$. To this aim, we plan to adapt the ReDDE.top method [1] to our setting; such a method is among the most effective in the literature for textual DIR, and it resembles somewhat so-called kNN-classifiers [8]. Essentially, for each query $q_i$, using the Sample DB, we estimate the 'goodness' of query $q_i$ w.r.t. $q$ in retrieving answers to $q$. This is done via a *scoring function* $s(q_i \mid q)$. Eventually, given query $q$ and its rewritings $q_1, \ldots, q_{rw}$, we rank the rewritings in decreasing order of the score $s(q_i \mid q)$ and select only the top-$s$ (with $s \ll |r(q, \mathcal{M})| = rw$, *e.g.*, $s = 10$) among them to be submitted to the real databases in $\mathcal{DB}$.

**Ranked List Merging.** Given the selected top-$s$ queries $q_1, \ldots, q_s$ identified in the automated resource selection phase, we have to submit them to the real databases in $\mathcal{DB}$. We assume here that the each of these queries $q_i$ returns a ranked list of scored answers, $l_i = \{\langle \bar{\mathbf{t}}^i_1, s^i_1 \rangle, \ldots, \langle \bar{\mathbf{t}}^i_{|\ell_i|}, s^i_{|\ell_i|} \rangle\}$. If no score is provided, we may assume that the score is determined by the rank of the tuple in some way, for instance, $s = (r_{\max} - r + 1)/r_{\max}$, where $r_{\max}$ is the number of returned tuples in a ranked list and $r$ is the rank of tuple $\bar{\mathbf{t}}$ in that list. Therefore, we have $s$ ranked lists $\ell_1, \ldots, \ell_s$ of tuples from which we have to build a unique list from which we select the top-$d$ tuples only. While there are many proposals to do rank merging, as illustrated at the beginning of Section 3, we plan to adapt a simple ranked list merging method, namely [11], which , despite its simplicity, is one among the most effective methods. Given then the merged list $\ell_{\texttt{norm}}$ of scored answers to query $q$, it remains then to return just the top-$d$ ones, which concludes.

## 4 Conclusions

We have preliminary discussed a framework for processing CQs on heterogeneous DWIR. We have employed techniques drawn from the field of Distributed Information Retrieval to sample DWIRs, to select a small set of rewritings to be processed on the sources, and to search the merged results. We believe that the adoption of DIR algorithms and methodologies constitutes a step forward in the automation of the integration of DWIRs. An experimental evaluation of the techniques proposed in this paper, which is currently been carried out, will show how effective our framework is. Apart from validating the proposed techniques with experiments, we plan the following future work.

1. We want to extend the language of mapping rules to be more expressive than the simple one-to-one correspondance between global and local relational entities, which in fact corresponds to a GLAV (*global-local-as-view*, which is both global-as-view and local-as-view) mapping. Our first goal would be to have a GLAV approach where mapping rules have a conjunction of atoms in the body. This will require a new technique for ranking the rewritings.
2. We plan to incorporate an *ontology* on top of the mediated schema; such an ontology will provide a further semantic layer that defines properties of the mediated data. Possible languages for such an ontology could be those in the *Datalog*$^{\pm}$ family [4], those of the *DL-lite* family [2] or simply Datalog.

*Acknowledgments.* Andrea Calì acknowledges support by the EPSRC project "Logic-based Integration and Querying of Unindexed Data" (EP/E010865/1).

## References

1. J. Arguello, J. Callan, and F. Diaz. Classification-based resource selection. In *Proc. of the 18th ACM Conf. on Information and Knowledge Management*, pages 1277–1286, 2009. ACM.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
3. A. Calì, D. Calvanese, and D. Martinenghi. Dynamic query optimization under access limitations and dependencies. *J. of Universal Computer Science*, 15(1):33–62, 2009.
4. A. Calì, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris. Datalog+/-: A Family of Logical Knowledge Representation and Query Languages for New Applications. In *Proc. of 25th Intl Symp. on Logic in Computer Science*, pages 228–242, 2010.
5. A. Cali and D. Martinenghi. Querying data under access limitations. In *Proc. of the 2008 IEEE 24th Int. Conf. on Data Engineering*, pages 50–59, 2008.
6. J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.
7. J. Caverlee, L. Liu, and J. Bae. Distributed query sampling: A quality-conscious approach. In *Proc. of the 29th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 340–347, 2006.
8. P. Harrington. *Machine Learning in Action*. Manning, 2012.

9. M. Lenzerini. Data integration: a theoretical perspective. In *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS-02)*, pages 233–246, 2002.
10. C. Li and E. Chang. Query planning with limited source capabilities. In *Proc. of ICDE*, pages 401–412, 2000.
11. I. Markov, A. Arampatzis, and F. Crestani. On CORI results merging. In *Proc. of the 35th European Conf. on Advances in Information Retrieval*, pages 752–755, 2013. Springer-Verlag.
12. H. Nottelmann and U. Straccia. A probabilistic, logic-based framework for automated web directory alignment. In Zongmin Ma, editor, *Soft Computing in Ontologies and the Semantic Web*, volume 204 of *Studies in Fuzziness and Soft Computing*, pages 47–77. Springer Verlag, 2006.
13. H. Nottelmann and U. Straccia. Information retrieval and machine learning for probabilistic schema matching. *Information Processing & Management*, 43:552–576, 2007.
14. M. E. Renda and U. Straccia. Web metasearch: Rank vs. score based rank aggregation methods. In *Proc. of the 18th Annual ACM Symposium on Applied Computing*, pages 841–846, 2003.
15. M. Shokouhi and L. Si. Federated search. *Found. Trends Inf. Retr.*, 5(1):1–102, 2011.
16. M. Shokouhi and J. Zobel. Robust result merging using sample-based score estimates. *ACM Trans. Inf. Syst.*, 27(3):14:1–14:29, 2009.
17. L. Si and J. Callan. Unified utility maximization framework for resource selection. In *Proc. of the Thirteenth ACM Int. Conf. on Information and Knowledge Management*, pages 32–41, 2004.
18. U. Straccia. Top-$k$ retrieval for ontology mediated access to relational databases. *Information Sciences*, 198:1–23, 2012.
19. U. Straccia. On the top-k retrieval problem for ontology-based access to databases. In Sławomir Pivert, Olivier; Zadrożny, editor, *Flexible Approaches in Data, Information and Knowledge Management*, volume 497 of *Studies in Computational Intelligence*, chapter 5, pages 95–114. Springer Verlag, 2014.
20. U. Straccia and N. Madrid. A top-k query answering procedure for fuzzy logic programming. *Fuzzy Sets and Systems*, 205:1–29, 2012.
21. U. Straccia and R. Troncy. Towards distributed information retrieval in the semantic web: Query reformulation using the omap framework. In *3rd European Semantic Web Conf.*, volume 4011 of *Lecture Notes in Computer Science*, pages 378–392. Springer Verlag, 2006.
22. P. Thomas. To what problem is distributed information retrieval the solution? *J. Am. Soc. Inf. Sci. Technol.*, 63(7):1471–1476, 2012.
23. P. Thomas and D. Hawking. Server selection methods in personal metasearch: A comparative empirical study. *Inf. Retr.*, 12(5):581–604, 2009.
24. C. Yu, K.-L. Liu, W. Meng, Z. Wu, and N. Rishe. A methodology to retrieve text documents from multiple databases. *IEEE Trans. on Knowl. and Data Eng.*, 14(6):1347–1361, 2002.
25. A. Zimmermann, N. Lopes, A. Polleres, and U. Straccia. A general framework for representing, reasoning and querying with annotated semantic web data. *J. of Web Semantics*, 11:72–95, 2012.