A semantically enabled architecture for crowdsourced Linked Data management

Elena Simperl
Institute AIFB
Karlsruhe Institute of Technology
Germany
elena.simperl@kit.edu

Maribel Acosta
Institute AIFB
Karlsruhe Institute of Technology
Germany
maribel.acosta@kit.edu

Barry Norton
Ontotext AD
Bulgaria
barry.norton@ontotext.org

ABSTRACT

Increasing amounts of structured data are exposed on the Web using graph-based representation models and protocols such as RDF and SPARQL. Nevertheless, while the overall volume of such open, or easily accessible, data sources reaches critical mass, the ability of potential consumers to use them in novel applications and services is predicated on the availability of purposeful means to query and manage the data, while taking into account and mastering its essential features in terms of decentralization, heterogeneity of schema, varying quality, and scale. Many aspects of these challenges are necessarily tackled through a combination of algorithmic techniques and manual effort. In the literature on traditional data management the theoretical and technical groundwork to realize and manage such combinations is being established. In this paper we build upon these ideas and propose a semantically enabled architecture for crowdsourced data management systems which uses formal representations of tasks and data to automatically design and optimize the operation and outcomes of human computation projects. The architecture is applied to the context of Linked Data management to address specific challenges of Linked Data query processing such as identity resolution and ontological classification. Starting from a motivational scenario we explain how query-processing tasks can be decomposed and translated into MTurk projects using our semantic approach, and roadmap the extensions to graph-based data management technology that are required to achieve this vision.

1. INTRODUCTION

Linked Data refers to a set of guidelines and best practices for publishing and accessing structured data on the Web.[1] It builds upon established Web technologies, in particular HTTP and URIs, extended with Semantic Web representation formats and protocols such as RDF, RDFS, OWL and SPARQL, by which data from different sources can be shared, interconnected and used beyond the application scenarios for which it was originally created. RDF is a central building block of the Linked Data technology stack. It is a graph-based data model based on the idea of making statements about (information and non-information) resources on the Web in terms of triples of the form `subject predicate object`.[2] The object of any RDF triple may be used in the subject position in other triples, leading to a directed, labeled graph typically referred to as an 'RDF graph'. Both nodes and edges in such graphs are identified via URIs; nodes represent Web resources, while edges stand for attributes of such resources or properties connecting them. Schema information can be expressed using languages such as RDFS and OWL, by which resources can be typed as classes described in terms of domain-specific attributes, properties and constraints.[3] RDF graphs can be natively queried using the query language SPARQL.[4] A SPARQL query is composed of graph patterns and can be stored as RDF triples together with any RDF domain model using SPIN to facilitate the definition of constraints and inference rules in ontologies.[5]

Over the past years Linked Data has established itself as the de facto means for the publication of structured data over the Web, enjoying amazing growth in terms of the number of organizations committing to use its core principles and technologies for exposing and interlinking data sets for seamless exchange, integration and reuse.[6] More and more ICT ventures offer innovative data management services on top of Linked (Open) Data. A first notable example comes from the public administration sector, with a wide array of governmental institutions in many countries worldwide exposing their data according to open-access policies and Linked Data principles and technologies. Other equally promising application domains include life sciences, cultural heritage, and media and broadcasting, with the BBC, the New York Times and the Guardian adopting the same approach to publish metadata about their content in order to increase Web traffic, improve site navigation and facilitate the integration of self-owned and public data sets.

Per design Linked Data management solutions aim to reach a high level of automation with respect to the processing of an open and decentralized data space bringing together data sources published by different parties, of varying quality and using heterogeneous conceptual schemas and vocabularies. This is achieved by using a minimal set of established Web technologies known for their robustness and scalability, paired with a uniform data model

[1] `http://www.w3.org/DesignIssues/LinkedData.html`

[2] `http://www.w3.org/RDF/`
[3] `http://www.w3.org/TR/rdf-schema/`, `http://www.w3.org/TR/owl-ref/`
[4] `http://www.w3.org/TR/rdf-sparql-query/`
[5] `http://spinrdf.org/`
[6] See also recent statistics of the Linked Open Data Cloud at `http://www4.wiwiss.fu-berlin.de/lodcloud/`

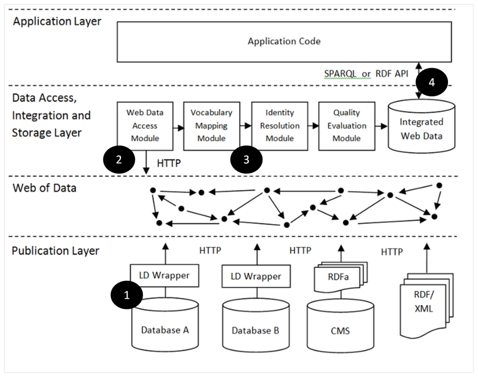

Figure 1: Generic architecture for applications consuming Linked Data (according to [5])

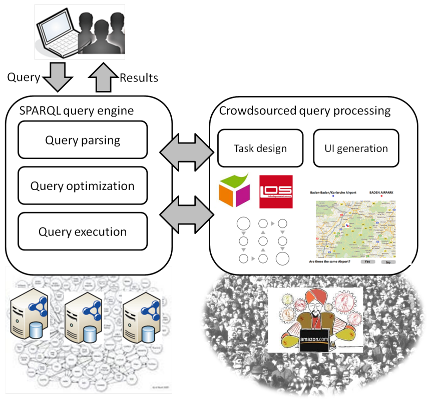

Figure 2: Overview of our approach

and links between datasets facilitating exploration and integration. Nevertheless, while the core technological building blocks of this approach are maturing, and large amounts of useful data are continuously made available on the Web, the actual experience in developing Linked Data applications reveals that many aspects of Linked Data management remain, for principled or technical reasons, heavily reliant on human intervention. Figure 1 shows a generic architecture of applications consuming data exposed using Linked Data principles and technologies [5]. The two layers that lend itself to LD-specific tasks are the publication layer, and the data access, integration and storage layer, respectively. Both contain various aspects amenable to crowdsourcing, as we will see later on: (i) publishing legacy data as RDF: identifying suitable vocabularies and extending them, conceptual modeling, defining mapping rules from the legacy sources to Linked Data; (ii) Web data access: discovering or adding missing information about a resource; (iii) Vocabulary mapping and identity resolution: defining correspondences between related resources (iv) Query processing over integrated data sets: aligning different styles or attributes of data integrated from heterogeneous data sources, detecting lack of knowledge. In these scenarios, human contributions can serve different purposes, from undertaking the task itself, to generating data to train specific automatic techniques, and validating (intermediary) outcomes of such techniques.

Recent approaches in the area of database management systems [1, 4, 9] are laying out the theoretical and technical foundations for designing and building crowdsourcing-enabled data management technology. In this paper we build upon these ideas and propose a semantically enabled architecture for crowdsourced data management systems which uses formal representations of tasks and data to automatically design and optimize the operation and outcomes of human computation projects. The architecture is applied to the context of Linked Data management to address specific challenges of Linked Data query processing such as identity resolution and ontological classification. Starting from a motivational scenario we explain how query-processing tasks can be decomposed and translated into MTurk projects using our semantic approach, and roadmap the extensions to graph-based data management technology that are required to achieve this vision.

2. CROWDSOURCING LINKED DATA MANAGEMENT

Our general idea to use microtask crowdsourcing for Linked Data management envisions the enhancement of core components of the architecture presented in Figure 1 with integrated crowdsourcing features that deal with the packaging of specific tasks as MTurk HITs, and with the integration of the crowdsourcing results with existing semi-automatic functionality. Figure 2 schematically depicts this idea for a query processing scenario, which is further discussed in Section 3. Dealing with MTurk includes user interface management capabilities, in order to produce optimal human-readable descriptions of specific tasks operating on specific data, to increase workers' productivity, and to reduce unintended behavior such as spam, but also workflow design capabilities for handling complex tasks. As described in [2] we propose to use SPARQL graph patterns to describe tasks and data to drive the generation of HITs interfaces, as well as extensions of Linked Data management languages and components, most prominently query processing, mapping and identity resolution, and Linked Data wrappers with crowdsourcing operators. The corresponding Linked Data components need to interact with the MTurk platform to post specific HITs, assess the quality of the outcomes, and exploit these in a particular application. This interaction can occur offline, when crowdsourcing input is being used to gradually improve the quality of computational methods, and online, which may require specific optimizations to predict the time-to-completion of crowdsourced tasks.

Each task that is subject to a crowdsourcing exercise is describes using graph patterns from the SPARQL query language representing input to the human task and its desired output. The input pattern expresses a query over the existing collection of Linked Data, necessary to provide the input to the MTurk task. The output pattern represents a query that should succeed when the additional assertions that result from the human task have been added to the knowledge. This is similar to the characterization of Linked Open Services [6] and Linked Data Services [11], and also the characterization of discovery goals in [8]. In [2] we discuss and give examples of such patterns for tasks such as identity resolution and ontological classification, which are key to crowdsourced query processing, as we will see in the next section.

The inclusion of commonly-used predicates in the SPARQL patterns used to describe tasks and data, predicates which have preconfigured screen representations in Linked Data browsers[7] means that tasks can be given HTML-based representations simply based

[7]For instance the retrieval of an object to a `foaf:depiction` predicate and rendering as an image, the mapping of geospatial `wgs84` predicates on a map.

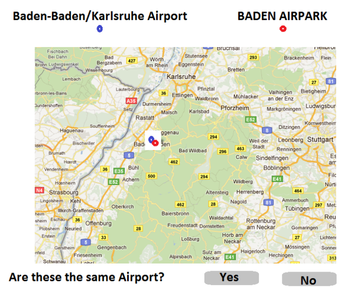

Figure 3: Interface for the identity resolution task generated from SPARQL descriptions

on existing technology.[8] This is a significant advantage of the Linked Data nature of the data sets being subjected to crowdsourcing in this approach, whereas questions related to the optimal phrasing of the task require verbalization techniques to provide a human-readable interface to graph-based data which is intended, at least in its original form, to machine-only consumption [3].

For illustration purposes, consider the case of identity resolution in Linked Data. Although identifiers for individual RDF resources can be directly reused in assertions in other data sets (usually as the object in triples), it is often the case that a new identifier scheme is created for an overlapping set of resources. Identity resolution is concerned with the definition of links, typically using predicates such as owl:sameAs or skos:narrower/broader to document correspondences and connections between different entities. Such links can be created by data providers or other parties manually, but more often they are subject to specific semi-automatic link discovery techniques, whose outcomes have to be enhanced through human input (used for training or validation purposes). The identity resolution of two types of entities (weather stations and airports) defined in two different Linked Data sets METAR (MÉTéorologique Aviation Régulière) and DBpedia[9] could be represented using SPARQL graph patterns as follows. The input graph: {?station a metar:Station; rdfs:label ?slabel; wgs84:lat ?slat; wgs84:long ?slong . ?airport a dbp-owl:Airport; rdfs:label ?alabel; wgs84:lat ?alat; wgs84:long ?along} stands for the query to be issued against the two data sets, whose results will form the data space to be subject to crowdsourcing. The output graph: {OPTIONAL {?airport owl:sameAs ?station}} expresses the condition which will be fulfilled when the crowdsourcing project is accomplished, in other words, when all airport instances in one data set will be matched to the corresponding weather stations in the other. Existing Linked Data browsing technology can be used to transform the graphs satisfying these two queries into an HTML page with side-by-side maps, showing the labels, with a button that will assert the sameAs link, and another which will not (hence the OPTIONAL keyword in the output graph). Filters can be used, for instance with geospatial extensions, to find proximal candidates. The user interface that could be formed from these graphs is shown in Figure 3.

In addition, the usage of this kind of descriptions forms the basis for the definition of semantic workflow management functionalities which could be used to handle the design of more complex tasks [10], though a more detailed analysis of typical patterns and their characteristics is needed in order to implement the approach. Additional optimizations, in terms of, e.g., the selection of tasks which need to be answered by the crowd, and cannot be inferred from existing answers are briefly discussed in the next section. The general line of reasoning is that by using a declarative approach to describing the overall crowdsourcing process, in particular SPARQL and other Linked Data formats and representations, we are able to resort to an established technology infrastructure consisting of query engines, reasoners and browsers to implement a large share of the envisioned crowdsourcing architecture. A second, equally important category of components is specific to the operation of the microtask platform, including quality assessment and resource management, where we would resort to existing approaches in the crowdsourcing literature.

3. CROWDSOURCING QUERY PROCESSING

In our approach both human and computational tasks can be described and executed using the same SPARQL-based language. A crowdsourcing-enabled SPARQL engine seamlessly combines information from both, and includes methods to decide for which types of queries and for which data it resorts to external human-based services, computes query plans taking into account performance estimates of these services, evaluates the outcomes, and integrates the correct ones into the overall query execution. In realizing this approach it is necessary to extend the algebra into which the query is translated to explicitly represent human tasks, and then make use of these extensions in deriving an appropriate evaluation strategy. While current triple stores have found little advantage from caching results between queries, this will be critical in implementing queries including human tasks. Nevertheless, experimental caching implementation can be reused to realize this. The real challenge, however, in caching human results is to account for updates to the underlying datasets. Where data loading, of multiple datasets, is explicit, materialization can be applied and the question is what degree of existing results can be carried over when a new version of a dataset is added. In the case of federation there are many open questions about versioning that the community has yet to answer.

As a running informal example we will use the provision of a Linked Data set of weather reporting (METAR) stations, and the provision of a Linked Service [7]. These are a superset of commercial airports, some of which are included in the popular Linked Data sets DBpedia and Geonames.[10]

3.1 Extensions to SPARQL and VoID

The RDF based schema to describe Linked Data sets is VoID ('Vocabulary of Interlinked Datasets').[11] Under this scheme, a data set is defined as a collection of data published and maintained by a single provider and it is annotated with the label `void:Dataset`. The interlinking of two different data sets is modeled with the label `void:Linkset`. A link set in VoID is a subclass of a data set, where each triple in the link set meets the following conditions: its subject is a resource hosted in one data set, its object is a resource hosted in another data set, and its predicate corresponds to the `void:linkPredicate` of the link set.

Example 1. (Specification of Data Sets) Getting back to our previous example, the following triple patterns correspond in VoID to the definition the data sets METAR and Geonames, and their inter-

[8]See also `http://km.aifb.kit.edu/sites/spark/`
[9]`http://dbpedia.org`

[10]`http://www.geonames.org`
[11]`http://semanticweb.org/wiki/VoiD`

linking with the specification of the linkset :METAR2Geonames through the predicate owl:sameAs.

```
:METAR rdf:type void:Dataset .
:Geonames rdf:type void:Dataset .

:METAR2Geonames rdf:type void:Linkset ;
    void:linkPredicate owl:sameAs ;
    void:target :METAR ;
    void:target :Geonames .
```

Based on the definition of link set, interlinking two data sets through any predicate can be performed automatically, by the implementation of the following restriction by a SPARQL engine.

```
CONSTRUCT {
    _:d1d2 a void:Linkset ;
           void:linkPredicate ?predicate ;
           void:target ?dataset1 ;
           void:target ?dataset2 .
} WHERE {
      ?x1 ?predicate ?x2 .
      ?x1 void:inDataset ?dataset1 .
      ?x2 void:inDataset ?dataset2 .
      FILTER (?dataset1 != ?dataset2) }
```

In order to support crowdsourcing tasks in queries that involve incomplete data or user-perceived comparisons, we will use an extension to the VoID vocabulary which specify which entities of a data set could be crowdsourced in terms of their domain-specific typing as classes and properties.

Example 2. (Crowdsourced class) The vocabulary VoID is extended with the label void:crowdClass to allow the creation of classes for which ontological classification of instances will be crowdsourced. Accordingly, all subclasses of the crowdClass are also defined (implicitly) as crowdsourced entities. The following example defines the class metar:Airport and its corresponding subclass `metar:CommercialHubAirport` of the data set METAR as classes with crowdsourced classification.

```
metar:Airport void:inDataset :METAR .
metar:CommercialHubAirport void:inDataset :METAR;
    rdfs:subClass metar:Airport .

metar:Airport rdf:type void:crowdClass .
```

Example 3. (Crowdsourced property) In a similar fashion, the instantiation of ontological properties can be crowdsourced with the VoiD extension `void:crowdProperty`. Analogously to classes, all subproperties of the `crowdProperty` are also subject to crowdsourcing. The following example defines the property gn:alternateName and its corresponding subproperty shortName of the data set Geonames as properties whose instantiation requires inputs from the crowd.

```
gn:alternateName void:inDataset :METAR .
gn:shortName void:inDataset :METAR ;
    rdfs:subProperty gn:alternateName .

gn:alternateName rdf:type void:crowdProperty .
```

Properties related to general ontology languages such OWL are treated as extensions of SPARQL operators, and are modeled in our architecture as tasks. The application of crowdsourcing to both class- and property-oriented instantiations could be further specified using formalisms such as SPIN in order to define specific types of instances or to constrain the cases in which crowdsourcing is assumed to be the most appropriate choice by design.

3.2 Crowdsourced query processing tasks

3.2.1 *Identity resolution*

Identity resolution involves the creation of `owl:sameAs` links, either by comparison of metadata or by investigation of links on the human Web. The following example illustrates a scenario where the crowd is needed to decide whether two resources refer to the same real-world entity.

Example 4. This SPARQL query retrieves the names (labels) of METAR stations that correspond to airports. In order to do this, the crowd should decide if a station refers to a given airport in real-world by the resolution of the `owl:sameAs` link.

```
SELECT ?label WHERE {
    ?station a metar:Station;
        rdfs:label ?label .
    ?airport a metar:Airport .
    ?station owl:sameAs ?airport .}
```

The execution of identity resolution tasks might be very expensive, in particular if the data set is large and all the identifiers should be compared. Consider the scenario where the data set is comprised of n identifiers and the crowd must compare only once all the resources by pairs, then, the total number of queries sent to the human web is $\binom{n}{2}$. In order to reduce this number, the proposed architecture uses the semantics of the resources by implementing in the framework SPIN a transitive property to infer facts automatically, instead of solving them by human intervention. This may reduce the number of human tasks up to 33%, in the best case.

3.2.2 *Ontological classification*

While the Semantic Web concentrated heavily on (OWL) ontologies, and therefore the classification of resources, Linked Data tends to shift the emphasis on the relationships between resources. Furthermore, due to the promoted use of generic vocabularies, is it not always possible to automatically infer classification from the properties used by applying established reasoning methods and techniques. Classification relates to (but is not subsumed by) the identity resolution task. For example, commercial airports can be identified and are partially classified in DBpedia and Geonames, other kinds of METAR station are not recorded or easily derivable and require human input.

Example 5. The following SPARQL query retrieves the names (labels) of METAR stations that correspond to commercial airports. In order to answer this query, first the stations are classified by the crowd in specific types of stations, e.g., commercial airport, private airfield, weather balloon, etc.

```
SELECT ?label WHERE {
    ?station a metar:CommercialHubAirport;
        rdfs:label ?label.}
```

Without using semantic technologies, a traditional crowdsourcing system should classified the data set by comparing each resource against the others. Assuming that the data set is comprised of n resources and they are compared by pairs and only once, then, the total number of comparisons performed by human tasks in a traditional system is $n \cdot (n-1)$. Our architecture is able to infer classification from the semantics of the data and incorporates the properties already solved by human intelligence in order to execute automatic inferences and reduce the number of tasks sent to the crowd.

3.2.3 Missing information

Due to the decentralized nature of the data publishing process in a Linked Data scenario, the resulting datasets are often of insufficient quality. A prominent example in this respect are missing links between entities, which may hamper the feasibility of the overall approach as an automated solution to large-scale data integration. A second, equally important category of incompleteness problems is related to the absence of natural-language labels of resources, which challenge the ways application developers interact with the data, and the generation of end-user interfaces for application consuming Linked Data. Missing information might be present in every aspect of a query processing scenario, thus, it not always feasible to handle it adequately. Nevertheless, translation tasks might be identified by the SPARQL keyword LANG.

Example 6. The following SPARQL query retrieves the names (labels) in German of METAR stations that correspond to commercial airports. In order to execute this query, the labels of airports that are not expressed in German language should be translated by human intervention.

```
SELECT ?label WHERE {
    ?airport a metar:CommercialHubAirport;
        rdfs:label ?label .
        FILTER (LANG(?label) = "de")}
```

3.2.4 Ordering

Despite that the ‘uniform data model’ in Linked Data is graph-like, and bears minimal order, having means to rank Linked Data content along specific dimensions is typically deemed useful for querying and browsing. This includes situations in which a specific ordering is imposed over the data; for instance, a temporal order defined via time stamps metadata, but also situations in which such orderings need to be defined via less straightforward built-ins; for instance, the ordering of pictorial representations of entities. The latter is symptomatic for the Web of Data, as in an open world there can be a number of alternative human-oriented representations of certain resources and their metadata, and many different ways to order these entities, for example, in a data set with n resources these ones can be ordered in $n!$ different ways. In some cases, there are relationships between resources that cannot be inferred by applying reasoning techniques or that correspond to subjective comparisons and should be solved by human input.

Example 7. The following SPARQL query retrieves all airports and their pictures, and the pictures should be ordered according to the more representative image of the given airport. In order to solve subjective comparisons, in this example of pictures, we propose a SPARQL extension where the order modifier `CROWD` is added to the `ORDER BY` clause.

```
SELECT ?airport ?picture  WHERE {
    ?airport a metar:Airport;
        foaf:depiction ?picture .
} ORDER BY CROWD(?picture,
"Most representative image for %airport")
```

Complementarily, specific types of similarity computations or partitions of the data that humans can deal with easily would be subject to crowdsourcing services, while the remaining task would still be addressed automatically. The proposed architecture is able to automatically solve tasks that can be inferred by the semantics of the resources. These computational tasks are expressed as SPARQL queries and are sketched in Table 1. The presented queries are not the only solution, there might be more sophisticated queries to perform more complex reasoning on the data, but with these simple restrictions expressed in SPIN, a significant number of facts can be automatically inferred without requiring human intervention.

Table 1: Computational tasks expressed as SPARQL queries

Task	Generalized SPARQL Query
Identity Resolution	`CONSTRUCT { ?a owl:sameAs ?c . } WHERE { ?a owl:sameAs ?b . ?b owl:sameAs ?c .}`
Classification	`CONSTRUCT { ?a a ?b . ?b rdfs:subClassOf ?c . } WHERE { ?a rdfs:subClassOf ?c . ?b rdfs:subClassOf ?b1 . ?b1 rdfs:subClassOf ?c .}`
Ordering	`CONSTRUCT { {(?a ?b) a rdf:List .} } WHERE { (?a ?x) a rdf:List . (?x ?b) a rdf:List .}`

3.3 Query parsing, optimization and execution

Since the proposed architecture is SPARQL-based, query plan generation and execution are similar to the ones performed by traditional SPARQL query engines. As shown in Figure 2, our architecture has a parser, optimizer and execution components. To illustrate the query processing steps, consider the following query.

Example 8. This SPARQL query retrieves the labels in german of commercial airports located in Baden-Württemberg, ordered by the better human-readable description of the airport given in the comment.

```
(1) SELECT ?label WHERE {
(2)     ?x a metar:CommercialHubAirport;
(3)       rdfs:label ?label;
(4)       rdfs:comment ?comment;
(5)       owl:sameAs ?y .
(6)    ?y geonames:parentFeature ?z .
(7)    ?z owl:sameAs
    <http://dbpedia.org/resource/Baden-Wuerttemberg> .
(8)    FILTER (LANG(?label) = "de")
(9)} ORDER BY CROWD(?comment,
     "Better description of %x")
```

When a SPARQL query is issued, the *parser* decomposes the input query in terms of the data sets that should be accessed to produce answers and rewrites the query based on this information. The query presented in Example 8 is decomposed into sub-queries by the parser as follows: graph-patterns (2)-(5) and (8) must be evaluated in METAR data set, while graph-patterns (6) and (7) must be evaluated in Geonames data set. The *optimization* module generates a logical plan using the data sets definition to determine which parts of the query should be solved by human input. In the running example, consider that the class metar:Airport is defined as a void:crowdClass, then a possible logical plan could be: first classify the existing METAR stations (variable ?x) as commercial airports to solve graph-pattern (2); solve graph-patterns (5) and (7), which correspond to identity resolution tasks of airports and locations, respectively; join the previous results to obtain the commercial airports located in Baden-Württemberg; to solve (8) the labels of the retrieved airports should be translated into German labels, if they are not; and finally the comments related to these airports should be ordered according to a subjective comparison in (9). Traditional databases optimization techniques such as predicate push-down and join-ordering can be applied. The optimized logical plan is translated into a physical plan, and it is evaluated by the *execution* module, which is able to automatically solve some parts of the query by posing SPARQL sub-queries against Linked Data sets or SPARQL endpoints, and by inferencing facts from the restrictions sketched in Table 1. Complementarily the query engine invokes the *crowdsourced query processing* component to generate tasks that should be performed by humans. These tasks are described as query patterns as explain in [2], as a basis for the creation of HTML interfaces for the corresponding HITs.

4. RELATED WORK

Combining traditional data management technology and crowdsourcing has recently received some attention in the area of databases [1], with approaches such as CrowdDB [4] and TurkDB [9] proposing extensions of established query languages and processing techniques to deal with the challenging inherently arising when dealing with less deterministic computational resources such as humans in environments with high performance and scalability constraints. By comparison our work specifically targets graph-based representation formats and protocols, in particular Linked Data, and proposes the usage of the same technologies, SPARQL, SPIN and Semantic Web vocabularies such as VoiD to induce crowdsourcing functionality to Linked Data query processing. This declarative approach facilitates the automatic design of HITs interfaces, and reduces the number of tasks which need to be executed manually by taking into account semantic properties of the data. There is an increasing body of research available that looks into methods and techniques to improve worker productivity and HITs design, with the most promising findings being published at the annual HCOMP workshop.[12] These results are complementary to our work, as they crowdsourcing-specific optimizations rather than data management-related ones.

5. OUTLOOK

In this paper we have outlined an architecture able to combine Linked Data management and human effort in order to improve the outcomes of crowdsourcing projects. This is achieved first by extending traditional data management technologies, through the replacement of existing components with human-augmented components. Then, we follow a declarative model where both human and computational tasks are described with semantic technologies.

We have shown for query processing scenarios how the well-defined structure and semantics of standard-compliant Linked (Open) Data sets, and the proposed small extensions to VoiD and SPARQL can support the automatic breakdown of a project and the generation of tasks for further crowdsourcing.

In the future we plan to semantically define workers and to extend the specifications of human tasks in order to incorporate resource management capabilities in the proposed architecture, optimizing the processes of tasks pricing and workers' assignment. Also, we are working towards a first release of a Sesame-based implementation of the query engine including the operators and functionality briefly introduced in the previous section.

[12]See `http://www.humancomputation.com/`

Acknowledgements

The research leading to this paper was partially supported by the Network of Excellence PlanetData, funded by the European Community's Seventh Framework Programme FP7/2007-2013 under the contract 257641.

6. REFERENCES

[1] A. Doan, R. Ramakrishnan, and A. Halevy. Crowdsourcing systems on the World-Wide Web. *Communications of the ACM*, 54:86–96, 2011.

[2] B. Norton E. Simperl and D. Vrandecic. Crowdsourcing tasks in Linked Data management. In *Proceedings of the 2nd workshop on consuming Linked Data COLD2011 co-located with the 10th International Semantic Web Conference ISWC 2011*, 2011.

[3] B. Ell, D. Vrandečić, and E. Simperl. Labels in the Web of Data. In *Proceedings of the 10th International Semantic Web Conference (ISWC2011)*. Springer, October 2011.

[4] M. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: answering queries with crowdsourcing. In *Proceedings of the 2011 International Conference on Management of Data SIGMOD 2011*, pages 61–72, 2011.

[5] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web Theory and Technology. Morgan & Claypool, 2011.

[6] B. Norton and R. Krummenacher. Consuming dynamic linked data. In *Proceedings of the First International Workshop on Consuming Linked Data COLD2010*, volume 665. CEUR-WS.org, November 2010.

[7] B. Norton and R. Krummenacher. Geospatial Linked Open Services. In *Proceedings of the Workshop Towards Digital Earth*, volume 640. CEUR-WS.org, September 2010.

[8] B. Norton and S. Stadtmüller. Scalable Discovery of Linked Services. In *Proc. 4th Intl. Workshop on REsource Discovery (RED 2011)*, volume 737. CEUR-WS.org, May 2011.

[9] A. Parameswaran and N. Polyzotis. Answering Queries using Humans, Algorithms and Databases. In *Conference on Inovative Data Systems Research CIDR 2011*, 2011.

[10] B. Norton R. Krummenacher and A. Marte. Towards linked open services and processes. In *Proceedings of the Future Internet Symposium FIS 2010*, pages 68–78, 2010.

[11] S. Speiser and A. Harth. Integrating Linked Data and Services with Linked Data. In *Proc. 8th Extended Semantic Web Conference ESWC2011*, volume 6643. Springer, June 2011.