Accepted Manuscript

Enhancing answer completeness of SPARQL queries via crowdsourcing

Maribel Acosta, Elena Simperl, Fabian Flöck, Maria-Esther Vidal

PII: S1570-8268(17)30030-6
DOI: http://dx.doi.org/10.1016/j.websem.2017.07.001
Reference: WEBSEM 440

To appear in: *Web Semantics: Science, Services and Agents on the World Wide Web*

Received date : 25 August 2016
Revised date : 2 June 2017
Accepted date : 7 July 2017

Please cite this article as: M. Acosta, E. Simperl, F. Flöck, M. Vidal, Enhancing answer completeness of SPARQL queries via crowdsourcing, *Web Semantics: Science, Services and Agents on the World Wide Web* (2017), http://dx.doi.org/10.1016/j.websem.2017.07.001

Enhancing Answer Completeness of SPARQL Queries via Crowdsourcing

Maribel Acosta[a,*], Elena Simperl[b], Fabian Flöck[c], Maria-Esther Vidal[d,1]

[a] *Institute AIFB, Karlsruhe Institute of Technology, Germany*
[b] *University of Southampton, United Kingdom*
[c] *GESIS - Leibniz Institute for the Social Sciences, Germany*
[d] *Computer Science Department, Universidad Simón Bolívar, Venezuela*

Abstract

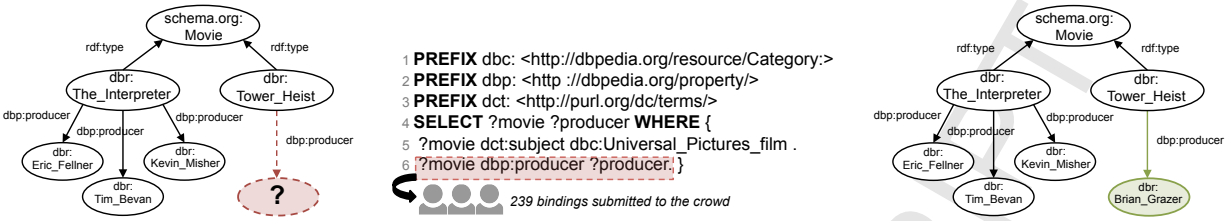
Linked Open Data initiatives have encouraged the publication of large RDF datasets into the Linking Open Data (LOD) cloud, including DBpedia, YAGO, and Geo-Names. Despite the size of LOD datasets and the development of (semi-)automatic methods to create and link LOD data, these datasets may be still incomplete, negatively affecting thus accuracy of Linked Data processing techniques. We acquire query answer completeness by capturing knowledge collected from the crowd, and propose a novel hybrid query processing engine that brings together machine and human computation to execute SPARQL queries. Our system, HARE, implements these hybrid query processing techniques. HARE encompasses several features: (1) a completeness model for RDF that exploits the characteristics of RDF in order to estimate the completeness of an RDF dataset; (2) a crowd knowledge base that captures crowd answers about missing values in the RDF dataset; (3) a query engine that combines on-the-fly crowd knowledge and estimates provided by the RDF completeness model, to decide upon the sub-queries of a SPARQL query that should be executed against the dataset or via crowd computing to enhance query answer completeness; and (4) a microtask manager that exploits the semantics encoded in the dataset RDF properties, to crowdsource SPARQL subqueries as microtasks and update the crowd knowledge base with the results from the crowd. Effectiveness and efficiency of HARE are empirically studied on a collection of 50 SPARQL queries against the DBpedia dataset. Experimental results clearly show that our solution accurately enhances answer completeness.

Keywords: RDF Data, Crowd Knowledge, Query Execution, Crowdsourcing, Hybrid System, Microtasks, Completeness Model, SPARQL Query

1. Introduction

Following the Linked Data principles[2], Semantic Web technologies facilitate the integration and publication of open data into the Linking Open Data (LOD) cloud. During the last decade, the LOD cloud has grown considerably, passing from comprising nine datasets in 2007 to more than 1,000 in 2014 [29]. The Resource Description Framework[3] (RDF) and Semantic Web tools are used to describe and publicly make available data in the LOD cloud. RDF is a semi-structured data model where entities are represented as resources; connections between resources are described as triples composed of subjects, predicates, and objects. RDF triples represent positive statements, i.e., negative statements cannot be modeled. Further, the *open world assumption* is assumed for RDF triples, e.g., because RDF datasets may be incomplete, a movie can be associated with producers even if no triples represent this statement in an RDF dataset. Additionally, class hierarchies in ontologies can be used to describe the types of the resources, and resources of the same class can be characterized by different set of properties. For example, in the DBpedia dataset [18], the resource dbr:The_Interpreter is typed as

*Corresponding author.
Email addresses: maribel.acosta@kit.edu (Maribel Acosta), e.simperl@soton.ac.uk (Elena Simperl), fabian.floeck@gesis.org (Fabian Flöck), mvidal@ldc.usb.ve (Maria-Esther Vidal)
[1] On sabbatical leave at the AIFB, Karlsruhe Institute of Technology
[2] http://www.w3.org/DesignIssues/LinkedData
[3] https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/



(a) The movies dbr:The_Interpreter is linked to three producers via the dbp:producer predicate, contrary to dbr:Tower_Heist whose producer values are missing in DBpedia.

(b) Query to *select movies (and producers) filmed by Universal Pictures*. This query produces 1,222 results, yet, there are 1,461 movies by Universal Pictures. 239 movies are not associated with a producer in DBpedia (e.g., dbr:Tower_Heist).

(c) The crowd stated that *Brian Grazer is a producer of Tower Heist*. Hence, a value for the property dbp:producer of dbr:Tower_Heist is dbr:Brian_Grazer.

Figure 1: Motivating example. (a) Portion of DBpedia for movies and producers. Missing values in the RDF graph are highlighted. (b) SPARQL query executed against DBpedia. Portions of the query (highlighted) affected by missing values are crowdsourced. (c) Crowd answers are mapped into RDF to augment the result of queries.

schema.org:Movie and linked to three producers via the dbp:producer property, while the resource dbr:Tower_Heist does not have this property, as shown in Figure 1a.

As in traditional semi-structured data models, the semi-structured nature of RDF allows for creating datasets that result from integrating multiple, and typically heterogenous and unstructured data sources. However, RDF datasets may lack of explicit meta-data and, if exists, this may be incomplete. Furthermore, although RDF data can be correct, a large number of missing values may occur, thus negatively impacting completeness of tasks of Linked Data consumption and query processing. To illustrate, let us consider a query that selects movies, including their producers, that have been filmed by Universal Pictures. Such query can be formulated in SPARQL as in Figure 1b and the executed against DBpedia. The query execution returns no producers for 239 out of the 1,461 movies filmed by Universal Pictures. An inspection to the query results reveals that DBpedia has no producers for dbr:Tower_Heist, however, this movie has actually three producers. This is an example of *missing values*. With cases like this being a common occurrence in RDF datasets, further techniques are needed to improve data quality in terms of completeness and subsequent query processing results.

The Database and Semantic Web communities have extensively studied methods for assuring data quality in traditional databases [23] as well as on Web data [2, 13, 35]. Despite all these developments, common sense knowledge acquired from humans may be required for improving effectiveness of automatic methods of data quality assessment [6, 10, 13, 26]. In the context of data management, crowdsourcing have been used to design advanced query processing systems that combine human and computational intelligence [15, 21, 25]. Albeit effective for relational databases, such approaches are less feasible for a Linked Data scenario, which is confronted with autonomous RDF datasets. We overcome limitations of crowd-based solutions for relational query processing, and tackle the problem of *automatically identifying portions of a SPARQL query against an RDF dataset that yield incomplete results*; missing values are assessed via microtask crowdsourcing. Tackling this problem requires query evaluation techniques against RDF datasets able to preserve the formal properties of SPARQL query execution as established in the Evaluation problem [27, 30]. In addition, resorting to the crowd to assess RDF data demands strategies to collect reliable answers from human contributors efficiently. Therefore, in this work, we investigate the following research questions:

RQ1 Can answers of SPARQL queries be completed via hybrid computation without incurring additional complexity in query evaluation?

RQ2 Can answer completeness of SPARQL queries be augmented via microtasks?

RQ3 What is the impact of exploiting the semantics of RDF resources on crowd effectiveness and efficiency when solving missing values?

We propose HARE, a hybrid query processing system that combines human and computational capabilities to run queries against RDF datasets. HARE aims at enhancing answer completeness of SPARQL queries by resolving missing values in datasets via microtask crowdsourcing. Following our running example, HARE is able to crowdsource

portions of SPARQL queries, as in Figure 1b, and complete missing values in RDF datasets with the answers from the crowd as depicted in Figure 1c. To detect missing values, HARE relies on the Local Closed-World Assumption (LCWA) which assumes that parts of the dataset are complete. HARE provides a highly flexible crowdsourcing-enabled SPARQL query execution: No extensions to SPARQL are required, and users can configure the level of expected answer completeness in each query execution (denoted τ). In recent work [4], we develop a simple model able to enhance the completeness of answers of SPARQL queries that access objects in RDF triples. In this work, we generalize our prior approach, and propose an RDF completeness model able to estimate the completeness of the RDF resources that play any role in RDF triples. This, in turn, allows for improving answer completeness in a broader range of SPARQL queries. For instance, in the query movies that have been filmed in New York City by Universal Pictures and produced by Brian Grazer, movies correspond to subjects in DBpedia triples. Based on the new RDF completeness model, HARE may decide to resort to the crowd for enriching DBpedia on movies and on producers to deliver higher quality results.

Furthermore, HARE encompasses a knowledge base that captures the knowledge collected from the crowd, which is opportunistically exploited to discern whether the crowd is likely to solve a question accurately. Additionally, we propose a SPARQL query engine able to efficiently combine crowd answers and intermediate SPARQL results, and produce fuzzy mappings of the variables in the SPARQL queries. To formalize the fuzzy semantics of SPARQL queries, a fuzzy SPARQL algebra comprises part of the contributions of this work. Finally, the HARE microtask manager includes the user interface generator which is able to exploit the semantics of RDF resources to build human-readable interfaces that facilitate the collection of right answers from the crowd.

The quality of the HARE hybrid query processing techniques has been empirically evaluated in a crafted collection of 50 SPARQL queries against DBpedia (version 2014). The goal of the experiments is to analyze the performance of HARE when queries are executed directly against the dataset and the crowdsourcing platform CrowdFlower [1]. Empirical results clearly show that HARE can reliably augment response completeness while crowd answers achieved accuracy values from 0.84 to 0.96. Furthermore, the majority of the query answers are produced in reasonable time via crowdsourcing, i.e., in all the studied queries at least 75% of the answers are collected 12 minutes after the first task was submitted to the crowd. Finally, the semantically enriched microtasks produced by HARE are able to provide assistance to the crowd, and speed up the process of crowd answering by at least one order of magnitude with respect to tasks built without semantics. These results confirm that the incorporation of crowd knowledge into query processing techniques can effectively augment the completeness of SPARQL query answers.

This paper is an extension to previous work of ours [4], where we presented a hybrid query engine to enhance answers of SPARQL queries – with variables only in the object position – using crowdsourcing. The novel contributions of our current work are summarized as follows:

- An extended RDF completeness model that now fully exploits the topology of RDF graphs to estimate the answer completeness of SPARQL queries with variables in the subject, predicate, or object position.
- The definition of contradiction between statements provided by the crowd has been extended and we define a new metric for measuring this contradiction.
- A formal definition of the operations carried out by the proposed microtask manager.
- The algorithm of the HARE query optimizer is now defined.
- A fuzzy set semantics of the SPARQL query language; we formally prove the complexity of computing the solution of SPARQL queries under the proposed semantics.
- The query engine has been re-defined to be able to evaluate SPARQL queries respecting the proposed SPARQL fuzzy set semantics.
- An extensive empirical evaluation that demonstrates the impact of HARE on the efficiency and effectiveness of the crowd.

The rest of the paper is structured as follows. Section 2 presents an analysis of the related work while Section 3 presents the fundamentals of the RDF data model and the SPARQL query language. Section 4 formalizes the problem solved by HARE

and describes the main components of the HARE architecture. The HARE completeness model is defined in Section 5, and the representation of the crowd knowledge is presented in Section 6. Section 7 describes the HARE microtask manager and Section 8 presents the query optimizer. A proposed fuzzy semantics of SPARQL and the query engine are defined in and Section 9. Experimental results are reported in Section 10, and we conclude in Section 11 with an outlook to future work.

2. Related Work

2.1. Hybrid Query Processing for Relational Data

The database community has proposed several human/computer query processing architectures for relational data. Approaches such as CrowdDB [15], Deco [24, 25], Qurk [19], and CrowdOp [14] target scenarios in which existing microtask platforms are directly embedded in query processing systems. These systems provide declarative languages tailored to facilitate a highly adaptive design of hybrid query execution pipelines.

CrowdDB [15] introduces SQL-like data definition and query languages to specify tables, columns, or operators that are subject to crowdsourcing. Similarly, Deco [24, 25] is a declarative approach that allows for the specification of fetch rules to indicate how data can be obtained from humans, and resolution rules to specify how conflicts in crowdsourced data are solved. Qurk [21] defines a specification language to describe microtasks in terms of type of question, input, and output. Furthermore, Qurk is able to generate query plans that combines both relational tables and crowd tasks to reduce the number of tasks submitted to the crowd [20]. CrowdOp [14] relies on cost-based query optimization to generate query plans that efficiently gather unknown values in relational tables from the crowd.

These relational engines require database administrators or crowd-based workflow designers to specify what to crowdsource during query execution time. The focus is mainly on the architectural and formalism design, as well as on the efficient implementation of the actual crowdsourcing components, assuming that specific classes of queries (e.g., subjective comparisons) specified at design time will *always* be outsourced to the crowd. One important challenge is reducing the number of tasks posed to the crowd, since the delivery time of the crowd increases whenever tasks compete for the attention of the same workers. Approaches such as Deco [24] have tackled this issue by proposing caching strategies, while CrowdDB [15] attempts to reduce the number of tasks to be outsourced by taking advantage of structural properties in the relational data.

To conclude, the studies of crowdsourcing-enabled relational databases provide evidence about how specific design parameters of microtasks influence the performance of queries executed with hybrid systems. However, these insights cannot be directly transferred to the Web of Data due to several reasons: *i*) SPARQL queries may span over a large number of statements and even several datasets. It is therefore unrealistic to expect a SPARQL engine designer to specify rules for queries that would trigger a crowdsourcing task. *ii*) The semi-structured nature of Web data makes it very hard to assess the quality of datasets upfront and to identify subgraphs which should be subject to crowdsourced curation (e.g., missing or incorrect values). *iii*) Queries over Web accessible RDF datasets (e.g., via SPARQL endpoints) are typically posed by users autonomously and – contrary to relational crowdsourcing scenarios – precisely determining at design time the attributes that required to be crowdsourced is not possible.

HARE takes the lessons learned in crowd-based relational databases and applies them to a scenario that exhibits formally different characteristics regarding the ways data is produced and consumed. First, Linked Data sets are assumed to be correct but potentially incomplete, and crowd knowledge is exploited to enhance query completeness and enrich Linked Data sets. In HARE, crowd answers are captured in crowd knowledge bases and the RDF incompleteness model is used to devise optimization strategies for effective query execution. HARE optimization techniques make sure that human contributions are sought only in those cases in which it will most likely lead to result improvements, hence, reducing both the overall costs and the average time needed to collect crowd answers. Additionally, HARE leverages the semantics encoded in RDF datasets and their ontologies to generate microtask interfaces tailored for types or classes of the data that will be collected from the crowd. Overall, although HARE implements a hybrid human/computer query processing architecture, it differs from crowd-based relational databases in the ability of exploiting the RDF model, semantics of the data, and the wisdom of the crowd to acquire the microtasks that will allow for augmenting query answer and Linked Data enrichment.

2.2. Crowd-based Linked Data Management

Crowdsourcing has also been applied in other areas of data management. Demartini et al. [11, 12] propose ZenCrowd, a hybrid approach that relies on paid crowdsourcing for matching linked datasets and linking collections of Web pages to the Linking Open Data cloud. ZenCrowd implements a probabilistic framework to identify candidate datasets and suitable crowd workers, and applies crowdsourcing to a corpus of news articles to suggest new links. Additionally, ZenCrowd is able to link two instances of different schemas or ontologies; thus, automatic entity extraction and linking is enriched and enhanced with crowd knowledge. ZenCrowd model takes advantage of probabilistic networks to gather evidences collected from algorithmic linkers and the crowd to produce confidence scores of the predicted matches. Similarly, CrowdMAP [28] tackles the problem of ontology matching, and reports on the evaluation of existing alignment algorithms and how precision and recall can be enhanced using crowd labor. HARE also resorts to paid crowdsourcing for hybrid computation but with a different purpose. Instead of matching instances and combining evidences from linker algorithms and the crowd, HARE depends on estimates derived from the RDF completeness model and crowd knowledge to decide completeness of Linked Data sets. HARE solves this decision problem during query processing time, and at the level of RDF triples obtained with the evaluation of triple patterns in SPARQL queries.

OASSIS [7] is a recommendation system that mines frequent patterns from personal data collected via crowdsourcing. Patterns to mine are specified in OASSIS-QL, a SPARQL-like language. OASSIS exploits general knowledge from ontological concepts to reason over the data from the crowd in order to reduce the number of subsequent crowdsourced questions needed to discover a new pattern. The problem of determining the number of questions to be sent to the crowd has been also studied by Mozafari et al. [22] and Trushkowsky et al. [33]. Mozafari et al. propose machine learning algorithms that rely on the bootstrap theory to precisely estimate uncertainty scores of labels that will be requested from the crowd in one or different batches. The approach is general enough to be treated as a black-box and adapted to solve the optimization task of deciding when to stop asking in different crowd-based problems, e.g., entity resolution, image search, or sentiment analysis. Trushkowsky et al. present a statistical model based on sampling techniques to estimate the cardinality of crowd answers. Both solutions are tailored to decide the stoping point of microtasks with a large number of answers; however, appropriate training datasets or sample populations have to be crafted to generate robust estimates. HARE also tackles this decision problem, but implements a lightweight model that does not require training data or sample populations. In contrast to the work by Mozafari et al. and Trushkowsky et al., HARE utilizes knowledge collected from the crowd and the RDF completeness model to estimate an upper bound on the number of iterations the same question will be sent to the crowd.

2.3. Web Data Quality Assessment

Crowdsourcing techniques have been also applied to deal with data quality problems such as completeness and correctness. KATARA [10] is a system to cleanse tabular data by using a combination of RDF knowledge bases (KBs) and crowdsourcing; tabular data may be incorrect while KBs are assumed to be correct but may be incomplete. KATARA discovers patterns that align table columns with ontological definitions in KBs, identifying types and relationships of the columns. Patterns are then validated via crowdsourcing; correct patterns are used to generate possible repairs for data entries in tables, and to potentially complete data in the reference KBs. HARE also assumes KBs are correct but potentially incomplete, and implements query processing strategies that rely on crowd knowledge and the RDF completeness model to augment query completeness. HARE makes use of the enhanced answers to enrich the KBs; any type of RDF triples can be added to the KBs. Contrary, KATARA is limited to the data stored in the tabular datasets, and RDF triples of the form (s, p, o) can only be added to the KBs, whenever s and o appear in the tabular dataset.

Acosta et al. [6] also tackle the data quality assessment problem and propose a human-based workflow to *detect* quality issues in RDF graphs. The proposed workflow is a variant of the Find-Fix-Verify crowdsourcing pattern [8] to effectively combine knowledge from experts and crowd workers to determine if an RDF triple is potentially incorrect; missing values cannot be detected. HARE also resorts to the crowd to assess quality issues in KBs. However, HARE assumes that KBs are correct but

potentially incomplete, and exploits an RDF completeness model and crowd knowledge bases not only to *decide* incompleteness, but also to *acquire* a hybrid processing task to *enhance* query answer and KB completeness.

Finally, the problem of automatically constructing knowledge bases has been addressed by Dong et al. [13]. Dong et al. propose unsupervised strategies for both resolving conflicts from knowledge extracted from different data sources and finding the correct values. Knowledge in the integrated KB is represented as RDF triples of the form (s, p, o). This approach assumes that values of s, p are already known in a gold standard knowledge base, e.g., *Freebase*, and the goal is to identify correct values of o. If an RDF triple (s, p, o) occurs in the gold standard knowledge base, then the triple is considered correct. However, the decision of incorrectness is made based on the Local Closed-World Assumption (LCWA) which assumes that the gold standard knowledge base is locally complete. Thus, if (s, p, o) does not occur in the gold standard knowledge base, but there is at least one triple (s, p, o_1) with o_1 different from o, then (s, p, o) is assumed incorrect. Contrary, if there is no such triple sharing the same subject and predicate in the gold standard, the original triple (s, p, o) is excluded and not classified. HARE assumes that the KB is correct but possibly incomplete, and applies LCWA differently. The number of different values of o in triples of the form (s, p, o) as well as the knowledge about the types of the resource s in the dataset are used to estimate the completeness of an RDF resource with respect to p. Estimates of completeness are exploited by the HARE query engine to decide if the answer of a triple pattern is incomplete. Crowd knowledge is then acquired to enhance query completeness, as well as to potentially enrich the KB.

3. Preliminaries: RDF and SPARQL

According to the LD principles, data published on the Web should be described using the Resource Description Framework (RDF). RDF is a graph-based data model, where nodes in the graph can be linked via directed labeled edges. Each pair of connected nodes is called an *RDF triple*. RDF nodes correspond to *resources* or *literals* (strings). Resources can be either identified by a Universal Resource Identifier (URI) or unidentified, denominated *blank nodes*, which model existential variables in the graph. RDF graphs are also denominated *RDF datasets*. We follow the notation from Pérez et al. [27] and Schmidt et al. [30], and present the formal definition of RDF triples and graphs.

Definition 1 (RDF Triple and Dataset [27]) *Let U, B, L be disjoint infinite sets of URIs, blank nodes, and literals, respectively. A tuple $(s, p, o) \in (U \cup B) \times (U) \times (U \cup B \cup L)$ is denominated an RDF triple, where s is called the subject, p the predicate, and o the object. An RDF dataset or RDF graph is a set of RDF triples. When $s \in L$ or $p \in (B \cup L)$, then the tuple (s, p, o) is called a generalized RDF triple and the dataset where it contained is called a generalized RDF dataset [17].*

The recommended language for querying RDF data is SPARQL. In this work, we focus on SPARQL SELECT queries which return the set of variables and their mapping results. SELECT queries and SPARQL expressions are defined in the following.

Definition 2 (SPARQL Expression, Select Query [30]) *Let V be a set of variables disjoint from $U \cup B \cup L$. A SPARQL expression is built recursively as follows. (1) A triple pattern $t \in (U \cup V) \times (U \cup V) \times (L \cup U \cup V)$ is an expression. (2) If Q_1, Q_2 are expressions and R is a filter condition, then Q_1 FILTER R, Q_1 UNION Q_2, Q_1 OPT Q_2, Q_1 AND Q_2 are expressions. Let Q be a SPARQL expression and $S \subset V$ a finite set of variables. A SPARQL SELECT query is an expression of the form* $\text{SELECT}_S(Q)$.

The evaluation of SPARQL queries over an RDF dataset is based on mappings. Each mapping represents a possible answer for a given SPARQL expression or query.

Definition 3 (SPARQL Mappings [30]) *A mapping is a partial function $\mu : V \to (B \cup L \cup U)$ from a subset of variables to RDF terms. The domain of a mapping μ, $dom(\mu)$, is the subset of V for which μ is defined. Two mappings μ_1, μ_2 are compatible, written $\mu_1 \sim \mu_2$, if $\mu_1(x) = \mu_2(x)$ for all $x \in dom(\mu_1) \cap dom(\mu_2)$. Further, $vars(t)$ denotes all variables in triple pattern t, and $\mu(t)$ is the triple pattern obtained when replacing all $x \in dom(\mu) \cap vars(t)$ in t by $\mu(x)$.*

We focus on SPARQL query evaluation under *set semantics* [27]. The semantics of SPARQL query evaluation is defined by an algebra over the previously defined mappings.

Definition 4 (SPARQL Algebra [27, 30]) *Let Ω, Ω_l, Ω_r be mapping sets, R denotes a filter condition, and $S \subset V$ a finite set of variables. The expression of SPARQL algebraic operations are defined as follows:*

$$
\begin{array}{lcl}
\Omega_l \bowtie \Omega_r & := & \{\mu_l \cup \mu_r \mid \mu_l \in \Omega_l, \mu_r \in \Omega_r : \mu_l \sim \mu_r\} \\
\Omega_l \cup \Omega_r & := & \{\mu \mid \mu \in \Omega_l \text{ or } \mu \in \Omega_r\} \\
\Omega_l \setminus \Omega_r & := & \{\mu_l \in \Omega_l \mid \text{for all } \mu_r \in \Omega_r : \mu_l \nsim \mu_r\} \\
\Omega_l \mathbin{⟕} \Omega_r & := & (\Omega_l \bowtie \Omega_r) \cup (\Omega_l \setminus \Omega_r) \\
\pi_S(\Omega) & := & \{\mu_1 \mid \exists \mu_2 : \mu_1 \cup \mu_2 \in \Omega \wedge \\
 & & \quad dom(\mu_1) \subseteq S \ \wedge \ dom(\mu_2) \cap S = \emptyset\} \\
\sigma_R(\Omega) & := & \{\mu \in \Omega \mid \mu \models R\}
\end{array}
$$

Where $\models$ refers to built-in boolean functions defined by Pérez et al. [27].

The result of evaluating SPARQL queries over RDF datasets is a function that translates queries and expressions into algebraic SPARQL operations [30]. This is formally defined as follows.

Definition 5 (SPARQL Semantics [27, 30]) *Let D be an RDF dataset, t a triple pattern, and Q, Q_1, Q_2 SPARQL expressions, R a filter condition, and $S \subset V$ a finite set of variables. Let $[[\cdot]]_D$ be a function that translates SPARQL expressions into SPARQL algebra operators as follows:*

$$
\begin{array}{lcl}
[[t]]_D := \{\mu \mid dom(\mu) = vars(t) \text{ and } \mu(t) \in D\} & & \\
[[Q_1 \text{ AND } Q_2]]_D & := & [[Q_1]]_D \bowtie [[Q_2]]_D \\
[[Q_1 \text{ OPT } Q_2]]_D & := & [[Q_1]]_D \mathbin{⟕} [[Q_2]]_D \\
[[Q_1 \text{ UNION } Q_2]]_D & := & [[Q_1]]_D \cup [[Q_2]]_D \\
[[Q \text{ FILTER } R]]_D & := & \sigma_R([[Q]]_D) \\
[[\text{SELECT}_S(Q)]]_D & := & \pi_S([[Q]]_D)
\end{array}
$$

Furthermore, a Basic Graph Pattern (BGP) is a sequence of triple patterns and filter expressions that are combined with ANDs. All SPARQL expressions are called graph patterns. In the remainder of this paper, we will refer to the expression $\text{SELECT}_S(Q)$ as *SPARQL query Q*.

To analyze the complexity of SPARQL query evaluation, the associated decision problem EVALUATION is defined as follows: Given a mapping μ, an RDF dataset D, and a SPARQL expression or query Q as input: is $\mu \in [[Q]]_D$?.

Theorem 1 ([27, 30]) *The* EVALUATION *problem is in (1)* PTIME *for expressions constructed using only* AND *and* FILTER*; (2)* NP*-complete for expressions constructed using* AND*,* FILTER*, and* UNION *operators; (3)* PSPACE*-complete for graph pattern expressions.*

The definition of EVALUATION is based on set semantics for SPARQL query evaluation.

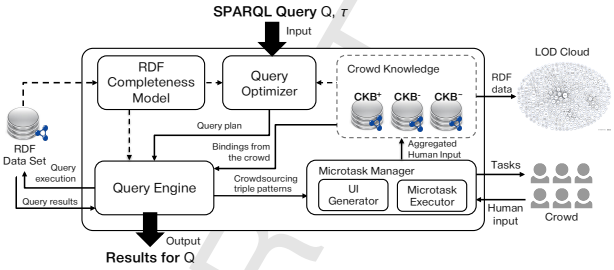


Figure 2: The HARE architecture. The input is a SPARQL query Q and a quality threshold τ. The query optimizer and engine detects portions of Q that might yield incomplete results. HARE query engine combines intermediate results from the dataset with values provided by the crowd to augment the answer of Q. Potential missing values are crowdsourced by the microtask manager. Human input is stored as RDF data in the crowd knowledge bases CKB^+, CKB^-, $CKB^{\sim}$.

4. Problem Definition and Our Approach

Given an RDF dataset D and a SPARQL query Q to be evaluated over D, i.e., $[[Q]]_D$. Consider D^* the virtual dataset that contains all the triples that *should be* in D. The problem of *identifying portions of Q that yield missing values* is defined as follows. For all Basic Graph Pattern (BGP) $B = \{t_1, t_2, ..., t_n\}$ in Q, where t_i is a triple pattern, identify the greatest subset $P \in 2^B$ such that:

$$[[P]]_D \subset [[P]]_{D^*} \tag{1}$$

Once P has been identified, the problem of *resolving the missing values* to enhance the final answer of Q consists on creating mappings μ such that:

$$\mu \notin [[P]]_D \ \wedge \ \mu \in [[P]]_{D^*} \tag{2}$$

We propose HARE, a query engine that automatically identifies portions of a SPARQL query that might yield incomplete results and resolves them via crowdsourcing. Figure 2 depicts the components of HARE, which receives as input a SPARQL

query Q and a quality threshold τ. The *RDF Completeness Model* estimates the completeness of portions of a dataset. The *Query Optimizer* generates a plan from Q, executed by the *Query Engine*. The engine takes into consideration τ, the completeness model, and RDF triples collected from the crowd to tackle the first problem presented in Equation (1). Potential missing values are passed to the *Microtask Manager*, which contacts the crowd to generate the mappings μ to tackle the second problem presented in Equation (2). The HARE engine efficiently combines results retrieved from the dataset with human input to produce the final results for Q.

5. RDF Completeness Model

We propose a model to estimate the completeness of portions of RDF datasets. The intuition behind our model is to capture the number of different subjects, predicates, and objects in RDF triples, i.e., the *multiplicity of RDF resources*. Then, the model compares the multiplicity of resources with the *aggregated multiplicity of classes* in the dataset, i.e., the multiplicity of all resources that belong to the same class. In the following, we define the multiplicity of RDF resources. We say that a resource r *occurs* in dataset D if exists an RDF triple in D where r is either the subject, predicate, or object.

Definition 6 *Let s, p, o be RDF resources. The **multiplicity of RDF resources** in a dataset D is defined as the number of subjects ($MS_D(o|p)$), objects ($MO_D(s|p)$), and predicates ($MP_D(s|o)$) that appear in RDF triples (s,p,o) in D as follows:*

$$\begin{aligned} MS_D(o|p) &:= |\{s \mid (s,p,o) \in D\}| \\ MO_D(s|p) &:= |\{o \mid (s,p,o) \in D\}| \\ MP_D(s|o) &:= |\{p \mid (s,p,o) \in D\}| \end{aligned}$$

Example 1 *Consider the RDF graph D depicted in Figure 3 which contains four nodes of type* schema.org:Movie. *In this figure, movies are linked to their producers via the* dbp:producer *predicate. In this example, the multiplicity is computed for all the nodes of type movies and their producers. For instance, the resource $s =$* dbr:Legal_Eagles *has two values for the predicate $p =$* dbp:producer*, therefore, $MO_D(s|p)$ is 2 in this case. The non-zero values for MS_D, MO_D, and MP_D for movies and producers in the dataset D are as follows:*

$$\begin{aligned} &MS_D(\text{dbr:Sheldon\_Kahn} \mid \text{dbp:producer}) = 1 \\ &MS_D(\text{“Ivan\_Reitmann”} \mid \text{dbp:producer}) = 1 \\ &MS_D(\text{“Kris\_Thykier”} \mid \text{dbp:producer}) = 1 \\ &MS_D(\text{dbr:Eric\_Fellner} \mid \text{dbp:producer}) = 2 \\ &MS_D(\text{dbr:Tim\_Bevan} \mid \text{dbp:producer}) = 2 \\ &MS_D(\text{dbr:Kevin\_Misher} \mid \text{dbp:producer}) = 1 \\ \\ &MO_D(\text{dbr:Legal\_Eagles} \mid \text{dbp:producer}) = 2 \\ &MO_D(\text{dbr:Trash\_(2014\_film)} \mid \text{dbp:producer}) = 3 \\ &MO_D(\text{dbr:The\_Interpreter} \mid \text{dbp:producer}) = 3 \\ \\ &\text{For all } s,o \text{ that occur in } D,\ MP_D(s|o) = 1 \end{aligned}$$

Following the intuition of our model, we now look at the multiplicity of all resources that belong to the same class. Next, we define the aggregated multiplicity per subject, predicate, and object of resources that belong to a given class. We assume that sub-class relationships (specified via the rdfs:subClassOf) are materialized in D.

Definition 7 *Let C, C_1, and C_2 be classes in a dataset D. The **aggregated multiplicity of a class** is given by the multiplicity of its RDF resources: $AMS_D(C|p)$ denotes the aggregate multiplicity of subjects of class C for predicate p; $AMO_D(C|p)$ denotes the aggregate multiplicity of objects of class C for predicate p; and $AMP_D(C_1|C_2)$ denotes the aggregate multiplicity of predicates between subjects of class C_1 and objects of class C_2. The aggregated multiplicity of classes in a dataset D is defined as follows:*

$$\begin{aligned} &AMS_D(C|p) := \\ &\quad f(\{\{MS_D(o|p) \mid (s,p,o) \in D \ \wedge\ (o,a,C) \in D\}\}) \\ &AMO_D(C|p) := \\ &\quad f(\{\{MO_D(s|p) \mid (s,p,o) \in D \ \wedge\ (s,a,C) \in D\}\}) \\ &AMP_D(C_1|C_2) := \\ &\quad f(\{\{MP_D(s|o) \mid (s,p,o) \in D \ \wedge\ (s,a,C_1) \in D \wedge \\ &\qquad\qquad (o,a,C_2) \in D\}\}) \end{aligned}$$

Where:

- *(r,a,C) corresponds to the triple (r, rdf:type, C), which states that the resource r belongs to the class C,*
- *$f(.)$ is an aggregation function.*

The aggregation function f in Definition 7 determines how the multiplicity of individual RDF resources is combined. Given that the multiplicity

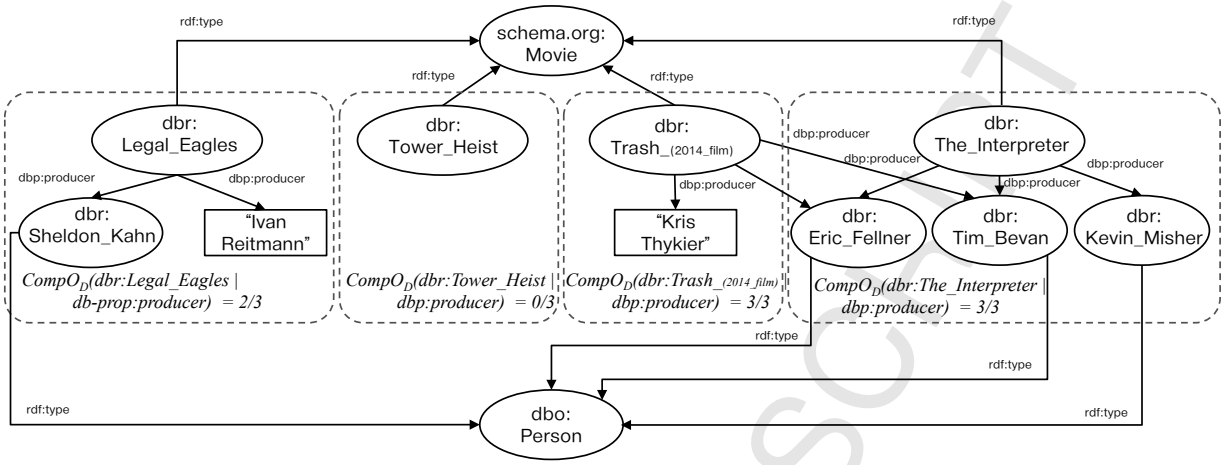


Figure 3: Portion of DBpedia for movies. schema.org:Movie and dbo:Person are classes. The resources dbr:Legal_Eagles, dbr:Tower_Heist, dbr:Trash_(2014_film), and dbr:The_Interpreter are instances of the schema.org:Movie class. Movies are linked to producers via the dbp:producer predicate. Movies are annotated with the object completeness $CompO_D$ value for the dbp:producer predicate, e.g., $CompO_D$ for db:Legal_Eagles is 2/3 since this movie has two producers, and AMO_D for the class schema.org:Movie is three. The object completeness of producers for the resources dbr:Trash_(2014_film) and dbr:The_Interpreter is 3/3. The movie dbr:Tower_Heist has no producers, thus $CompO_D = 0$.

of resources in RDF datasets may exhibit a skewed distribution, in our approach f corresponds to the median. By choosing the median for the computation of f, outliers do not affect the estimation of the aggregated multiplicity of classes.

Example 2 *Suppose the class schema.org:Movie comprises only the four movies in Figure 3, and the aggregation function f is the median. The aggregated object multiplicity of the class schema.org:Movie with respect to the predicate dbp:producer, i.e., AMO_D(schema.org:Movie | dbp:producer), is computed over the values of MO_D from Example 1 as median($\{\{2,3,3\}\}$), which results in a value of 3. The non-zero values for AMS_D, AMO_D, and AMP_D for the classes schema.org:Movie and dbo:People in the dataset D from Figure 3 are as follows:*

$$\begin{aligned} AMS_D(\text{dbo:Person} \mid \text{dbp:producer}) &= 1 \\ AMO_D(\text{schema.org:Movie} \mid \text{dbp:producer}) &= 3 \\ AMP_D(\text{schema.org:Movie} \mid \text{dbo:Person}) &= 1 \end{aligned}$$

The completeness of an RDF resource is given by the multiplicity of the resource and the aggregated multiplicity of all classes where the resource belongs to. The class with the highest multiplicity determines how complete the resource is.

Definition 8 *Let s, p, and o be RDF resources, with $(s,a,C_{s1}) \in D, ..., (s,a,C_{sn}) \in D$ and $(o,a,C_{o1}) \in D, ..., (o,a,C_{om}) \in D$. The **completeness of RDF resources** is given by the multiplicity of RDF resources and the classes that they belong to. $CompS_D(o|p)$ denotes the completeness of subjects in D for resource o via the predicate p; $CompO_D(s|p)$ denotes the completeness of objects in D for resource s via the predicate p; and $CompP_D(s|o)$ denotes the completeness of predicates in D that link resources s and o. The completeness of RDF resources ($[0.0; 1.0]$) in a dataset D is defined as follows:*

$$CompS_D(o|p) := \begin{cases} \frac{MS_D(o|p)}{AMS'_D} & AMS'_D{\neq}0 \ \wedge \ MS_D{<}AMS'_D \\ 1.0 & otherwise \end{cases}$$

$$CompO_D(s|p) := \begin{cases} \frac{MO_D(s|p)}{AMO'_D} & AMO'_D{\neq}0 \ \wedge \ MO_D{<}AMO'_D \\ 1.0 & otherwise \end{cases}$$

$$CompP_D(s|o) := \begin{cases} \frac{MP_D(s|o)}{AMP'_D} & AMP'_D{\neq}0 \ \wedge \ MP_D{<}AMP'_D \\ 1.0 & otherwise \end{cases}$$

Where:

$$AMS'_D = max(AMS_D(C_{o1}|p), ..., AMS_D(C_{om}|p)),$$
$$AMO'_D = max(AMO_D(C_{s1}|p), ..., AMO_D(C_{sn}|p)),$$
$$AMP'_D = max(AMP_D(C_{s1}|C_{o1}), ..., AMP_D(C_{sn}|C_{om})).$$

Example 3 *Consider the RDF graph D from Figure 3. According to Definition 8, the object completeness ($CompO_D$) of the resource dbr:Legal_-*

Eagles *for the predicate* dbp:producer *is computed as* $MO_D(\text{dbr:Legal\_Eagles} \mid \text{dbp:producer})$ *(cf. Example 1) divided by* $AMO_D(\text{schema.org:Movie} \mid \text{dbp:producer})$ *(cf. Example 2), i.e.,* $\frac{2}{3} = 0.667$. *In the same way, the object completeness for the resources* dbr:Trash_(film_2014) *and* dbr:The_Interpreter *for the predicate* dbp:producer *is* $\frac{3}{3}$, *as depicted in Figure 3. Furthermore, consider that the movie* db:The_Interpreter *also belongs to the class* dbo:Film, *and the aggregated multiplicity of this class is* $AMO_D(\text{dbo:Film} \mid \text{dbp:producer}) = 5$. *Then, the object completeness* $CompO_D(\text{dbr:The\_Interpreter} \mid \text{dbp:producer})$ *is* $\frac{3}{5} = 0.6$, *estimating that two out of five producers of this movie are not represented in the dataset.*

In general, completeness values $CompS_D$, $CompO_D$, and $CompP_D$ close to 0.0 point to a large number of missing subjects, objects, and predicates in the dataset D, respectively.

6. Representation of the Crowd Knowledge

RDF triples allow for representing positive facts, i.e., negative triples cannot be modeled. However, considering negative knowledge is crucial to model the *local close world assumption* which, in turn, allows for avoiding redundant questions to the crowd. For example, if the crowd has stated that a given movie has no producers, the crowd will not be asked again about the producers for that movie. Moreover, using crowd knowledge effectively demands the representation of negative or even unknown statements: in some cases, human contributors establish or confirm facts, while in others they might assert that a statement cannot hold or that they do not know the answer to a question. Therefore, in HARE, the knowledge from the crowd is captured in three knowledge bases modeled as fuzzy sets to store positive, negative, and unknown facts: CKB^+, CKB^-, and $CKB^\sim$. CKB^+ comprises RDF triples that should belong to the dataset (positive facts). CKB^- lists all triples that should not exist (negative facts) according to the crowd. Finally, $CKB^\sim$ contains the associations that the crowd could not confirm or deny (unknown facts). In all crowd knowledge bases, triples are annotated with a membership degree m (> 0). m represents a score of the reliability of the crowd answer and, in this work, it is obtained from the worker's trust value reported by the Crowdflower platform.

Definition 9 *Given D an RDF dataset and CROWD a pool of human resources. Let D^* be a virtual finite RDF dataset such that it is composed of all the triples that 'should' be in D. The* ***knowledge of CROWD****, denoted CKB, is defined as a 3-tuple as follows:*

$$CKB = (CKB^+, CKB^-, CKB^\sim)$$

where $CKB^+, CKB^-, CKB^\sim$ are ***fuzzy RDF datasets*** *of the form (T, m) where T is a generalized RDF dataset and:*

- $m : T \rightarrow (0.0; 1.0]$, *where $m((s,p,o))$ is the membership degree of the triple $(s,p,o) \in T$ to the corresponding fuzzy set, and states the reliability of the crowd answer,*
- $(s,p,o) \in T^+$ *with* $CKB^+ = (T^+, m)$ *iff* $(s,p,o) \in U \times U \times (U \cup L)$ *and, according to CROWD, (s,p,o) belongs to D^*,*
- $(s,p,o) \in T^-$ *with* $CKB^- = (T^-, m)$ *iff* $(s,p,o) \in (U \cup B \cup L) \times (U \cup B \cup L) \times (U \cup B \cup L)$ *and, according to CROWD, (s,p,o) does not belong to D^*; and for all $(s,p,o) \in T^-$ it holds that $(s,p,o) \notin D^*$,*
- $(s,p,o) \in T^\sim$ *with* $CKB^\sim = (T^\sim, m)$ *iff* $(s,p,o) \in (U \cup B \cup L) \times (U \cup B \cup L) \times (U \cup B \cup L)$ *and, according to CROWD, the membership of (s,p,o) to D^* is unknown.*

Example 4 *CROWD is enquired to provide values of the predicate* dbp:producer *for the movie* dbr:Tower_Heist, *and links between* dbr:Tower_Heist *and the person* dbr:Brian_Grazer. *Suppose that the crowdsourced answers are as follows:*

(i) "Brian Grazer is a producer of Tower Heist", with confidence equal to 0.9,

(ii) "There is no relationship between Tower Heist and Brian Grazer", with confidence equal to 0.04,

(iii) "Tower Heist has no producers", with confidence equal to 0.06,

(iv) "I do not know the relationship between Tower Heist and Brian Grazer", with confidence equal to 0.01.

The previous CROWD answers are then stored in the corresponding CKB.[4] *For instance, answer (i)*

[4] For the sake of readability, in the following examples a triple (s,p,o) stored in CKB is represented as $(s,p,o,m((s,p,o)))$.

asserts facts that should be in D, therefore it is stored in CKB^+ as follows:

CKB^+:
(dbr:Tower_Heist, dbp:producer, dbr:Brian_Grazer, 0.9)

Answers (ii) and (iii) correspond to facts that should not be in the dataset D, therefore:

CKB^-:
(dbr:Tower_Heist, _:p1, dbr:Brian_Grazer, 0.04)
(dbr:Tower_Heist, dbp:producer, _:o, 0.06)

Lastly, in answer (iv) CROWD has declared that the vetted fact is unknown, hence:

$CKB^\sim$:
(dbr:Tower_Heist, _:p2, dbr:Brian_Grazer, 0.01)

Given that CKB contains triples that are not in D, it is important to consider the information stored in CKB when determining the completeness of resources. We therefore account the answers previously retrieved from *CROWD*. Analogous to Definition 8, we define the completeness of a resource considering the knowledge captured in CKB.

Definition 10 *Let s, p, and o be RDF resources, with $(s, a, C_{s1}) \in D, ..., (s, a, C_{sn}) \in D$ and $(o, a, C_{o1}) \in D, ..., (o, a, C_{om}) \in D$. The **completeness of RDF resources with respect to the crowd knowledge base** CKB is given by the multiplicity of RDF resources in CKB and the classes that they belong to in the dataset D. $CompS_{CKB}(o|p)$ denotes the completeness of subjects in CKB for resource o via the predicate p; $CompO_{CKB}(s|p)$ denotes the completeness of objects in CKB for resource s via the predicate p; and $CompP_{CKB}(s|o)$ denotes the completeness of predicates in CKB that link resources s and o. The completeness of RDF resources ($[0.0; 1.0]$) with respect to CKB is defined as follows:*

$$CompS_{CKB}(o|p) := \begin{cases} \frac{MS_{CKB}(o|p)}{AMS'_D} & AMS'_D \neq 0 \wedge \\ & MS_{CKB} < AMS'_D \\ 1.0 & otherwise \end{cases}$$

$$CompO_{CKB}(s|p) := \begin{cases} \frac{MO_{CKB}(s|p)}{AMO'_D} & AMO'_{CKB} \neq 0 \wedge \\ & MO_{CKB} < AMO'_D \\ 1.0 & otherwise \end{cases}$$

$$CompP_{CKB}(s|o) := \begin{cases} \frac{MP_{CKB}(s|o)}{AMP'_D} & AMP'_D \neq 0 \wedge \\ & MP_{CKB} < AMP'_D \\ 1.0 & otherwise \end{cases}$$

Where MS_{CKB}, MO_{CKB}, and MP_{CKB} are defined as follows:

$$MS_{CKB}(o|p) := |\{s \mid (s,p,o) \in T^+ \wedge (s,p,o) \notin D\}|$$
$$MO_{CKB}(s|p) := |\{o \mid (s,p,o) \in T^+ \wedge (s,p,o) \notin D\}|$$
$$MP_{CKB}(s|o) := |\{p \mid (s,p,o) \in T^+ \wedge (s,p,o) \notin D\}|$$

and:

$$AMS'_D = max(AMS_D(C_{o1}|p), ..., AMS_D(C_{om}|p)),$$
$$AMO'_D = max(AMO_D(C_{s1}|p), ..., AMO_D(C_{sn}|p)),$$
$$AMP'_D = max(AMP_D(C_{s1}|C_{o1}), ..., AMP_D(C_{sn}|C_{om})).$$

Although the crowd knowledge bases may contain rdf:type or rdfs:subClassOf statements, Definition 10 only takes into consideration the class and sub-class annotations that are specified the dataset D. In this way, the estimation of completeness exploits the information encoded in ontological definitions in D, which are assumed to be correct.

Example 5 *Consider the status of the crowd knowledge base CKB^+ given in Example 4 and the aggregated multiplicity for classes in D shown in Example 2. According to CKB^+, the object multiplicity of the resource dbr:Tower_Heist for the predicate dbp:producer is 1. Therefore, the object completeness in CKB ($CompO_{CKB}$) of the resource dbr:Tower_Heist for dbp:producer is computed as: MO_{CKB}(dbr:Tower_Heist | dbp:producer) divided by AMO_D(schema.org:Movie | dbp:producer), i.e., $\frac{1}{3} = 0.33$.*

The representation of crowd knowledge as CKB^+, CKB^-, and $CKB^\sim$ allows for easily modeling contradictions or unknownness in *CROWD*.

6.1. Crowd Contradiction

A contradiction arises when members of the *CROWD* assert that a certain value exists and does not exist. An example of contradiction is given in Example 4, where the crowd confirms that *Tower Heist has a producer and is Brian Grazer* (in CKB^+) but also states that the movie *Tower Heist has no producers* (in CKB^-). In order to detect correspondences like these among triples in CKB^+ and CKB^- we introduce the relation of subsumption for generalized RDF triples.

Definition 11 *Given an RDF triple $(s, p, o) \in U \times U \times (U \cup L)$. Let _:bs, _:bp, _:bo be RDF blank*

nodes. The relation of ***subsumption of generalized RDF triples*** *is defined as follows:*

$$\begin{array}{lll}(s,p,o) & \sqsubseteq & (s,p,o)\\ (s,p,o) & \sqsubseteq & (\_{:}bs,p,o)\\ (s,p,o) & \sqsubseteq & (s,\_{:}bp,o)\\ (s,p,o) & \sqsubseteq & (s,p,\_{:}bo)\end{array}$$

Example 6 *The generalized RDF triples* $t2$ =(dbr:Tower_Heist, dbp:producer, _:o) *and* $t3$ =(dbr:Tower_Heist, _:p, dbr:Brian_Grazer) *subsume the triple* $t1$ =(dbr:Tower_Heist, dbp:producer, dbr:Brian_Grazer)*, i.e.,* $t1 \sqsubseteq t2$ *and* $t1 \sqsubseteq t3$*.*

Property 1 *Given a dataset* D *and an RDF generalized triple* $(s,p,o) \in D$*, the triples* $(s',p',o') \in D$ *subsumed by* (s,p,o)*, i.e.,* $(s',p',o') \sqsubseteq (s,p,o)$*, can be computed in* $O(D)$*.*

In HARE, contradictions can be detected by computing subsumption relations between triples in $CKB^+ = (T^+, m)$ and $CKB^- = (T^-, m)$. Formally, a *CROWD* contradiction occurs when exists triples $(s_1,p_1,o_1) \in T^+$ and $(s_2,p_2,o_2) \in T^-$ such that:

$$(s_1,p_1,o_1) \sqsubseteq (s_2,p_2,o_2)$$

Example 7 *To illustrate the concept CROWD contradictions, consider the triples in* CKB^+ *and* CKB^- *in Example 4. The first contradiction in* CKB *corresponds to the existence of producers of the movie* dbr:Tower_Heist*. In* CKB^+ *it is confirmed that Tower Heist has a producer and is Brian Grazer, i.e.,* t_1 =(dbr:Tower_Heist, dbp:producer, dbr:Brian_Grazer)*. However, according to* CKB^-*, the movie Tower Heist has no producers, i.e.,* t_2 =(dbr:Tower_Heist, dbp:producer, _:o)*. Given that* $t_1 \sqsubseteq t_2$*, this is considered a contradiction. Another contradiction that occurs in the* CKB *from Example 4 corresponds to the relationship between* dbr:Tower_Heist *and* dbr:Brian_Grazer*. According to* t_1*, these resources are related via the* dbp:producer *predicate. Nonetheless, as stated in* CKB^-*, there is no relationship between* dbr:Tower_Heist *and* dbr:Brian_Grazer*, i.e.,* t_3 =(dbr:Tower_Heist, _:p1, dbr:Brian_Grazer)*. This is a contradiction since* $t_1 \sqsubseteq t_3$*.*

When querying the crowd knowledge, the contradiction degree about statements in $CKB^+ = (T^+, m)$ and $CKB^- = (T^-, m)$ is measured by considering the membership degree of contradictory triples. Given a triple pattern t evaluated against CKB, we denote $m^+(t)$ the average membership degree of triples in CKB^+ that match t. Analogously, we denote $m^-(t)$ the average membership degree of triples in CKB^- that contradict triples that match t in CKB^+. Finally, the contradiction degree $C(t)$ for triple pattern t is computed as the harmonic mean between $m^+(t)$ and $m^-(t)$. The selection of the harmonic mean allows for comparing the rate to which triples are contradicted in CKB^+ and CKB^-. Formally, $C(t)$, with values in $[0.0; 1.0]$, is computed as follows:

$$C(t) = \begin{cases} 2 \cdot \frac{m^+(t) \cdot m^-(t)}{m^+(t) + m^-(t)} & \text{if } m^+(t) + m^-(t) \neq 0 \\ 1.0 & \text{otherwise} \end{cases} \tag{3}$$

With:

$$\begin{array}{lll} m^+(t) & = & avg(\{\{m(\mu(t)) \mid \mu \in [[t]]_{T^+}\}\}),\\ m^-(t) & = & avg(\{\{m(s,p,o) \mid (s,p,o) \in T^-,\\ & & \mu \in [[t]]_{T^-} \text{ with } \mu(t) = (s,p,o) \vee\\ & & \exists \mu \in [[t]]_{T^+},\ \mu(t) \sqsubseteq (s,p,o)\}\}). \end{array} \tag{4}$$

We assume by default that human knowledge captured in the CKB is contradictory. Therefore, $C(t)$ is 1.0 when there is no information about the crowd performance regarding t, which happens when $m^+(t)$ and $m^-(t)$ are equal to zero. $C(t) = 1.0$ indicates high contradiction, as specified in Equation 3.

Example 8 *Assume that the triple pattern* t =(dbr:Tower_Heist, dbp:producer, ?producer) *is executed against the* CKB *from Example 4. When* t *is executed against* CKB^+*, only the triple* $t1$ = (dbr:Tower_Heist, dbp:producer, dbr:Brian_Grazer, 0.9) *matches* t*, i.e.,* $t1 = \mu(t)$ *and* $\mu \in [[t]]_{CKB^+}$*. Therefore,* $m^+(t)$ *is equal to* $avg(\{\{0.9\}\})$*, i.e.,* $m^+(t) = 0.90$*. To compute* $m^-(t)$ *it is necessary to obtain the triples in* CKB^- *that contradict the triples that match* t *in* CKB^+*. In Example 8, it is shown that* t_2 =(dbr:Tower_Heist, dbp:producer, _:o, 0.06) *and* t_3 =(dbr:Tower_Heist, _:p1, dbr:Brian_Grazer, 0.04) *contradict* $t1$*. Then,* $m^-(t)$ *is computed as* $avg(\{\{0.06, 0.04\}\})$*, i.e.,* $m^-(t) = 0.05$*. Finally, the contradiction degree about producers of the movie* dbr:Tower_Heist *is* $2 \cdot \frac{0.90 \cdot 0.05}{0.90 + 0.05}$*, i.e.,* $C(t) = 0.094$*.*

Contradiction values close to 0.0 indicate high consensus on the (non-)existence of triples in the virtual dataset D^*.

6.2. Crowd Unknownness

Statements for which members of *CROWD* has declared to be unknowledgeable about are stored in $CKB^{\sim}$. Given a triple pattern t, the unknownness degree $U(t)$ of t is computed as the average membership degree of triples that match t in $CKB^{\sim}$. Formally, $U(t)$ ($[0.0; 1.0]$) is computed as follows:

$$U(t) = m^{\sim}(t) \tag{5}$$

With: $m^{\sim}(t) = avg(\{\{m(\mu(t)) \mid \mu \in [[t]]_{T^{\sim}}\}\})$ and $CKB^{\sim} = (T^{\sim}, m)$.

When no triples in $CKB^{\sim} = (T^{\sim}, m)$ match a triple pattern t ($[[t]]_{T^{\sim}} = \emptyset$) then $U(t) = 0.0$, which means that *CROWD* is knowledgeable w.r.t. t.

Example 9 *Suppose that the triple pattern* t =(dbr:Tower_Heist, dbp:producer, ?producer) *is executed against the CKB from Example 4. The triple* (dbr:Tower_Heist, dbp:producer, _:o, 0.01) *in* $CKB^{\sim}$ *matches* t. *The crowd unknownness about the producers of the movie* dbr:Tower_Heist *is* $m^{\sim}(t) = avg(\{\{0.01\}\})$. *Then, the crowd unknownness for* t *is* $U(t) = 0.01$.

In general, unknownness values close to 1.0 indicate that *CROWD* has shown to be unknowledgeable about the vetted fact. High uncertainty values point that *CROWD* does not have the knowledge to answer this question, and hence it is not useful to assess this fact with the crowd any further.

7. Microtask Manager

The microtask manager creates human tasks from triple patterns, submits them to the crowdsourcing platform, and gathers the crowd answers. This component is composed of the user interface generator and the microtask executor.

Table 1: Predicates dereferenced by the user interface generator to build the HARE microtasks. Objects are displayed using appropriate HTML tags.

Predicate	Object Type	HTML Tag
rdfs:label	Literal	...
rdfs:comment	Literal	...
foaf:depiction	URI	
foaf:homepage	URI	<a>...
foaf:isPrimaryTopicOf	URI	<a>...
geo:lat geo:long	(Typed) Literal (Typed) Literal	Map API

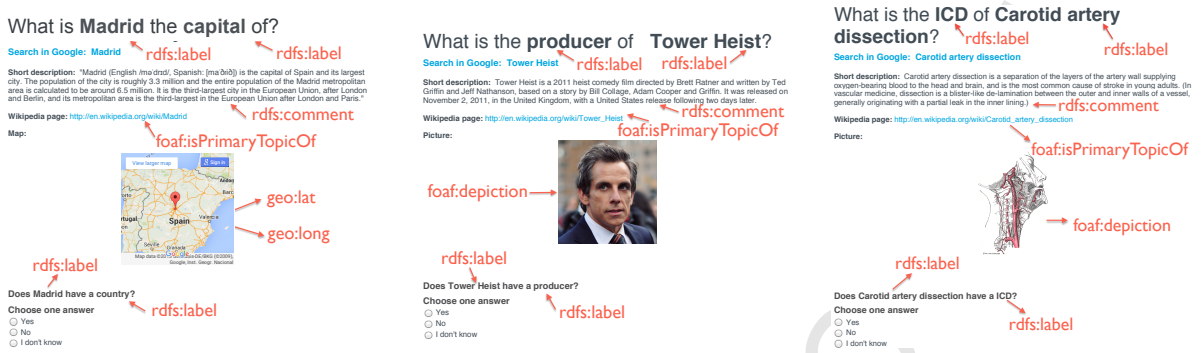
7.1. User Interface Generator

The user interface generator receives as input the triple patterns to be crowdsourced. This component is able to generate interfaces for triple patterns with at most one variable. In addition, this component exploits the semantics of RDF resources in triple patterns to build rich human-readable interfaces to RDF data. A HARE microtask created by the user interface generator is defined as follows.

Definition 12 *A* ***microtask*** *MT is a set of 2-tuples* (t, h_t) *where* t *is a triple pattern and* h_t *corresponds to human readable information related to* t. *The granularity of a microtask MT is denoted* $|MT|$, *i.e., the number of triple patterns contained in a single task.*

The human-readable information h_t is obtained by the user interface generator by dereferencing URIs in the triple pattern t. For example, a HARE microtask displays "Tower Heist" obtained via the rdfs:label, instead of showing the resource URI http://dbpedia.org/resource/Tower_Heist. However, displaying only the labels of resources when generating user interfaces might generate ambiguity and, in consequence, incorrect answers may be retrieved from the crowd. To illustrate, assume that the different films dbr:Beauty_and_the_Beast_(1991_film) and dbr:Beauty_and_the_Beast_(2017_film) have the label "The Beauty and the Beast".[5] Consider now that the triple pattern (dbr:Beauty_and_the_Beast_(2017_film), dbp:producer, ?o) is crowdsourced. Then, the user interface generator would create a microtask that asks "What is the producer of The Beauty and the Beast?". In this case, the crowd could interpret that question is referring to the film of 1991, which would be incorrect. This simple example illustrates how using the value of only one property to describe the resource may generate ambiguity when contacting the crowd. In order to avoid this, HARE exploits the semantic descriptions of resources and includes further properties in the microtasks. The more properties to describe the resources are included in the microtasks, the smaller the probability that all the values of those properties are ambiguous.

[5] The values of rdfs:label of the resources in DBpedia are directly extracted from the URIs (which unequivocally identifies a resource), therefore cases like the one in the example are rare in DBpedia. This particularity, however, does not necessarily hold for all datasets, making the values of rdfs:label ambiguous. Furthermore, ambiguity may still arise in DBpedia if the property foaf:name is used instead.



(a) Microtask generated for (?country, dbo:capital, dbr:Madrid).

(b) Microtask generated for (dbr:Tower_Heist, dbp:producer, ?producer).

(c) Microtask generated for (dbr:Carotid_artery_dissection, dbp:icd, ?icd).

Figure 4: HARE microtasks. The depicted interfaces are built exploiting the semantics of RDF resources from different knowledge domains: (a) geography, (b) movies, and (c) life sciences. Predicates used to build each interface are highlighted. The crowd selects *"Yes"* when the requested value exists, *"No"* when it does not exist, and *"I don't know"* when the existence of the value is unknown.

The user interface generator displays the values (if available) of different properties of RDF resources (cf. Table 1) such as short description (via rdfs:comment), picture (via foaf:depiction), geo-location depicted in a map (via geo:lat and geo:long), and links to the homepage (via foaf:homepage) and further documents (via foaf:isPrimaryTopicOf). Providing details like these in microtasks has also proven to assist the crowd in providing right answers [6]. The object of the different predicates are displayed using HTML tags according to the object type. For instance, a picture obtained via the foaf:depiction predicate is rendered using the `img` HTML tag. Figure 4 depicts microtask interfaces generated for three triple patterns. Table 1 lists the RDF predicates that are dereferenced to build the HARE microtask interfaces.

The HARE microtasks first enquire the crowd about existence of a value for the triple pattern. For instance, for the triple pattern t =(dbr:Madrid, dbo:country, ?country) the task displays: *"Does Madrid have a country?"*. We provide three possible answers to this question: *"Yes"*, *"No"*, and *"Unknown"*. For example, the answer *"Yes"* states that there exists a value for the variable ?country, i.e., that Madrid has a country; the answer *"No"* states that Madrid has no country; and *"Unknown"* indicates that the crowd does not know the answer. When the answer is *"Yes"*, a second question requires the crowd to provide values, e.g., *"What is the country of Madrid?"*. The provided values are bindings of the triple pattern variables – in our example, instantiations of the variable ?country – which are used to complete missing values in RDF datasets.

7.2. Microtask Executor

The microtask executor submits the human tasks created by the user interface generator to the crowdsourcing platform. Answers provided by *CROWD* in each task are retrieved by the microtask executor and processed in order to update the crowd knowledge bases (cf. Section 6) accordingly.

Definition 13 *Let t be a triple pattern crowdsourced in a microtask MT. The* ***crowd answer*** *of MT for t is a 3-tuple of the form (a_t, μ_t, M_t), where $a_t \in \{$"Yes", "No", "Unknown"$\}$ indicates the existence of the value crowdsourced in t, μ_t is the mapping of variables in t to RDF terms, and M_t corresponds to metadata about the performance of the crowd when assessing t. When $a_t =$ "Yes", then $\mu_t(x) \in (U \cup L)$, otherwise $\mu_t(x) \in B$, for all $x \in vars(t)$. $\mu_t(t)$ is the triple obtained when replacing all $x \in dom(\mu_t)$ in t by $\mu_t(x)$.*

Example 10 *Consider that the triple pattern $t =$(dbr:Tower_Heist, dbp:producer, ?producer) is crowdsourced, where CROWD is enquired to provide producers for the movie dbr:Tower_Heist. The crowd answer (i) "Brian Grazer is a producer of Tower Heist" with confidence 0.9 from Example 4 is retrieved by the microtask executor as ("Yes",* {producer → dbr:Brian_Grazer}, 0.9*). Analogously, the crowd answer (iii) "Tower Heist has no producers" with confidence 0.06 is modeled as ("No",* {producer → _:o}, 0.06*).*

In HARE crowd answers for a triple pattern t, a_t indicates whether $\mu_t(t)$ provided by *CROWD* is stored, i.e., either in CKB^+, CKB^-, or $CKB^\sim$. The metadata M_t about the performance of the crowd is used to compute the membership degree m (cf. Definition 9) of the answer in CKB. In our implementation, we utilized the worker's trust value directly provided by Crowdflower as the membership degree m of a mapping $\mu_t(t)$. The microtask executor processes the crowd answers and update CKB as follows.

Definition 14 *Given a crowd answer (a_t, μ_t, M_t) of a microtask where triple pattern t is crowdsourced. Let $CKB = (CKB^+, CKB^-, CKB^\sim)$ be the crowd knowledge. The value of a_t determines the* ***crowd knowledge base to be updated****: "Yes" $\rightsquigarrow CKB^+$, "No" $\rightsquigarrow CKB^-$, and "Unknown" $\rightsquigarrow CKB^\sim$. Let (T, m) be the crowd knowledge base selected according to a_t. Consider t' the triple pattern obtained by replacing all RDF blank nodes in $\mu_t(t)$ by fresh variables. The update of (T, m) considers the following two cases:*

- *If there are triples in T that match t' ($[[t']]_T \neq \emptyset$), then the membership degree of each solution of $[[t']]_T$ is updated as follows:*

$$\forall \mu \in [[t']]_T, m(\mu(t)) := max(m(\mu(t)), M_t)$$

- *Otherwise, if there are no matches ($[[t']]_T = \emptyset$), then $\mu_t(t)$ is added to T and annotated with the membership degree m as follows:*

$$T := T \cup \{\mu_t(t)\},\ m(\mu_t(t)) := M_t$$

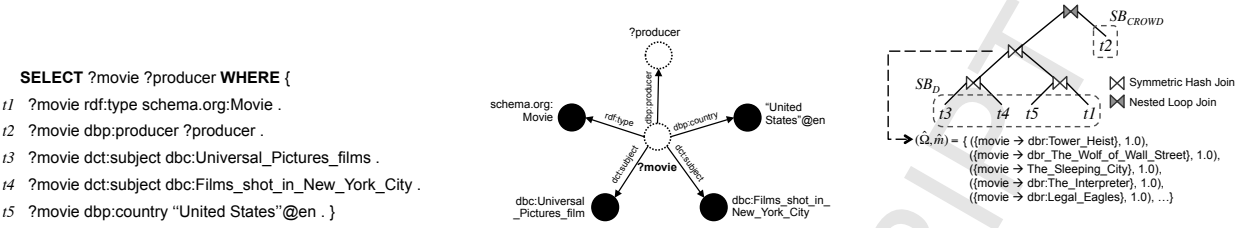
Example 11 *Assume that CKB contains the triples shown in Example 4. Consider that the triple pattern $t =$ (dbr:Tower_Heist, dbp:producer, ?producer) is crowdsourced and one of the answers provided by CROWD is ("Yes", {producer → dbr:Kim_Roth}, 0.85). In this case, the triple pattern $t' =$ (dbr:Tower_Heist, dbp:producer, dbr:Kim_Roth) is evaluated against the triples stored in CKB^+. Since no triples in CKB^+ match t', the triple (dbr:Tower_Heist, dbp:producer, dbr:Kim_Roth) provided by CROWD is considered new and added to CKB^+ with membership degree equal to 0.85. Now consider that another answer ("No", {producer→ _:o1}, 0.05) is provided by CROWD, i.e., $\mu_t(t) =$ (dbr:Tower_Heist, dbp:producer, _:o1). Therefore, the triple pattern $t' =$ (dbr:Tower_Heist, dbp:producer, ?o) is built by replacing the blank node _:o1 by the variable ?o and executed against the triples in CKB^-. Given that CKB^- contains $t1 =$ (dbr:Tower_Heist, dbp:producer, _:o, 0.06), there is a match for t'; thus, no triples are added to CKB^- but the membership of $t1$ is updated as $max(0.05, 0.06) = 0.06$.*

Algorithm 1: HARE BGP Optimizer

```
Input: A BGP B of a SPARQL query Q.
Output: A plan query T_Q, a decomposition (SB_D, SB_CROWD).
SB_D = ∅, SB_CROWD = ∅
// Partition triple patterns and get multiplicity
for tp_i ∈ B do
    if |vars(tp_i)| > 1 // Triple patterns with one constant
    then
        SB_CROWD = SB_CROWD ∪ {tp_i}
    else
        SB_D = SB_D ∪ {tp_i}
        tp_i.m = M_D(tp_i)
// Phase 1: Order patterns in SB_D such that tp'_i.m ≤ tp'_{i+1}.m
S = ⟨tp'_1, tp'_2, ..., tp'_k⟩
// Build bushy star-shaped groups (SSGs)
while exists s_i, s_j in S such that |vars(s_i) ∩ vars(s_j)| = 1 do
    Select s_i, s_j in S with lowest values i, j
    S = S.append((s_i ⋈_SHJ s_j))
    S.remove(s_i)
    S.remove(s_j)
// Phase 2: Build hybrid SSGs adding triples from SB_CROWD
for tp_i ∈ SB_CROWD do
    Select s from S such that |vars(s) ∩ vars(tp_i)| = 1
    s = (s ⋈_NL tp_i)
// Phase 3: Join hybrid SSGs in bushy trees
T_B = set(S)
do
    T'_B = T_B
    Select s_i, s_j from T_B such that
    vars(s_i) ∩ vars(s_j) ≠ ∅
    T_B = T_B ∪ {(s_i ⋈_SHJ s_j)} − {s_i, s_j}
while T'_B ≠ T_B
// Phase 4: Place Cartesian products among hybrid SSGs
while T_B > 1 do
    Select s_i, s_j with s_i ≠ s_j from T_B
    T_B = T_B ∪ {(s_i ⋈_SHJ s_j)} − {s_i, s_j}
return T_B, (SB_D, SB_CROWD)
```

8. Query Optimizer

The HARE optimizer devises physical plans that can be executed efficiently. Given a SPARQL query Q, the HARE optimizer reorders the triple patterns within BGPs, respecting the ordering of UNION, OPTIONAL, and FILTER operators specified in Q. Triple patterns from Q are grouped into hybrid star-shaped groups; star-shaped groups (SSGs) share exactly one variable [34] and contain triple patterns that are executed against the dataset D and against *CROWD*. Then, hybrid stars are combined in a bushy tree plan. Both star-shaped queries and bushy plans have proven to reduce the size of intermediate results [34], which reduces the



(a) Query to *select movies and producers of movies filmed in New York City by Universal Pictures*. This query contains a BGP $B = \{t1, t2, t3, t4, t5\}$.

(b) Hybrid SSG for BGP B around ?movie. Nodes correspond to terms in subject or object position. Filled nodes are constants.

(c) A physical plan $\mathcal{T}_B$, partitions SB_D and SB_{CROWD} for B. Results $(\hat{\Omega}, \hat{m})$ obtained when executing sub-plan against D.

Figure 5: Example of HARE query optimization. (a) SPARQL query from running example. (b) Hybrid Star-Shaped Group (SSG) of the BGP contained in the query. (c) Query plan against DBpedia and *CROWD*.

number of questions posed to *CROWD*. The proposed HARE optimizer extends the optimization techniques of nLDE [5], by generating hybrid SSGs and grouping them in bushy trees, instead of grouping SSGs in left-linear plans.

Given a SPARQL query Q, the HARE optimizer processes each BGP B contained in Q in four phases, as shown in Algorithm 1. First, the optimizer decomposes each BGP B into two partitions: SB_D comprises triple patterns executed against the dataset D, and SB_{CROWD} contains triple patterns that may be crowdsourced. To build SB_D and SB_{CROWD}, the optimizer follows the intuition that the evaluation of triple patterns with few bound arguments (i.e., triple patterns with several variable) will generate mappings that may yield missing values. In this way, the optimizer generates a decomposition that increases the chances of completing the query answers when contacting the crowd. Therefore, the optimizer implements the following heuristic: triple patterns where only the subject, predicate, or object is bound to a constant are added to SB_{CROWD} for resorting to the crowd the completion of missing values. The other triple patterns are annotated with their multiplicity M_D and added to SB_D. Given a triple pattern $t = (s, p, o)$, M_D is obtained as follows:
if $vars(t) = \{s\}$, then $MS_D(o|p)$,
if $vars(t) = \{o\}$, then $MO_D(s|p)$,
if $vars(t) = \{p\}$, then $MP_D(s|o)$.

For example, consider the query from Figure 5a, composed of one BGP B with five triple patterns $t1$, $t2$, $t3$, $t4$, and $t5$. The optimizer starts by computing M_D for each triple pattern and building the partitions SB_D and SB_{CROWD} (lines 2-8, Algorithm 1). For instance, $t1$ = (?movie, rdf:type, schema.org:Movie) is added to SB_D and annotated with the corresponding multiplicity $M_D(t1) = 90,063$. Analogously, the triple pattern $t2$ = (?movie, dbp:producer, ?producer) is processed and added to SB_{CROWD}. After all patterns are processed, it is obtained that $SB_D = \{t1, t3, t4, t5\}$ and $SB_{CROWD} = \{t2\}$. Then, triple patterns in SB_D are ordered (line 9) according to their multiplicity values. In our example, the result is $S = \langle t3, t4, t5, t1\rangle$. Ordering triple patterns by their multiplicity allows for grouping in stars the most selective patterns, and consequentially, evaluating selective patterns first during query execution.

In phase 1, Algorithm 1 proceeds to build SSGs with patterns in S (lines 10-14); patterns are combined using Symmetric Hash Join operators ($\bowtie_{SHJ}$), to evaluate them against the dataset D simultaneously. Following our running example, the optimizer first joins $t3$ and $t4$ since they share exactly one variable (?movie) and add this sub-plan to S, i.e., $S = \langle t5, t1, (t3 \bowtie_{SHJ} t4)\rangle$. In a second iteration, the algorithm joins $t5$ and $t1$, hence, $S = \langle (t3 \bowtie_{SHJ} t4), (t5 \bowtie_{SHJ} t1)\rangle$. Sub-plans $(t3 \bowtie_{SHJ} t4)$ and $(t5 \bowtie_{SHJ} t1)$ in S are joined in a subsequent iteration, since triple patterns $t3$, $t4$, $t5$, and $t1$ share the variable ?movie.[6] At this point, S contains one SSG combined in the bushy tree $((t3 \bowtie_{SHJ} t4) \bowtie_{SHJ} (t5 \bowtie_{SHJ} t1))$.

In the second phase, Algorithm 1 builds hybrid SSGs by combining bushy trees in S with triple patterns in SB_{CROWD} (lines 15-17). In this phase, the optimizer places Nested Loop Join operators ($\bowtie_{NL}$) such that intermediate results produced by bushy tree plans are used to instantiate triple patterns in

[6] In Algorithm 1, the variables of a sub-plan s_i, i.e., $vars(s_i)$, are defined as the union of the variables of triple patterns contained in s_i.

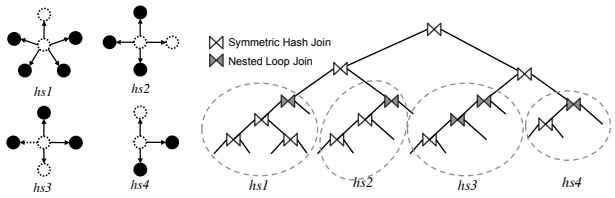


Figure 6: HARE optimizer: Phases 3 and 4. Bushy tree plan for four hybrid SSGs $hs1$, $hs2$, $hs3$, $hs4$.

SB_{CROWD}. In our example, the bushy tree subplan $((t3 \bowtie_{SHJ} t4) \bowtie_{SHJ} (t5 \bowtie_{SHJ} t1))$ is joined with the triple pattern $t2$ in SB_{CROWD}, producing the hybrid SSG depicted in Figure 5b.

In phases 3 and 4 of Algorithm 1, the optimizer combines the hybrid SSGs built in the previous phase in bushy trees using Symmetric Hash Joins. Figure 6 depicts four hybrid SSGs which are combined in a bushy tree plan in phases 4 and 5. In our running example, these phases are not executed since the query from Figure 5a only has one SSG. The outcome of Algorithm 1 for the BGP in the SPARQL query from Figure 5a is the plan $\mathcal{T}_B$ and partition (SB_D, SB_{CROWD}) depicted in Figure 5c.

9. Query Engine

The HARE query engine gathers information from LOD datasets, its crowd knowledge bases containing curated results of prior crowdsourced tasks, and the crowd itself. Because RDF data collected from the crowd knowledge bases and the crowd is not necessarily precise, our approach uses fuzzy RDF to capture multiple degrees of vagueness and imprecision when combining data from crowd-based sources. We have extended the semantics of SPARQL queries in a way that correct data from LOD datasets and vague data from crowd knowledge bases can be merged during SPARQL query processing. Specifically, we extend the set-based SPARQL semantics to model degrees of membership of a mapping to the evaluation of a SPARQL expression. This is different from prior related work [9, 16, 32, 36] which, in addition to providing a new fuzzy semantics for SPARQL, extend the language itself to represent fuzzy queries. In HARE, users do not need to be aware of vagueness, and continue to specify queries using SPARQL. When some form of crowd knowledge is required to complete the query answer, the proposed SPARQL semantics will allow for representing the degree of imprecision or vagueness that the corresponding mapping belongs to the answer.

Definition 15 *Let $\mathcal{M}$ be the universe of all SPARQL mappings [30]. A **SPARQL mapping fuzzy set** is a tuple $(\hat{\Omega}, \hat{m})$, where $\hat{\Omega}$ is a mapping set and $\hat{m} : \mathcal{M} \rightarrow (0.0; 1.0]$ is a partial function with respect to $\mathcal{M}$ such that $\hat{m}(\mu)$ is defined for all $\mu \in \hat{\Omega}$. Given $\mu \in \hat{\Omega}$, we refer to $\hat{m}(\mu)$ as the membership degree of μ to $\hat{\Omega}$.*

Definition 16 *Let $F := (\Omega, \hat{m})$, $F_l := (\Omega_l, \hat{m}_l)$, $F_r := (\Omega_r, \hat{m}_r)$ be mapping fuzzy-sets, $S \subset V$ a finite set of variables, and R be a filter condition. The expression of **SPARQL fuzzy set algebraic operations** are as follows:*

$$
\begin{aligned}
&F_l \bowtie F_r := (\hat{\Omega}, \hat{m}'), \text{ where:} \\
&\quad \hat{\Omega} := \{\mu_l \cup \mu_r \mid \mu_l \in \Omega_l, \mu_r \in \Omega_r : \mu_l \sim \mu_r\}, \\
&\quad \hat{m}'(\mu) := \bigoplus_{(\mu_l,\mu_r)\in\{(\mu_l^*,\mu_r^*)\in\Omega_l\times\Omega_r \mid \mu_l^*\cup\mu_r^*=\mu\}} \\
&\qquad\qquad (\hat{m}_l(\mu_l) \otimes \hat{m}_r(\mu_r)) \text{ for all } \mu \in \hat{\Omega}. \\
&F_l \cup F_r := (\hat{\Omega}, \hat{m}'), \text{ where:} \\
&\quad \hat{\Omega} := \{\mu_{lr} \mid \mu_{lr} \in \Omega_l \text{ or } \mu_{lr} \in \Omega_r\}, \\
&\quad \hat{m}'(\mu) := \hat{m}_l(\mu) \oplus \hat{m}_r(\mu) \text{ for all } \mu \in \hat{\Omega}. \\
&F_l \setminus F_r := (\hat{\Omega}, \hat{m}'), \text{ where:} \\
&\quad \hat{\Omega} := \{\mu_l \in \Omega_l \mid \text{ for all } \mu_r \in \Omega_r : \mu_l \nsim \mu_r\}, \\
&\quad \hat{m}'(\mu) := \hat{m}_l(\mu) \text{ for all } \mu \in \hat{\Omega}. \\
&F_l \barwedge F_r := (F_l \bowtie F_r) \cup (F_l \setminus F_r) \\
&\pi_S(F) := (\hat{\Omega}, \hat{m}'), \text{ where:} \\
&\quad \hat{\Omega} := \{\mu_1 \mid \exists \mu_2 : \mu_1 \cup \mu_2 \in \Omega \wedge \\
&\quad dom(\mu_1) \subseteq S \ \wedge \ dom(\mu_2) \cap S = \emptyset\}, \\
&\quad \hat{m}'(\mu) := \bigoplus_{u_+\in\{u_+^*\in\Omega \mid \pi_S(\{\mu_+^*\})=\{\mu\}\}} \hat{m}(\mu_+) \\
&\qquad\qquad \text{for all } \mu \in \hat{\Omega}. \\
&\sigma_R(F) := (\hat{\Omega}, m'), \text{ where:} \\
&\quad \hat{\Omega} := \{\mu \in \Omega \mid \mu \models R\}, \text{ and} \\
&\quad \hat{m}'(\mu) := \hat{m}(\mu) \text{ for all } \mu \in \hat{\Omega}.
\end{aligned}
$$

Where $\models$ refers to built-in boolean functions defined in [27], and the operators $\otimes$ and $\oplus$ correspond to t-norms and t-conorms, respectively, such that $a \otimes b \neq 0$ and $a \oplus b \neq 0$, for $a \neq 0$ and $b \neq 0$. The quantifier $\bigoplus$ is defined as:

$$\bigoplus_{0\leq i\leq n} a_i := a_0 \oplus (\bigoplus_{0<i\leq n} a_i)$$

In HARE, $\otimes$ and $\oplus$ correspond to the conjunction and disjunction operators from Gödel logic:

$$a \otimes b := min(a, b)$$
$$a \oplus b := max(a, b)$$

In the following, the proposed fuzzy set semantics of SPARQL is defined. This semantics make use of the algebra operators from Definition 16.

Definition 17 *Let $D = (T, m)$ be a fuzzy RDF dataset, t a triple pattern, and Q, Q_1, Q_2 SPARQL expressions, R a filter condition, and $S \subset V$ a finite set of variables. Let $[[\cdot]]_D^{\mathcal{F}}$ be a function that translates **SPARQL expressions into SPARQL fuzzy set algebra operators** as follows:*

$$
\begin{aligned}
&[[t]]_D^{\mathcal{F}} := (\hat{\Omega}, \hat{m}), \text{ where:} \\
&\quad \hat{\Omega} := \{\mu \mid dom(\mu) = vars(t) \text{ and } \mu(t) \in T\}, \\
&\quad \hat{m}(\mu) := m(\mu(t)), \text{ for all } \mu \in \Omega. \\
&[[Q_1 \text{ AND } Q_2]]_D^{\mathcal{F}} := [[Q_1]]_D^{\mathcal{F}} \bowtie [[Q_2]]_D^{\mathcal{F}} \\
&[[Q_1 \text{ OPT } Q_2]]_D^{\mathcal{F}} := [[Q_1]]_D^{\mathcal{F}} \mathbin{⟕} [[Q_2]]_D^{\mathcal{F}} \\
&[[Q_1 \text{ UNION } Q_2]]_D^{\mathcal{F}} := [[Q_1]]_D^{\mathcal{F}} \cup [[Q_2]]_D^{\mathcal{F}} \\
&[[Q \text{ FILTER } R]]_D^{\mathcal{F}} := \sigma_R([[Q]]_D^{\mathcal{F}}) \\
&[[\text{SELECT}_S(Q)]]_D^{\mathcal{F}} := \pi_S([[Q]]_D^{\mathcal{F}})
\end{aligned}
$$

Example 12 *Let $D = (T, m)$ be a fuzzy RDF dataset. Consider a SPARQL expression Q that retrieves from D resources with producers and directors, as follows: $Q = (t_1 \text{ AND } t_2)$, where t_1 = (?r, dbp:producer, ?p) and t_2 = (?r, dbp:director, ?d).*

The expression Q is then evaluated against D using fuzzy set semantics and, according to Definition 17:

$[[Q]]_D^{\mathcal{F}} = ([[t_1]]_D^{\mathcal{F}} \bowtie [[t_2]]_D^{\mathcal{F}})$

Assume that $[[t_1]]_D^{\mathcal{F}}$ and $[[t_2]]_D^{\mathcal{F}}$ generate the mapping-fuzzy sets $(\hat{\Omega}_1, \hat{m}_1)$ and $(\hat{\Omega}_2, \hat{m}_2)$, respectively, as follows:[7]

$(\hat{\Omega}_1, \hat{m}_1) = \{$
$\mu_1 = \{r \to$ dbr:Six_Weeks, $p \to$ dbr:Jon_Peters, $\hat{m}_1 = 0.80\}$,
$\mu_2 = \{r \to$ dbr:Tower_Heist, $p \to$ dbr:Kim_Roth, $\hat{m}_2 = 0.90\}\}$
$(\hat{\Omega}_2, \hat{m}_2) = \{$
$\mu_3 = \{r \to$ dbr:Six_Weeks, $d \to$ dbr:Toni_Bill, $\hat{m}_2 = 0.90\}\}$

We denote $(\hat{\Omega}, \hat{m})$ the result of combining $(\hat{\Omega}_1, \hat{m}_2)$ and $(\hat{\Omega}_r, \hat{m}_r)$ with the $\bowtie$ operator according to Definition 16. To illustrate the evaluation of the $\bowtie$ operator, first we look at the compatible mappings from $\hat{\Omega}_1$ and $\hat{\Omega}_2$. For instance, the mappings μ_1 and μ_3 are compatible, since $\mu_1(r) = \mu_3(r)$ which is dbr:Six_Weeks*, and r is the only variable they share. The compatible mappings are then combined:*

$\mu_4 = \mu_1 \cup \mu_2$
$= \{r \to$ dbr:Six_Weeks, $p \to$ dbr:Jon_Peters, $d \to$ dbr:Toni_Bill $\}$

Then, following Definition 16, $\hat{m}$ is computed for the combined mapping μ_4. In this case, $\hat{m}(\mu_4)$ is simply computed as:

[7] For the sake of readability, the values of $\hat{m}_1$ and $\hat{m}_2$ are presented inside of each solution mapping.

Algorithm 2: HARE BGP Executor

Input: A BGP B, an RDF dataset D, a crowd knowledge CKB, and threshold τ.
Output: The fuzzy result set $(\hat{\Omega}, \hat{m})$.

```
// 1. Get query plan and decomposition (Algorithm 1)
T_B, (SB_D, SB_CROWD) = hareOptimizer(B)
// 2. Evaluate bushy-tree plan T_B|SB_D against D
Ω̂ = [[T_B|SB_D]]_D, and m̂(μ) = 1.0 for all μ ∈ Ω̂
// 3. Evaluate triple patterns in T_B|SB_CROWD
for t_CROWD ∈ T_B|SB_CROWD do
    for μ ∈ Ω̂ do
        t = μ(t_CROWD)
        if Comp(t) < 1.0 then
            if P_CROWD(t) > τ then
                Invoke Microtask Manager with t
        Ω̂_1 = [[t]]_D, and m̂_1(μ') = 1.0 for all μ' ∈ Ω̂_1
        (Ω̂_2, m̂_2) = [[t]]^F_CKB+
        (Ω̂, m̂) = (Ω̂, m̂) ⋈ ((Ω̂_1, m̂_1) ∪ (Ω̂_2, m̂_2))
return (Ω̂, m̂)
```

$\hat{m}(\mu_4) = \hat{m}_1(\mu_1) \otimes \hat{m}_2(\mu_2)$
$= min(0.80, 0.90) = 0.80$

Finally, the mapping-fuzzy set $(\hat{\Omega}, \hat{m}) := [[Q]]_D^{\mathcal{F}}$ is:
$(\hat{\Omega}, \hat{m}) = \{$
$\{r \to$ dbr:Six_Weeks, $p \to$ dbr:Jon_Peters, $d \to$ dbr:Toni_Bill, $\hat{m} = 0.80\}\}$.

Theorem 2 *Given Q a SPARQL expression, D an RDF dataset, and $\hat{D} = (D, m)$ a fuzzy RDF dataset. Let $\Omega := [[Q]]_D$ and $(\hat{\Omega}, \hat{m}) := [[Q]]_D^{\mathcal{F}}$. Then, $\Omega = \hat{\Omega}$.*

The proof of Theorem 2 is presented in Appendix A. Theorem 2 states that the mapping set obtained when evaluating queries under set semantics is the same as when the evaluation is carried under fuzzy set semantics. Therefore, we can confirm that the same complexity results of EVALUATION [27, 30] apply when computing the solution mappings of queries under the proposed fuzzy set semantics.

Corollary 1 *The complexity of computing the mapping set of a SPARQL query under fuzzy set semantics is the same as when it is computed under set semantics.*

For the HARE query engine, we propose an efficient algorithm (Algorithm 2) that executes BGPs of SPARQL queries under fuzzy set semantics.

During query execution, the algorithm combines data from an RDF dataset D and a crowd knowledge base CKB that contains fuzzy sets of RDF data. In HARE, all triples in D are assumed to have membership degree equal to 1.0, since they are assumed to be correct. Algorithm 2 receives a plan $\mathcal{T}_B$, a decomposition SB_D and SB_{CROWD}, and a

threshold τ, provided by the user. The output of Algorithm 2 is a set of mappings $(\hat{\Omega}, \hat{m})$ that corresponds to the solution of BGP B. HARE physical join operators are implemented as extensions of the agjoin and anjoin [3] to process fuzzy RDF data.

Algorithm 2 first invokes (line 1) the HARE optimizer (cf. Section 8) to obtain query plan $\mathcal{T}_B$ and a decomposition SB_D and SB_{CROWD} as explained in . Sub-queries in $\mathcal{T}_B$ that are part of SB_D (denoted $\mathcal{T}_B|SB_D$) are executed against the dataset (Algorithm 2, line 2). Then, for each triple pattern t_{CROWD} in the plan that belongs to the partition SB_{CROWD}, denoted $\mathcal{T}_B|SB_{CROWD}$ (line 3), the algorithm checks whether the evaluation of t_{CROWD} instantiated with mappings μ in $\hat{\Omega}$ ($t = \mu(t_{CROWD})$) yields incomplete results. To do this, Algorithm 2 considers the completeness model and knowledge captured from the crowd (line 6). When the evaluation of t leads to incomplete answers, Algorithm 2 verifies if the crowd can provide the missing mappings (line 7). The probability of crowdsourcing the evaluation of t, denoted by $P_{CROWD}(t)$, is computed with the following formula:

$$P_{CROWD}(t) = \alpha \cdot \underbrace{(1 - Comp(t))}_{\text{Estimated incompleteness}} + (1-\alpha) \cdot \perp(\underbrace{\perp(m^+(t), m^-(t))}_{\text{Crowd confidence}}, \underbrace{\top(C(t), 1 - U(t))}_{\text{Crowd reliability}}) \quad (6)$$

Where:

- $\alpha \in [0.0, 1.0]$ is a score to weight the importance of the dataset completeness versus the crowd knowledge;
- $Comp(t)$ estimates the completeness of resources as of Definition 8 and Definition 10. Let $t = (s, p, o)$, $Comp(t)$ is as follows: if $vars(t) = \{s\}$, then $CompS_D(o|p) + CompS_{CKB}(o|p)$, if $vars(t) = \{o\}$, then $CompO_D(s|p) + CompO_{CKB}(s|p)$, if $vars(t) = \{p\}$, then $CompP_D(s|o) + CompP_{CKB}(s|o)$;
- $m^+(t)$ and $m^-(t)$ are the average membership degrees of t in CKB^+ and CKB^- as defined in Equation 4;
- $C(t)$ and $U(t)$ correspond to contradiction (cf. Equation 3) and unknownness (cf. Equation 5) levels exhibited by the crowd, respectively;
- $\top$ is a T-norm and $\perp$ a T-conorm to combine the values of crowd confidence and crowd reliability. We compute $\top$ as the Gödel T-norm, also called Minimum T-norm, which represents a weak conjunction of fuzzy sets. Analogously, $\perp$ is computed with the Maximum T-conorm, which represents a weak disjunction of fuzzy sets. HARE aims at crowdsourcing triple patterns where *CROWD* exhibits: i) high confidence values in positive or negative facts, i.e., $\perp(m^+(t), m^-(t))$; *or* ii) high levels of contradiction but low unknownness, i.e., $\top(C(t), 1 - U(t))$.

If $P_{CROWD}(t) > \tau$ holds, the engine invokes the microtask manager (cf. Section 7). Algorithm 2 terminates when all intermediate results are processed. We illustrate the execution of Algorithm 2 by evaluating the BGP B in query from Figure 5a against the DBpedia dataset D (partially depicted in Figure 3), where AMP_D(schema.org:Movie | dbp:producer) is 3, $\tau = 0.60$, and $\alpha = 0.50$. The plan $\mathcal{T}_B$ with intermediate results $\hat{\Omega}$ of SB_D are shown in Figure 5c. Triples previously collected from *CROWD* are shown below. For each mapping $\mu \in \hat{\Omega}$, Algorithm 2 (lines 4-8) proceeds as follows.

CKB^+:
(dbr:Tower_Heist, dbp:producer, dbr:Brian_Grazer, 0.90)
(dbr:The_Wolf_of_Wall_Street, dbp:producer, dbr:Leonardo_DiCaprio, 0.98)
(dbr:The_Sleeping_City, dbp:producer, dbr:Brian_Grazer, 0.12)

CKB^-:
(dbr:Tower_Heist, _:p1, dbr:Brian_Grazer, 0.04)
(dbr:Tower_Heist, dbp:producer, _:o, 0.06)

$CKB^{\sim}$:
(dbr:Tower_Heist, _:p2, dbr:Brian_Grazer, 0.01)

Iteration 1: An element of $\hat{\Omega}$ is selected, μ={movie → dbr:Tower_Heist}. The algorithm processes the triple pattern t_1 = (dbr:Tower_Heist, dbp:producer, ?producer), which is the result of instantiating μ in t. Given that MO_D(dbr:Tower_Heist | dbp:producer) = 0 (see Figure 3) and MO_{CKB}(dbr:Tower_Heist | dbp:producer) = 1, then $Comp(t_1) = 0.33$. Algorithm 2 computes the probability of crowdsourcing the triple pattern t_1 (line 7). The crowd knowledge bases CKB^+, CKB^-, $CKB^{\sim}$ have information about this triple pattern. As shown in Example 8 and Example 9, $C(t_1) = 0.094$ and $U(t_1) = 0.01$. The result of Equation 6 is $P_{CROWD}(t_1) = 0.78$, which is higher than $\tau = 0.60$, hence t_1 is crowdsourced.

Iteration 2: The next instance processed is μ={movie → dbr:The_Wolf_of_Wall_Street}. Assume that D has no producers for dbr:The_Wolf_of_Wall_Street. According to CKB^+, the multiplicity of this RDF resource for the property dbp:producer is MO_{CKB}(dbr:The_Wolf_of_Wall_Street | dbp:producer) = 1. The estimated completeness of this resource is 0.33<1 (Algorithm 2, line 6). Therefore the probability of crowdsourcing the pattern t_2=(dbr:The_Wolf_of_Wall_Street, dbp:producer, ?producer) is computed. Only CKB^+ contains triples associated with t_3, therefore $m^+(t_2) = 0.98$, while $m^-(t_2) = 0$ and $m^{\sim}(t_2) = 0$. Values of contradiction and unknownness are both zero for t_2. Lastly, the result of applying Equation 6 is $P_{CROWD}(t_2) = 0.82$, which is higher than $\tau = 0.60$, and t_2 is crowdsourced.

Iteration 3: The next element from $\hat{\Omega}$ is μ={movie → dbr:The_Sleeping_City}. Assume that D has no producers for the movie dbr:The_Sleeping_City, however, the multiplicity in CKB is MO_{CKB}(dbr:The_Sleeping_City | dbp:producer) = 1. The estimated completeness is in this case 0.33, then the algorithm processes t_3=(dbr:The_Sleeping_City, dbp:producer, ?producer). In this case, $m^+(t_3) = 0.12$, $m^-(t_3) = 0$, $m^{\sim}(t_3) = 0$. Values of contradiction and unknownness are both zero for t_3. Lastly, the result of applying Equation 6 is $P_{CROWD}(t_3) = 0.39$, which is lower than $\tau = 0.60$, and t_3 is not crowdsourced.

Iteration 4: In this iteration, the algorithm processes μ={movie → dbr:The_Interpreter}. According to Figure 3, the multiplicity value is MO_D(dbr:The_Interpreter | dbp:producer) = 3. In this case, $Comp$(dbr:The_Interpreter | dbp:producer) is 1.0 (line 6, Algorithm 2), then, this instance is not crowdsourced.

Note that the triple patterns t_2 and t_3 – processed in iterations 2 and 3 – share several commonalities: $Comp(t_2) = Comp(t_3) = 0.33$, $m^-(t_2) = m^-(t_3) = 0$, $m^{\sim}(t_2) = m^{\sim}(t_3) = 0$, $C(t_2) = C(t_3) = 0$, and $U(t_2) = U(t_3) = 0$. However, t_2 is submitted to the crowd, while t_3 is not crowdsourced. The reason for this is that *CROWD* exhibited low confidence when assessing dbr:The_Sleeping_City, therefore subsequent questions like t_3 about this resource are not posed against the crowd (for $\tau = 0.60$). On the contrary, since *CROWD* showed high confidence for dbr:The_Wolf_of_Wall_Street, then t_2 is crowdsourced. This illustrates the importance of taking into consideration the crowd confidence in Equation 6.

The configuration of the parameter τ allows for specifying the estimated completeness of the query answer. To illustrate this, consider the example

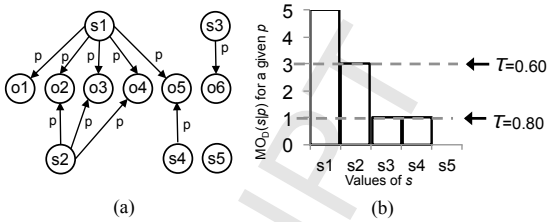


Figure 7: Effect of τ on the number of crowdsourced triple patterns. (a) Example of an RDF graph. (b) Distribution of values $MO_D(s|p)$ for each node in (a). When $\tau = 0.80$, only the pattern (s5,p,?o) is crowdsourced. When $\tau = 0.60$, patterns with predicate p and subjects s3, s4, s5 are crowdsourced.

shown in Figure 7 and $\alpha = 1$. Figure 7a depicts an RDF graph, where nodes are linked via the predicate p. Figure 7b presents the distribution of the multiplicity MO_D for nodes s1, s2, s3, s4, and s5. Whenever a user specifies $\tau = 0.80$, HARE crowdsources triple patterns whose estimated incompleteness is higher than 0.80, i.e., only the triple pattern (s5, p, ?o) is posed to the crowd. If $\tau = 0.60$, then (s3, p, ?o), (s4, p, ?o), and (s5, p, ?o) are crowdsourced, since their estimated incompleteness are 0.80, 0.80, and 1.0, respectively. The higher the value of τ, the lower the number of crowdsourced triple patterns.

Finally, Algorithm 2 combines in line 11 the mappings obtained from D (line 9) and mappings retrieved from the crowd stored in CKB^+ (line 10). The outcome of Algorithm 2 corresponds to the set of solutions $(\hat{\Omega}, \hat{m})$ of a BGP in Q, where each solution mapping is annotated with the membership degree $\hat{m}$ to $\hat{\Omega}$. Lastly, the HARE engine evaluates the rest of the operators in Q as specified in Definition 17.

The HARE engine does not increase the complexity of computing the result set of a SPARQL query Q. Note that, in comparison with a traditional SPARQL engine where a query is evaluated against an RDF dataset D, the HARE engine extends the evaluation of BGPs to incorporate the answers from the crowd, i.e., the query is evaluated using $D \cup CKB$. Formally, consider the SPARQL Evaluation problem [27, 30], we define the associated evaluation problem of executing a query against an RDF dataset D and the crowd knowledge base CKB, denoted by Evaluation$^{CROWD}(\mu, D, CKB, Q)$. Evaluation^{CROWD} is the problem of deciding if a mapping $\mu \in \hat{\Omega}$, where $(\hat{\Omega}, \hat{m})$ is computed by Algorithm 2 if Q is an expression composed of a triple

pattern or AND operators (Q is a BGP); otherwise $(\hat{\Omega}, \hat{m})$ is the result set of $[[Q]]^{\mathcal{F}}_{\hat{D}}$ as in Definition 17 with $\hat{D} = (D, m)$ and $m = 1.0$ for all $t \in D$.

Theorem 3 *The* EVALUATION^{CROWD} *problem is in (1)* PTIME *for expressions constructed using only* AND *and* FILTER*; (2)* NP*-complete for expressions constructed using* AND*,* FILTER*, and* UNION*; (3)* PSPACE*-complete for graph pattern expressions.*

Appendix B presents the proof of Theorem 3. With this theorem we have formally answered RQ1, i.e., portions of SPARQL queries can be completed without incurring additional time complexity. For answering RQ2 and RQ3 we have conducted an empirical study presented in the following section.

10. Experimental Study

Query Benchmark: We designed a benchmark of 50 queries[8] by analyzing triple patterns answerable by the DBpedia dataset (version 2014). We chose queries that do not return all possible results due to incomplete portions of DBpedia. The benchmark includes five categories with 10 queries each to study the crowd behavior across different domains: Sports, Music, Life Sciences, Movies, and History. Queries have between 3 and 6 triple patterns. The total number of query answers produced by DBpedia per knowledge domain is as follows:

Sports	Music	Life Sci.	Movies	History
125	116	304	1972	1299

We built a gold standard D^* of missing answers by removing portions of the dataset. Depending on the query, the gold standard contains between 8% and 97% of the query answer.

Implementation: HARE is implemented in Python 2.7.6. and CrowdFlower is used as the crowdsourcing platform. Initially, CKB is empty therefore we configure $\alpha = 1.0$ to consider only the completeness of the dataset. We implemented two variants of our approach which generate different microtasks: **HARE** that exploits the semantics of RDF resources as described in Section 7.1, and **HARE-BL** is a baseline approach that simply substitutes URIs with labels in the microtasks.

Crowdsourcing Configurations: *i) Task granularity*: We asked workers to solve a maximum of four RDF triples per task. *ii) Payments*: The monetary reward was 0.07 US dollars per task. *iii) Judgments*: We collected at least three answers per task. *iv) Gold Units (GU)*: GUs are verification questions to filter low-quality answers. In this work, the GUs were generated from the gold standard. The GU distribution was set to 10:90, i.e., for each 100 triples in the gold standard, 10 were GUs.

[8] https://sites.google.com/site/hareengine

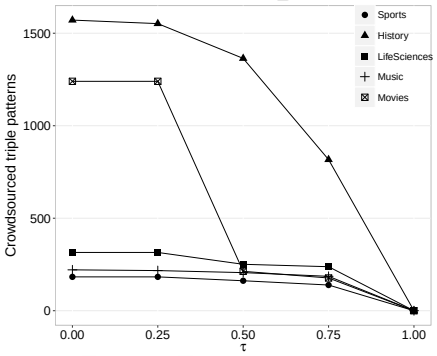


Figure 8: Per knowledge domain, number of crowdsourced triple patterns by HARE for different τ values. Benchmark queries produce different number of intermediate results across the domains which directly impacts on the number of crowdsourced triple patterns. In all domains, the number of crowdsourced triple patterns with τ=1.00 is zero; this corresponds to automatic query execution without crowdsourcing.

10.1. HARE Crowdsourcing Capabilities

We executed the benchmark queries with HARE to study its crowdsourcing capabilities. Given that CKB is initially empty, HARE solely relies on the estimated local incompleteness (computed by the completeness model) and the quality threshold τ (specified by the user) to submit a triple pattern to the crowd. We, therefore, measure the number of triple patterns that are crowdsourced when executing the benchmark queries with HARE for different values of the threshold τ. Figure 8 reports on these results aggregated per knowledge domain. It can be observed in Figure 8 that the number of crowdsourcing triple patterns differ per knowledge domain. In certain domains (such as History and Movies) the benchmark queries produce a large amount of results withe respect to queries for other domains. Figure 8 shows that in domains where queries produce large amount of results, HARE – based on the estimations of the completeness model – also crowdsources a large number of triple patterns. Moreover, Figure 8 also shows that the value

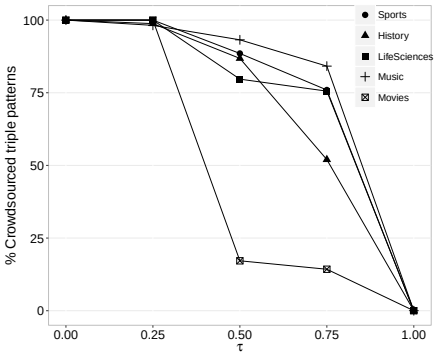


Figure 9: Per knowledge domain, effectiveness of the HARE completeness model with respect to the heuristics of the HARE optimizer. For τ>0.0, the completeness model is able to reduce the number of triple patterns to crowdsource in comparison to the optimizer.

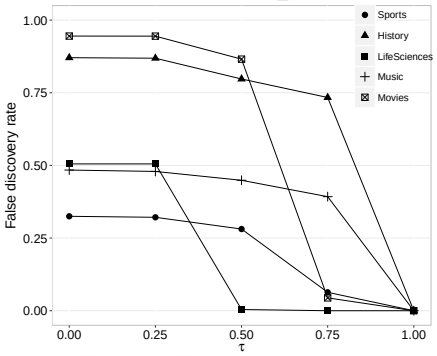


Figure 10: Per knowledge domain, upper bound of False Discovery Rate (FDR) achieved by HARE for different values of τ. A low number of FDR indicates that HARE crowdsources a low number of false positives. FDR is impacted by the number of query answers and τ.

of τ impacts the number of crowdsourced triple patterns. The higher the value of τ the lower is the requested completeness of the answer. As expected, the number of crowdsourced patterns decreases as the values of τ increases. When $\tau = 1.00$, no patterns are crowdsourced; this represents the case when query execution is carried out only against the dataset without invoking the crowd.

We now measure the effectiveness of the HARE completeness model in comparison to the simple heuristics of the HARE optimizer to identify missing values. In HARE, the optimizer considers the number of variables in triple patterns to decide which triple patterns should be crowdsourced during query execution. Since the optimizer does not take into consideration the completeness of resources, relying only on the optimizer could lead to submiting to the crowd large amount of unnecessary questions. To overcome this, HARE relies on the completeness model to decide which of the triple patterns identified by the optimizer are posed to the crowd. Based on the triple patterns identified by the heuristic of the optimizer, we measure the percentage of those triple patterns that are crowdsourced by HARE during query execution. Figure 9 presents these results. Note that $\tau = 0.0$ emulates the case in which HARE crowdsources the triple patterns following the heuristics of the optimizer, without considering the completeness model. For cases $\tau > 0.0$, the HARE engine relies on both the optimizer and the completeness model. We can observe in Figure 9 that when $\tau = 0.0$, HARE crowdsources 100% of the triple patterns identified by the optimizer, as expected. Furthermore, for $\tau \geq 0.50$, the completeness model is able to prune the triple patterns to submit to the crowd. In particular, in domains where the benchmark queries produce a large number of intermediate results, the completeness model reduces the number of crowdsourced triple patterns considerably. This can be observed, for example, in the domains History and Movies with $\tau = 0.75$, where the completeness model crowdsources around 52% and 14% (respectively) of the triple patterns.

Lastly, we contrast the number of missing values in the dataset and the number of crowdsourced triple patterns by HARE. In this setting, we define a *false positive* as a crowdsourced triple pattern for which the dataset produces at least one answer, i.e., the queried value is not completely missing in the dataset. This represents an upper bound of the number false positives produced by HARE.[9] We then measure the false discovery rate

[9]Note that, in practice, there might be real missing values that can be considered false positives with the given definition. For example, consider the movie dbr:Beauty_and_the_Beast_(2017_film). According to Wikipedia (As of June 1 2017), the infobox of this movie shows that the attribute "Screenplay by" has two values: Stephen Chbosky and Evan Spiliotopoulos. However, according to DBpedia (as of June 1 2017), for the property dbp:screenplay the movie only has one value: dbr:Stephen_Chbosky. For some given τ, HARE may crowdsource the triple pattern (dbr:Beauty_and_the_Beast_(2017_film), dbp:screenplay, ?x). As of our definition of false positives, crowdsourcing this triple pattern is a false positive, because the dataset produces one answer (dbr:Stephen_Chbosky). However, in reality, this triple pattern is not a false positive because the value of dbr:Evan_Spiliotopoulos is actually missing.

Table 2: Results when executing the benchmark with HARE-BL and HARE. Total number of crowdsourced triple patterns with each approach and answers retrieved from the crowd. Average and standard deviation of crowd workers' confidence as reported by CrowdFlower.

Knowledge Domain	# Triples to Crowd	# Crowd Answers	HARE-BL Worker Confidence	HARE Worker Confidence
Sports	69	376	0.93 ± 0.06	0.94 ± 0.06
Music	71	375	0.94 ± 0.06	0.95 ± 0.07
Life Sciences	82	460	0.90 ± 0.09	0.92 ± 0.07
Movies	120	1,035	0.88 ± 0.10	0.94 ± 0.06
History	160	917	0.90 ± 0.08	0.93 ± 0.07
Total	**502**	**3,163**	–	–

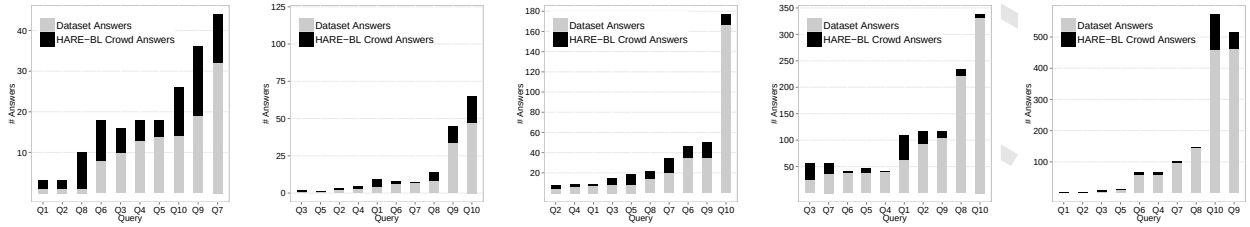
(FDR) defined as the number of false positives divided by the total number of crowdsourced triple patterns (which includes true positives and false positives). If the number of crowdsourced triple patterns is zero, then FDR is reported as zero. Figure 10 shows the FDR values achieved by HARE in different knowledge domains while varying τ. First, we can observe that FDR is impacted by both the number of answers produced by the queries and τ. In general, for knowledge domains where the queries produce a large number of results, FDR is high. For example, in queries about History and Movies, around 87% and 95% of the crowdsourced triple patterns correspond to false positives (as of our definition) for $\tau = 0.0$. Recall that $\tau = 0.0$ emulates the case in which HARE crowdsources the triple patterns following only the heuristics of the optimizer. This result indicates that solely relying on the simple heuristics of the optimizer is not enough to avoid crowdsourcing unnecessary triple patterns (false positives). Furthermore, for $\tau > 0.0$, the HARE engine also considers the completeness model. In these cases, we observe that FDR decreases while the values of τ increases, in particular for $\tau > 0.50$. These results demonstrate the effectiveness of the HARE completeness model.

10.2. Size of Query Answer

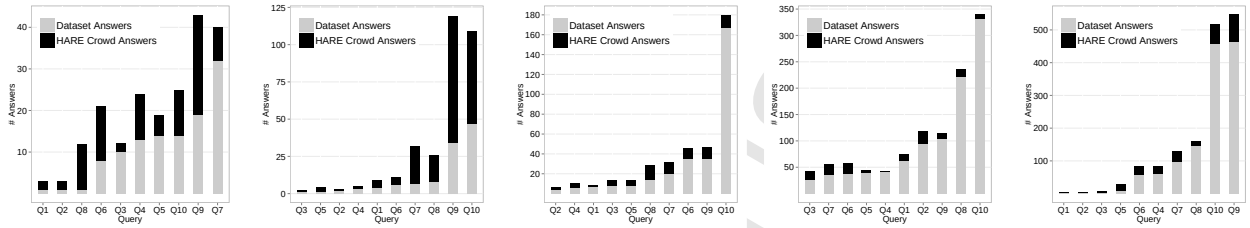
In this section, we compare the number of query answers obtained when the query is executed against the DBpedia dataset ($|[[Q]]_D|$) and with HARE, which combines results from the dataset and *CROWD* ($|[[Q]]_D^{CROWD}|$). For each benchmark query, we crowdsourced a random sample of triple patterns. The size of each sample is proportional to the percentage of missing values for which we the answers exist in the gold standard D^*. Table 2 reports on the number of crowdsourced triple patterns per knowledge domain. In total, we submitted to the crowd 502 RDF triple patterns with HARE and HARE-BL. First, we submitted the microtasks generated by HARE to CrowdFlower; after certain time, the microtasks generated by HARE-BL were crowdsourced under similar conditions. In total, we collected 3,163 crowd answers. Table 2 reports on the average and standard deviation of the crowd's confidence with each approach. Confidence[10] is reported directly by CrowdFlower and represents the validity of the crowd answer. Table 2 shows that the crowd's confidence is very high, indicating that most of the crowd answers are reliable. It is also important to note that there is no significant difference between the crowd confidence in both approaches; this indicates that crowd workers that solved tasks with HARE and HARE-BL are equally reliable.

Next, we analyze the number of answers produced by the dataset and the two variants of our approach: HARE-BL and HARE. Figure 11 lists the results per knowledge domain for each variant: the first row shows the number of answers obtained with HARE-BL, while the second row shows the results for HARE. In each query, we distinguish between the number of answers retrieved from the dataset and the ones obtained from the crowd. In addition, to measure the effectiveness of our approach, we compute the proportion of completeness (PC) per query. PC corresponds to the ratio of answers produced by HARE to the answers when the same query is executed only against the dataset, i.e., $PC = \frac{|[[Q]]_D^{CROWD}|}{|[[Q]]_D|}$. Minimum and maximum values of PC are reported per domain for HARE-BL and HARE in Figure 11.

[10] https://success.crowdflower.com/hc/en-us/articles/202703305#confidence_score



(a) HARE-BL - Sports. PC : (1.38, 10.00)
(b) HARE-BL - Music. PC : (1.00, 2.25)
(c) HARE-BL - Life Sci. PC : (1.06, 2.25)
(d) HARE-BL - Movies. PC : (1.02, 2.19)
(e) HARE-BL - History. PC : (1.00, 3.44)



(f) HARE - Sports. PC : (1.20, 12.00)
(g) HARE - Music. PC : (1.5, 4.57)
(h) HARE - Life Sci. PC : (1.08, 2.67)
(i) HARE - Movies. PC : (1.02, 1.65)
(j) HARE - History. PC : (1.10, 3.11)

Figure 11: Size of query answer (y-axis) achieved by baseline HARE-BL (first row) and HARE (second row) per query (x-axis) and domain. Benchmark queries (x-axis) are ordered by the number of answers produced when execution is carried over dataset. Crowd answers correspond to aggregated responses retrieved from crowd workers (including true positives and false positives). Minimum and maximum values of percentage of completeness (PC) are reported.

The results reported in Figure 11 indicate that the number on answers produced by the crowd with HARE is predominantly higher than with the baseline HARE-BL. Also, the values of PC achieved with HARE are always greater than 1.00 indicating that the crowd enhanced the number of answers of all benchmark queries. In contrast, HARE-BL was not able to enhance query answers for queries Q5-Music[11] and Q7-Music[12], and Q1-History[13]. Furthermore, for most queries, the values of PC are higher when microtasks were generated using the HARE approach. This suggests that crowd workers are more engaged to solve microtasks with semantically enriched interfaces; in addition, – as will be shown in Section 10.3 and Section 10.4– workers are also more effective and efficient when solving tasks generated with HARE than with the baseline. The only domain in which PC is lower in HARE than in HARE-BL is Movies. However, as discussed next in Section 10.3, the quality of the crowd answers obtained with HARE-BL are not as high as the crowd answer quality achieved with HARE.

Figure 12 depicts in more details the PC values achieved by HARE per knowledge domain. In all domains, the minimum PC values are higher than 1.0 indicating that HARE increased the number of answers in all SPARQL queries. It is important to highlight that PC values are affected by the esti-

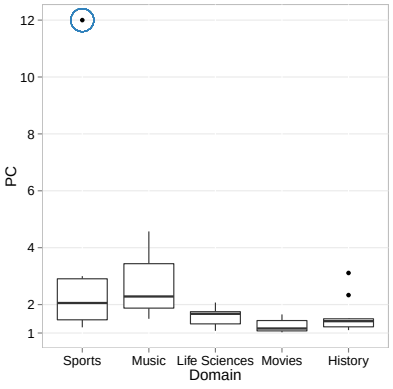


Figure 12: Portion of completeness (PC) achieved by HARE per knowledge domain. In all domains, HARE is able to enhance answer completeness on average. Highlighted value corresponds to query where HARE produced 12 times more answers than the dataset.

[11] Q5-Music: *Associated bands of Canadian Jazz Musicians*.
[12] Q7-Music: *Associated acts of Salsa Musicians*.
[13] Q1-History: *Places of British Military Occupations*.

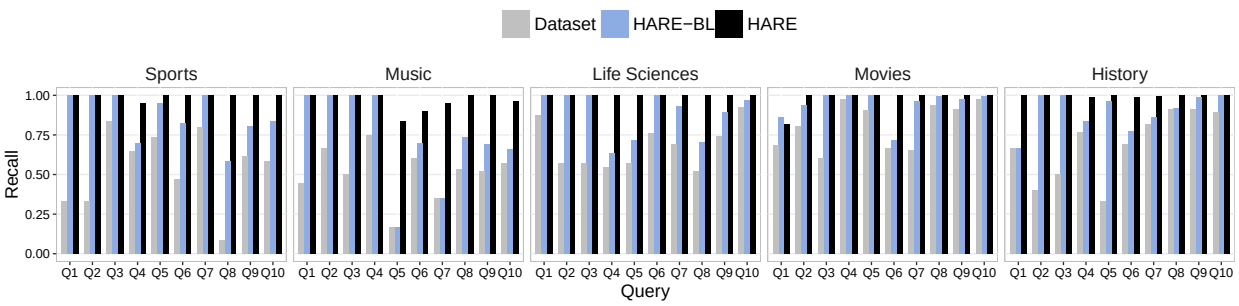


Figure 13: Recall: Query answer completeness (y-axis) obtained with DBpedia (Dataset) and our approaches HARE and HARE-BL per query (x-axis). HARE consistently outperforms the other approaches in all benchmarks queries.

mated completeness of the dataset. The more complete a dataset is, the smaller the opportunity to enhance query answer completeness. This is the case, for instance, in DBpedia in the Life Sciences and Movies knowledge domains, which exhibit high levels of completeness. Therefore, on average the *PC* values achieved in these domains are not as high as for other knowledge domains in DBpedia. For example, consider the query benchmark Q8-Sports where HARE is able to produce 12.00 times more answers than the DBpedia dataset (see highlighted datapoint in Figure 12), since Q8 only produces one result when it is executed against DBpedia.

Lastly, we report on the effectiveness of producing complete answers of the studied approaches. Effectiveness, in this case, can be measured as precision and recall of the answers obtained with the different approaches when compared to the gold standard D^*. However, note that the precision of the answers of executing queries against DBpedia is always 1.0, since we assume that the dataset is correct. Furthermore, the precision of the answers produced by HARE and HARE-BL is only impacted by the precision of the answers collected from the crowd (since HARE correctly produces the answers from the dataset). For this reason, we only analyze the recall values of the three approaches in this section. In Section 10.3, we present a detailed analysis of the quality of the crowd answers of HARE and HARE-BL. Figure 13 shows the recall values of the studied approaches per knowledge domain and queries. We can observe that the recall obtained when querying DBpedia varies among queries and knowledge domains. This indicates that completeness in DBpedia is heterogenous among different sub-graphs, in this case, represented by different knowledge domains. These results support the importance of taking into consideration the local completeness of resources. In contrast, HARE and HARE-BL are able to improve recall in comparison to executing queries automatically against DBpedia. This result indicates that our model is able to capture the skew distribution of values in real-world datasets. Among all the approaches, HARE consistently improves recall in all benchmark queries, even outperforming HARE-BL. This indicates that interfaces generated by HARE are able to gather more correct answers from the crowd than HARE-BL; this behavior is further analyzed in Section 10.3. When comparing the recall of HARE with respect to the dataset, we can conclude that the effectiveness of HARE on enhancing query answer size is independent from the local completeness of the dataset.

The experimental results presented in this section confirm that HARE correctly identifies sub-queries that produce incomplete results, and that microtask crowdsourcing can resolve missing values when executing SPARQL queries against RDF datasets. This answers our second research question RQ2.

10.3. Quality of Crowd Answers

To measure the quality of crowd answers, we compute precision and recall of the mappings retrieved from the crowd with respect to the gold standard D^*. For each query Q and crowdsourced triple pattern t in Q, a *true positive* corresponds to a mapping $\mu(t)$ provided by the crowd where $\mu(t) \in [[Q]]_{D^*}$. Analogously, a *false positive* is a mapping $\mu(t)$ from the crowd where $\mu(t) \notin [[Q]]_{D^*}$. Crowd answers equal to *"I don't know"* are considered neither true positives nor false positives, since the crowd has explicitly stated to be unknowledgeable about the existence of values for resolving t.

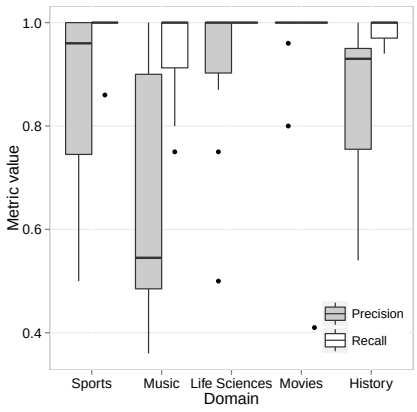


Figure 14: Precision and recall of crowd answers per domain. Median precision values is 0.55 in the Music domain and greater than 0.9 for the other domains. The median achieved in recall is 1.0 for all domains.

Figure 14 reports on the aggregated results of precision and recall of crowd answers obtained with HARE. It can be observed that precision values fluctuate over the knowledge domains. The lowest performance in terms of precision is obtained in the Music domain, where the median is 0.55. Still, the high value of the third quartile in the Music domain indicates that most of the precision values range from 0.55 to 0.90. Overall, the median precision values of HARE in the other domains are greater than 0.93. In turn, recall values are consistently high with median equal to 1.0.

Next, we conduct a fine-grained analysis of the quality of the crowd answers retrieved with HARE and the baseline HARE-BL. Results of precision and recall per query are reported in Table 3.

In terms of precision, the mean values reported in Table 3 indicate that the HARE approach led to higher precision than HARE-BL in four domains. With HARE the crowd was able to provide fully correct answers (precision equal to 1.00) for 25 out of 50 queries, while with HARE-BL only 13 queries were correctly answered. Furthermore, HARE achieves precision values from 0.62 up to 0.97, while HARE-BL precision ranges from 0.49 to 0.69. In 28 out of 50 benchmark queries, HARE outperforms HARE-BL in terms of precision. In 7 additional queries, HARE and HARE-BL exhibit the same performance. Out of the remaining cases, HARE-BL achieves higher precision than HARE in three queries which correspond to queries with multi-valuate attributes.[14] Still, in all the queries where HARE exhibits lower precision than HARE-BL, HARE leads to very high values of recall (from 0.95 to 1.00), indicating that the crowd is able to correctly identify true positives. It is important to note that with the HARE-BL approach, the majority of the crowd workers answered *"I don't know"* (N/A values in Table 3) in three benchmark queries. This provides evidence of the importance of our triple-based approach on the identification of portions of RDF graphs where the crowd is unknowledgeable. Thus, in subsequent requests, our approach will make use of this knowledge to avoid crowdsourcing these questions again.

In terms of recall, Table 3 shows that on average the quality of HARE is very high (from 0.92 to 1.00). In 49 out of 50 benchmark queries, HARE exhibits the same or better performance than HARE-BL, and in 32 queries HARE outperforms HARE-BL. Overall, the recall obtained with HARE is clearly higher than with HARE-BL. In particular, in 41 out of 50 queries, the crowd was able to resolve all missing values (i.e., recall equals to 1.00) with HARE. Only in Q1-Movies[15] the crowd achieved lower recall with HARE (recall 0.41) than HARE-BL (recall 0.55); nonetheless, in this case the precision of HARE (1.00) is higher than HARE-BL (0.34). It is important to point out that the recall values obtained with HARE-BL are heterogeneous within the knowledge domains. By contrary, with HARE the crowd is able to provide answers with high recall for queries in the studied domains.

In summary, the geometric mean values reported in Table 3 indicate that on average crowd answers exhibit higher quality with HARE than with the baseline HARE-BL in all studied knowledge domains. HARE microtasks assisted the crowd in reaching perfect precision and recall scores in 30 out of 50 SPARQL queries (60% of the benchmark). These experiments confirm that exploiting the semantics of RDF resources allows the crowd for effectively solving missing RDF values which, in turn, enhances the answer completeness of SPARQL queries. This answers research question RQ3 regarding the effectiveness of the crowd.

[14] This is the case of the following queries: Q8-Music *"Associated acts of German pop singers"*, Q9-Music *"Associated bands of Canadian Jazz Musicians"*, Q7-History *"Combatants of battles involving Portugal"*.

[15] Q1-Movies: *Gross of films shot in Spain.*

Table 3: Quality of crowd answers achieved by HARE and HARE-BL. Precision and recall values are reported for each query. Highlighted cells represent the cases where HARE exhibits a similar or better performance than HARE-BL. Precision equal to N/A corresponds to cases where the crowd answered *"I don't know"* in all query instances.

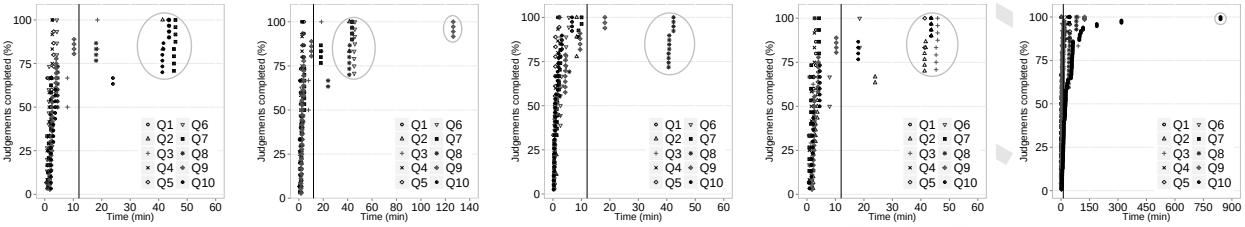
(a) **Precision** per query and geometric mean per knowledge domain

	Sports		Music		Life Sciences		Movies		History	
Query	HARE-BL	HARE	HARE-BL	HARE	HARE-BL	HARE	HARE-BL	HARE	HARE-BL	HARE
Q1	1.00	1.00	1.00	1.00	1.00	0.50	0.34	1.00	**N/A**	1.00
Q2	1.00	1.00	1.00	1.00	1.00	1.00	0.64	0.96	1.00	1.00
Q3	0.33	1.00	1.00	1.00	1.00	1.00	0.53	1.00	0.75	0.75
Q4	0.13	0.55	0.50	0.50	0.50	1.00	1.00	1.00	0.63	0.77
Q5	0.80	1.00	**N/A**	0.57	0.18	1.00	0.50	0.80	0.77	0.95
Q6	0.60	0.69	0.50	0.60	1.00	1.00	1.00	1.00	0.78	0.93
Q7	0.67	1.00	**N/A**	0.48	0.54	0.75	0.89	1.00	0.71	0.63
Q8	0.50	0.92	0.43	0.39	0.71	0.87	0.87	1.00	0.33	0.93
Q9	0.30	0.50	0.92	0.36	0.54	1.00	0.58	1.00	0.72	0.54
Q10	0.40	0.91	0.39	0.52	0.70	1.00	1.00	1.00	0.48	0.95
Mean	0.49	0.83	0.66†	0.62†	0.65	0.89	0.69	0.97	0.66†	0.81†

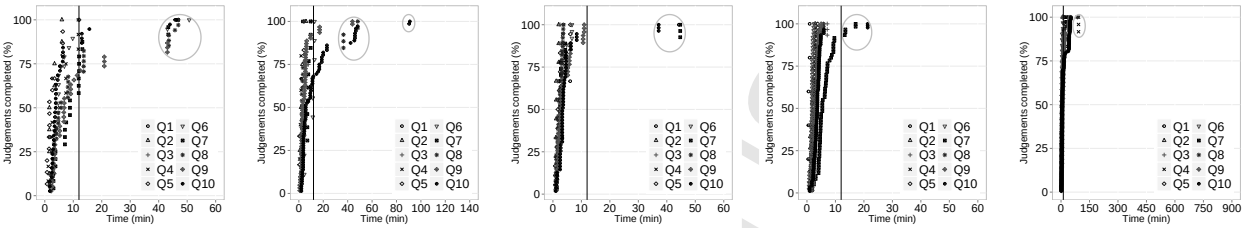
(b) **Recall** per query and geometric mean per knowledge domain

	Sports		Music		Life Sciences		Movies		History	
Query	HARE-BL	HARE	HARE-BL	HARE	HARE-BL	HARE	HARE-BL	HARE	HARE-BL	HARE
Q1	1.00	1.00	1.00	1.00	1.00	1.00	0.55	0.41	0.00	1.00
Q2	1.00	1.00	1.00	1.00	1.00	1.00	0.70	1.00	1.00	1.00
Q3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q4	0.14	0.86	1.00	1.00	0.20	1.00	1.00	1.00	0.28	0.94
Q5	0.80	1.00	0.00	0.80	0.33	1.00	1.00	1.00	0.94	1.00
Q6	0.67	1.00	0.25	0.75	1.00	1.00	0.16	1.00	0.27	0.96
Q7	1.00	1.00	0.00	0.92	0.78	1.00	0.89	1.00	0.24	0.95
Q8	0.55	1.00	0.43	1.00	0.38	1.00	0.87	1.00	0.07	1.00
Q9	0.50	1.00	0.35	1.00	0.58	1.00	0.70	1.00	0.84	1.00
Q10	0.60	1.00	0.20	0.91	0.54	1.00	0.88	1.00	0.98	1.00
Mean	0.67	0.98	0.54†	0.95†	0.60	1.00	0.70	0.92	0.46†	0.98†

† Geometric mean values computed excluding values N/A or 0.00 for HARE-BL and their corresponding pair for HARE.



(a) HARE-BL - Sports. Judgements (12'): 77%
(b) HARE-BL - Music. Judgements (12'): 79%
(c) HARE-BL - Life Sci. Judgements (12'): 92%
(d) HARE-BL - Movies. Judgements (12'): 79%
(e) HARE-BL - History. Judgements (12'): 62%



(f) HARE - Sports. Judgements (12'): 77%
(g) HARE - Music. Judgements (12'): 82%
(h) HARE - Life Sci. Judgements (12'): 97%
(i) HARE - Movies. Judgements (12'): 98%
(j) HARE - History. Judgements (12'): 75%

Figure 15: Crowd response time with HARE-BL (first row) and HARE (second row). The percentage of judgements completed (y-axis) in function of time (x-axis) is plotted per domain. Batches of judgements received last are highlighted. The percentage of judgements received until the 12th minute are reported per knowledge domain.

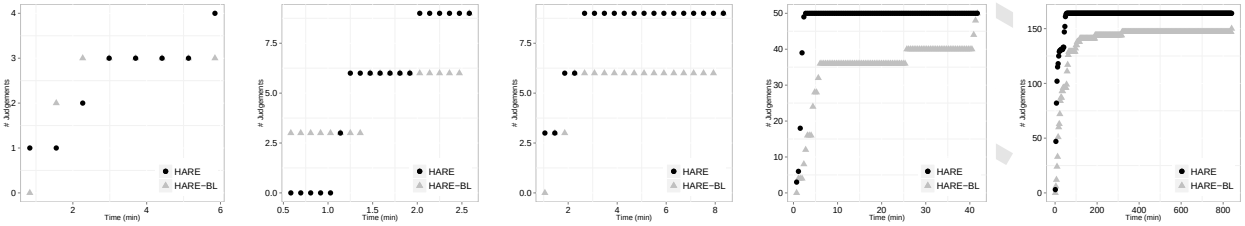
10.4. Crowd Response Time

We analyze the time efficiency of the crowd contacted by our approach when executing a query. Crowd response time per query corresponds to the elapsed time since the first task is posed to CrowdFlower until the last answer is retrieved from the crowd. Figure 15 lists the fraction of judgements (crowd answers) that were completed with HARE-BL and HARE as a function of time. For both studied approaches in all five domains, we observe a similar behavior: A small portion of judgements (highlighted in the plots of Figure 15) are finished much later than the vast majority.

Furthermore, in Figure 15 can be observed that, in general, the assignments are completed faster with the HARE approach. We therefore look at the percentage of judgements completed until a certain point in time with both approaches. For the HARE approach, at least 75% of the judgements are finished for all domains 12 minutes after the first task is released; the Movies domain exhibits the best observed scenario where 98% of judgements were finalized by this time with HARE. In the case of the HARE-BL approach, at the 12th minute, at least 62% of the judgements are finished for all domains. In particular, the crowd exhibits the best performance (in terms of time) with both approaches in the Life Sciences and Movies domains, achieving over 97% of the judgements with HARE. The slowest domain for both approaches is History, achieving 62% and 75% of the judgments by the 12th minute with HARE-BL and HARE, respectively.

In a subsequent step, we analyze the rate at which query answers are produced by the crowd with the HARE and HARE-BL variants. For each query, we compute the crowd answer distribution over time by sampling the number of judgements produced with each approach at different and identically distributed points in time. Examples of the obtained crowd answer distributions are plotted in Figure 16. In Figure 16a and Figure 16b, the answer distribution is very similar for HARE and HARE-BL, particularly for query Q2-Music (cf. Figure 16a), where several sampled points overlap in both approaches. By contrast, in Figure 16d and Figure 16e, the differences between the answer distribution with HARE and HARE-BL are notable.

In order to compare the answer distributions of both approaches, we conduct a statistical hypothesis test. We choose the nonparametric Kolmogorov-Smirnov [31] test since it is tailored to compare empirical distribution functions, in this case, of two samples. The null hypothesis H_0 in our study is that the answer distribution produced with HARE-



(a) Q2-Music. p-value : 1.000

(b) Q3-Life Sciences. p-value : 0.526

(c) Q2-Life Sciences. p-value : 6.621×10^{-5}

(d) Q9-Movies. p-value :$< 2.2 \times 10^{-16}$

(e) Q10-History. p-value :$< 2.2 \times 10^{-16}$

Figure 16: Crowd answer distribution over time. Number of judgements (y-axis) produced by the crowd with HARE and HARE-BL at different and identically distributed points in time (x-axis). p-values obtained with the Kolmogorov-Smirnov test [31] are reported. Answer distributions (a) and (b) are not significantly different; (c), (d), and (e) are significantly different ($p < 0.01$).

BL and HARE are *identical*; the alternate hypothesis H_a in our study states that the answer distribution produced with HARE-BL and HARE are *non-identical*. We conduct the test on all queries and report on the p-values obtained in Table 4. The results of the statistical test indicate that there is no significant difference among the answer distributions of HARE and HARE-BL mostly for selective queries such as Q2-Music and Q3-Life Sciences (cf. Figure 16a and Figure 16b). Nonetheless, for some of the selective queries, e.g., Q2-Life Sciences, the answer distribution with HARE and HARE-BL (as shown in Figure 16c) are nonidentical ($p < 0.01$). This indicates that the crowd answer rate with HARE and HARE-BL is still different when the number of judgements is low. In the case of non-selective queries, the crowd answer distribution obtained with the approaches HARE and HARE-BL are nonidentical ($p < 0.01$), as observed in Figure 16d and Figure 16e. In summary, the outcome of the statistical test confirms that the usage of semantics for generating microtasks impacts not only on the overall time of crowd response time, but also on the rate at which the answers are produced by the crowd. This answers RQ3 regarding the efficiency of the crowd.

As a final remark, it is worth mentioning that the crowd response time is not in the same order of magnitude as when queries are executed against a dataset. However, these experiments shed light on the trade-off between answer completeness and total execution time, whenever the proportion of completeness achieved by HARE is considered.

11. Conclusions and Outlook

This paper presents HARE, the first hybrid query engine over Linked Data to enhance the completeness of SPARQL query answers. HARE is able to execute SPARQL queries as a combination of machine and human-driven functionality. Our approach is tailored for RDF and Linked Data, i.e., data is assumed to be correct and potentially incomplete. No prior knowledge about the completeness of the data sources is expected from the users. HARE users may specify the desired level of query completeness and HARE handles the execution of queries and enrichment of the underlying data. No extensions to the SPARQL syntax are required.

Table 4: Statistical hypothesis test for crowd response time. p-values of applying the Kolmogorov-Smirnov test [31] to compare crowd answer distributions of HARE-BL and HARE. Values marked with *** indicate a difference significant at 0.01.

Query	Sports	Music	Life Sci.	Movies	History
Q1	0.056	0.054	0.056	$< 0.01^{***}$	0.056
Q2	$< 0.01^{***}$	1.000	$< 0.01^{***}$	$< 0.01^{***}$	0.270
Q3	$< 0.01^{***}$	0.270	0.526	$< 0.01^{***}$	$< 0.01^{***}$
Q4	$< 0.01^{***}$	0.336	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$
Q5	$< 0.01^{***}$	0.879	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$
Q6	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$
Q7	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$
Q8	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$
Q9	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$
Q10	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$	$< 0.01^{***}$

HARE implements the following novel features to improve the quality of SPARQL query answers: *i*) An RDF completeness model that relies on the topology of RDF graphs and the Local Closed-World Assumption (LCWA) to estimate the completeness of RDF resources. *ii*) Crowd knowledge bases (CKB) to store fuzzy RDF for modeling not only RDF *positive* facts (CKB^+), but also representing *negative* (CKB^-) and *unknown* statements ($CKB^\sim$). *iii*) A semantics-based microtask manager that makes use of Linked Data principles by dereferencing URIs to build user interfaces; the semantics of dereferenced URIs are exploited to properly render RDF resources in HTML interfaces. *iv*) A SPARQL fuzzy set semantics to represent the meaning of queries executed against fuzzy RDF datasets. *v*) A SPARQL query optimizer that implements techniques tailored for the topology of RDF graphs, and that generates hybrid bushy plans based on estimates about the completeness of RDF resources. *vi*) A SPARQL query engine that utilizes the RDF completeness model and the knowledge in CKB to decide on-the-fly which parts of a SPARQL query should resort to human computation.

We formally demonstrate that the time complexity of computing query results under the proposed fuzzy set semantics remains the same as when the computation is carried out under set semantics (Theorem 2). In addition, we also proved that computing the results of hybrid plans of SPARQL queries comes for free in terms of time complexity (Theorem 3). These theoretical results confirm that HARE is able to complete SPARQL query answers without adding complexity to the SPARQL Evaluation problem [27, 30], which answers our first research question RQ1.

We empirically measure the performance of HARE. First, we study the crowdsourcing capabilities of HARE and show the impact of varying the quality threshold τ on the number of crowdsourced triple patterns. Results also indicate that the number of intermediate results directly impact on the number of triple patterns submitted to the crowd. Also, we empirically show that the completeness model effectively reduces the number of false positives when crowdsourcing triple patterns for large τ. We then measure the number of answers produced by HARE and by the dataset. Our experiments confirm that HARE is able to *increase answer size* up to 12 times. We also observe that HARE consistently increases recall of query answers among all the benchmark queries. The results additionally show that the incompleteness degree vary notably among different sub-graphs (represented by knowledge domains) in DBpedia; values of recall confirm that our model is tailored for handling skewed value distributions in real-world datasets. This answers the second research question RQ2. In terms of quality, crowd answers have shown to be reliable with *precision* values from 0.62 to 0.97, while *recall* ranges from 0.92 to 1.00. Regarding efficiency, we observed that a large portion (up to 98%) of the human tasks submitted by HARE to the CrowdFlower platform are *finished* in less than 12 minutes. We statistically demonstrate that the distribution of crowd answers over time is significantly different ($p < 0.01$) when the interfaces are generated with and without semantics for non-selective queries. Our results show that exploiting the semantics of RDF resources can effectively increase the quality and efficiency of crowd answers; this answers our last research question RQ3. In summary, our empirical study shows that HARE implements a feasible solution to the studied problem.

In the future, we will concentrate on studying further approaches to accurately capture crowd answer *reliability*, i.e., to distinguish high quality workers from high confidence workers answers. We plan to extend the HARE techniques to pose, instead of triple-based, more complex microtasks against the crowd. Finally, we will consider other knowledge dimensions, besides contradiction and unknownness, to enhance the predictive power of HARE.

Acknowledgements

The authors would like to thank Rudi Studer for his valuable input. This work has been developed in the SemData project funded by the Marie Curie International Research Staff Exchange Scheme (IRSES).

References

[1] Crowdflower, http://crowdflower.com.

[2] Ziawasch Abedjan, Toni Grütze, Anja Jentzsch, and Felix Naumann. Profiling and mining RDF data with prolod++. In *ICDE 2014, Chicago, IL, USA, March 31 - April 4, 2014*, pages 1198–1201, 2014.

[3] M. Acosta, M.E. Vidal, T. Lampo, J. Castillo, and E. Ruckhaus. Anapsid: an adaptive query processing engine for sparql endpoints. In *ISWC2011*, pages 18–34, 2011.

[4] Maribel Acosta, Elena Simperl, Fabian Flöck, and Maria-Esther Vidal. HARE: A hybrid SPARQL engine to enhance query answers via crowdsourcing. In *K-CAP 2015, Palisades, NY, USA, October 7-10, 2015*, pages 11:1–11:8, 2015.

[5] Maribel Acosta and Maria-Esther Vidal. Networks of linked data eddies: An adaptive web query processing engine for RDF data. In *ISWC2015, Bethlehem, PA, USA, October 11-15, 2015*, pages 111–127, 2015.

[6] Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. Crowdsourcing linked data quality assessment. In *ISWC2013*, pages 260–276, 2013.

[7] Yael Amsterdamer, Susan B. Davidson, Tova Milo, Slava Novgorodov, and Amit Somech. OASSIS: query driven crowd mining. In *SIGMOD*, pages 589–600, 2014.

[8] Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, October 3-6, 2010*, pages 313–322, 2010.

[9] Jingwei Cheng, Z. M. Ma, and Li Yan. *DEXA 2010, Bilbao, Spain, August 30 - September 3, 2010, Proceedings, Part I*, chapter f-SPARQL: A Flexible Extension of SPARQL, pages 487–494. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[10] Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*, pages 1247–1261, 2015.

[11] G. Demartini, D. Difallah, and P. Cudré-Mauroux. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW2012*, pages 469–478, 2012.

[12] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Large-scale linked data integration using probabilistic reasoning and crowdsourcing. *VLDB J.*, 22(5):665–687, 2013.

[13] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610, 2014.

[14] Ju Fan, Meihui Zhang, Stanley Kok, Meiyu Lu, and Beng Chin Ooi. Crowdop: Query optimization for declarative crowdsourcing systems. *IEEE Trans. Knowl. Data Eng.*, 27(8):2078–2092, 2015.

[15] M. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: answering queries with crowdsourcing. In *SIGMOD*, pages 61–72, 2011.

[16] Olaf Hartig. Querying trust in rdf data with tsparql. In *ESWC 2009*, pages 5–20, Berlin, Heidelberg, 2009. Springer-Verlag.

[17] Patrick Hayes and Peter Patel-Schneider. RDF 1.1 semantics. W3C recommendation, W3C, February 2014. http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.

[18] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 2014.

[19] Adam Marcus, David R. Karger, Samuel Madden, Rob Miller, and Sewoong Oh. Counting with the crowd. *PVLDB*, 6(2):109–120, 2012.

[20] Adam Marcus, Eugene Wu, David R. Karger, Samuel Madden, and Robert C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.

[21] Adam Marcus, Eugene Wu, Samuel Madden, and Robert C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.

[22] Barzan Mozafari, Purnamrita Sarkar, Michael J. Franklin, Michael I. Jordan, and Samuel Madden. Scaling up crowd-sourcing to very large datasets: A case for active learning. *PVLDB*, 8(2):125–136, 2014.

[23] Felix Naumann. *Quality-Driven Query Answering for Integrated Information Systems*, volume 2261 of *Lecture Notes in Computer Science*. Springer, 2002.

[24] H. Park, R. Pang, A. G. Parameswaran, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: A System for Declarative Crowdsourcing. *PVLDB*, 5(12):1990–1993, 2012.

[25] Hyunjung Park and Jennifer Widom. Query optimization over crowdsourced data. *PVLDB*, 6(10):781–792, 2013.

[26] Hyunjung Park and Jennifer Widom. Crowdfill: collecting structured data from the crowd. In *SIGMOD*, pages 577–588, 2014.

[27] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3), 2009.

[28] C. Sarasua, E. Simperl, and N. Noy. CrowdMap: Crowdsourcing Ontology Alignment with Microtasks. In *Proceedings of ISWC2012*, pages 525–541, 2012.

[29] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the Linked Data best practices in different topical domains. In *ISWC*, pages 245–260. 2014.

[30] Michael Schmidt, Michael Meier, and Georg Lausen. Foundations of SPARQL query optimization. In *ICDT*, pages 4–33, 2010.

[31] Nickolay Smirnov. Table for estimating the goodness of fit of empirical distributions. *The annals of mathematical statistics*, 19(2):279–281, 1948.

[32] U. Straccia. *Foundations of Fuzzy Logic and Semantic Web Languages*. Chapman & Hall/CRC Studies in Informatics Series. CRC Press, 2013.

[33] Beth Trushkowsky, Tim Kraska, Michael J. Franklin, and Purnamrita Sarkar. Crowdsourced enumeration queries. In *ICDE*, pages 673–684, 2013.

[34] Maria-Esther Vidal, Edna Ruckhaus, Tomas Lampo, Amadís Martínez, Javier Sierra, and Axel Polleres. Efficiently joining group patterns in SPARQL queries. In *ESWC*, pages 228–242, 2010.

[35] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.

[36] Antoine Zimmermann, Nuno Lopes, Axel Polleres, and Umberto Straccia. A general framework for representing, reasoning and querying with annotated semantic web data. *Web Semant.*, 11:72–95, March 2012.

Appendix A. Proof of Theorem 2

Theorem 2 *Given Q a SPARQL expression, D an RDF dataset, and $\hat{D} = (D, m)$ a fuzzy RDF dataset. Let $\Omega := [[Q]]_D$ and $(\hat{\Omega}, \hat{m}) := [[Q]]^{\mathcal{F}}_{\hat{D}}$. Then, $\Omega = \hat{\Omega}$.*

Proof Let $\Omega := [[Q]]_D$ and $(\hat{\Omega}, \hat{m}) := [[Q]]^{\mathcal{F}}_{\hat{D}}$. We demonstrate that $\hat{\Omega} = \Omega$ by induction on the structure of Q. For the sake of readability, we denote $\mu \in [[Q]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in \hat{\Omega}$. It is important to highlight that for all $\mu \in \hat{\Omega}$ it holds that $\hat{m}(\mu) > 0$. This is guaranteed by the definition of $\hat{D}$, and the definition of operators $\otimes$ and $\oplus$ in Definition 16.

In the base case, Q is composed of a triple pattern tp. From Definition 17, it is obtained that $\hat{\Omega}$ and Ω are constructed in the same way, i.e., $\hat{\Omega} = \Omega$.

The induction hypothesis is $\mu \in [[Q']]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q']]_D$. We assume that the induction hypothesis holds for all SPARQL expression Q'. In the inductive case, Q is an expression composed of And, Union, Opt, Select, or Filter.

Case $Q := Q_1$ And Q_2.
We prove that $\mu \in [[Q_1 \text{ And } Q_2]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_1 \text{ And } Q_2]]_D$. By Definition 17, we obtain that $\mu \in [[Q_1 \text{ And } Q_2]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \bowtie [[Q_2]]^{\mathcal{F}}_{\hat{D}}$. According to Definition 16, $\mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \bowtie [[Q_2]]^{\mathcal{F}}_{\hat{D}}$ iff $\mu_1 \in [[Q_1]]^{\mathcal{F}}_{\hat{D}}$ and $\mu_2 \in [[Q_2]]^{\mathcal{F}}_{\hat{D}}$ for some μ_1, μ_2 such that $\mu_1 \sim \mu_2$ and $\mu = \mu_1 \cup \mu_2$. By induction hypothesis, it holds that $\mu_1 \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu_1 \in [[Q_1]]_D$ and $\mu_2 \in [[Q_2]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu_2 \in [[Q_2]]_D$. Since $\mu_1 \sim \mu_2$ and by definition of $\bowtie$ under set semantics (Definition 4), it holds that $\mu \in [[Q_1]]_D \bowtie [[Q_2]]_D$. With Definition 5 of the SPARQL semantics we have that $\mu \in [[Q_1 \text{ And } Q_2]]$. We conclude that $\hat{\Omega} = \Omega$, for this case.

Case $Q := Q_1$ Union Q_2.
We prove that $\mu \in [[Q_1 \text{ Union } Q_2]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_1 \text{ Union } Q_2]]_D$. We obtain that $\mu \in [[Q_1 \text{ Union } Q_2]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \cup [[Q_2]]^{\mathcal{F}}_{\hat{D}}$, applying Definition 17. According to the definition of $\cup$ in Definition 16, $\mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \cup [[Q_2]]^{\mathcal{F}}_{\hat{D}}$ iff $\mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}}$ or $\mu \in [[Q_2]]^{\mathcal{F}}_{\hat{D}}$. By induction hypothesis, it holds that $\mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_1]]_D$ or $\mu \in [[Q_2]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_2]]_D$. Applying the definition of $\cup$ under set semantics (Definition 4), we obtain $\mu \in [[Q_1]]_D \cup [[Q_2]]_D$. Lastly, applying the definition of the Union operator under set semantics Definition 5 it holds that $\mu \in [[Q_1 \text{ Union } Q_2]]$. We obtain that $\hat{\Omega} = \Omega$, for this case.

Case $Q := Q_1$Opt Q_2.
We prove that $\mu \in [[Q_1$Opt $Q_2]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_1$Opt $Q_2]]_D$. With Definition 17, it holds that $\mu \in [[Q_1$Opt $Q_2]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} :\bowtie [[Q_2]]^{\mathcal{F}}_{\hat{D}}$. Applying Definition 16, we obtain $\mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \bowtie [[Q_2]]^{\mathcal{F}}_{\hat{D}} \cup [[Q_1]]^{\mathcal{F}}_{\hat{D}} \setminus [[Q_2]]^{\mathcal{F}}_{\hat{D}}$. Here, we distinguish two sub-cases, where μ is generated by $[[Q_1]]^{\mathcal{F}}_{\hat{D}} \bowtie [[Q_2]]^{\mathcal{F}}_{\hat{D}}$ or by $[[Q_1]]^{\mathcal{F}}_{\hat{D}} \setminus [[Q_2]]^{\mathcal{F}}_{\hat{D}}$. In the first sub-case, the proof is the same as in the case of expressions. We conclude that $\mu \in [[Q_1$Opt $Q_2]]_D$, for the first sub-case. Lets now consider the second sub-case, i.e., $\mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \setminus [[Q_2]]^{\mathcal{F}}_{\hat{D}}$. With Definition 16, it follows that $\mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}}$ and there is no mapping $\mu_2 \in [[Q_2]]^{\mathcal{F}}_{\hat{D}}$ such that $\mu_1 \sim \mu_2$. By induction hypothesis, it is obtained that $\mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_1]]_D$. Lets assume by contradiction that $\mu_2 \in [[Q_2]]_D$ with $\mu_1 \sim \mu_2$. Applying the induction hypothesis again we obtain that $\mu_2 \in [[Q_2]]^{\mathcal{F}}_{\hat{D}}$; this contradicts our initial assumption about μ_2. By definition of $\setminus$ under set semantics, $\mu \in [[Q_1]]_D \setminus [[Q_2]]_D$. This demonstrates that $\mu \in [[Q_1 \text{ Opt } Q_2]]_D$, for the second sub-case. Finally, it is proved that $\hat{\Omega} = \Omega$, for this case.

Case $Q :=$ Select$_S(Q_1)$.
We prove that $\mu \in [[\text{Select}_S(Q_1)]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[\text{Select}_S(Q_1)]]_D$. By definition of Select under fuzzy set semantics (Definition 17) we obtain that $\mu \in [[\text{Select}_S(Q_1)]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in \pi_S([[Q_1]]^{\mathcal{F}}_{\hat{D}})$. With Definition 16, it holds that $\mu \cup \mu' \in [[Q_1]]^{\mathcal{F}}_{\hat{D}}$ and $dom(\mu) \subseteq S$ and $dom(\mu') \cap S = \emptyset$. By induction hypothesis, $\mu \cup \mu' \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \cup \mu' \in [[Q_1]]_D$. Given the characteristics of μ and μ' and by definition of π under set semantics, $\mu \in \pi_S([[Q_1]]_D)$. With the definition of Select under set semantics, we conclude that $\mu \in [[\text{Select}_S(Q_1)]]_D$ and $\hat{\Omega} = \Omega$, for this case.

Case $Q := Q_1$ Filter R.
We prove that $\mu \in [[Q_1 \text{ Filter } R]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q_1 \text{ Filter } R]]_D$ In this case, $[[Q_1 \text{ Filter } R]]^{\mathcal{F}}_{\hat{D}} := \sigma_R([[Q_1]]^{\mathcal{F}}_{\hat{D}})$ according to Definition 17. From Definition 16, it follows that $\mu \in \sigma_R([[Q_1]]^{\mathcal{F}}_{\hat{D}}) \Leftrightarrow \mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}}$ and $\mu \models R$. By induction hypothesis, it holds that $\mu \in [[Q_1]]^{\mathcal{F}}_{\hat{D}} \Leftrightarrow \mu \in [[Q]]_D$. Since $\mu \models R$, following the definition of σ_R under set semantics, we obtain that $\mu \in \sigma_R([[Q_1]]_D)$. Finally, $\mu \in [[Q_1 \text{ Filter } R]]_D$ and $\hat{\Omega} = \Omega$, in this case.

Appendix B. Proof of Theorem 3

Theorem 3 *The* Evaluation^{CROWD} *problem is in (1)* PTime *for expressions constructed using only* And *and* Filter*; (2)* NP*-complete for expressions constructed using* And*,* Filter*, and* Union*; (3)* PSpace*-complete for graph pattern expressions.*

Proof Let Q be a query, D an RDF dataset, and CKB a crowd knowledge base. Note that to solve the Evaluation$^{CROWD}(\mu, D, CKB, Q)$ problem, we just have to check if $\mu \in \hat{\Omega}$ where $(\hat{\Omega}, m)$ is computed either by Algorithm 2 or Definition 17 depending on the structure of Q. As defined in the Evaluation^{CROWD} problem, if Q is composed of a triple pattern or And operators, then Q is evaluated with Algorithm 2. The computation of the result set $\hat{\Omega}$ in Algorithm 2 is done

in four points of Algorithm 2: lines 2, 9, 10, and 11. In line 2, the engine evaluates the sub-query $\mathcal{T}_B|SB_D$ against D as defined in the SPARQL set semantics (Definition 5). Therefore, the time complexity of computing $\hat{\Omega}$ in line 2 is polynomial (cf. Theorem 1) w.r.t. the size of D and the number of triple patterns in $\mathcal{T}_B|SB_D$. Analogously, in line 9, the complexity of computing $\Omega_1 = [[t]]_D$ is also polynomial; more precisely, since t is a single triple pattern, $[[t]]_D$ can be computed in linear time w.r.t. the size of D. In line 10, Algorithm 2 computes $(\Omega_2, \hat{m}_2) = [[t]]^{\mathcal{F}}_{CKB^+}$. Based on Corollary 1, the time complexity of computing Ω_2 under fuzzy set semantics is the same as when using set semantics, i.e., Ω_2 can be computed in linear time w.r.t. the size of D (under the assumption that $|CKB^+| \ll |D|$). Lastly, in line 11, a SPARQL Union operator is added to the query evaluation. Note that the mapping sets Ω_1 (line 9) and Ω_2 (line 10) are the result of evaluating the same triple pattern in each one. Therefore, the problem of deciding whether $\mu \in \Omega_1 \cup \Omega_2$ in this case is in PTime [30].[16] We conclude that Evaluation^{CROWD} is in PTime for triple patterns or SPARQL expressions constructed with And operators. Regarding expressions constructed with other SPARQL operators, the Evaluation^{CROWD} problem specifies that Q is evaluated using fuzzy set semantics, i.e., $[[Q]]^{\mathcal{F}}_{\hat{D}}$ with $\hat{D} = (D, m)$ and $m = 1.0$ for all $t \in D$. By Corollary 1, the complexity of computing $\hat{\Omega}$ (the result set of $[[Q]]^{\mathcal{F}}_{\hat{D}}$) under fuzzy set semantics is the same as when it is computed under set semantics. In this case, the complexity of deciding if $\mu \in \hat{\Omega}$ is the same as in Evaluation.

[16] This is the case of the fragment $\mathcal{U}$ defined by Schimdt et al. [30]