Ontology Assisted Crowd Mining

Yael Amsterdamer[1], Susan B. Davidson[2], Tova Milo[1], Slava Novgorodov[1], and Amit Somech[1]

[1]Tel Aviv University, Tel Aviv, Israel
[2]University of Pennsylvania, Philadelphia, PA, USA

ABSTRACT

We present OASSIS (for Ontology ASSISted crowd mining), a prototype system which allows users to declaratively specify their information needs, and mines the crowd for answers. The answers that the system computes are *concise and relevant*, and represent *frequent, significant data patterns*. The system is based on (1) a generic model that captures both ontological knowledge, as well as the individual knowledge of crowd members from which frequent patterns are mined; (2) a query language in which users can specify their information needs and types of data patterns they seek; and (3) an efficient query evaluation algorithm, for mining semantically concise answers while minimizing the number of questions posed to the crowd. We will demonstrate OASSIS using a couple of real-life scenarios, showing how users can formulate and execute queries through the OASSIS UI and how the relevant data is mined from the crowd.

1. INTRODUCTION

Consider the following scenario: Ann is planning a vacation in New York City with her family. She is interested in finding combinations of popular child-friendly activities and a nearby restaurant to eat at afterwards, and related advice (e.g., whether to walk or rent a bike). She immediately thinks of two options: searching the web, or posting a question on some forum to receive input. However, both of these options have drawbacks.

Web search may return valuable information, but if Ann queries for child-friendly activities or for good restaurants she would still need to sift through the results to identify the appropriate combinations: not all good restaurants, even if child-friendly, are appropriate after a sweaty outdoor activity; a restaurant may be geographically close to some attraction but not easy to access; and so on. Moreover, much of the information is text-based so finding related advice (e.g., walk or bike) may be time consuming. Finally, the particular information she is looking for may not be recorded anywhere and obtaining it may require asking people. Alternatively,

Proceedings of the VLDB Endowment, Vol. 7, No. 13

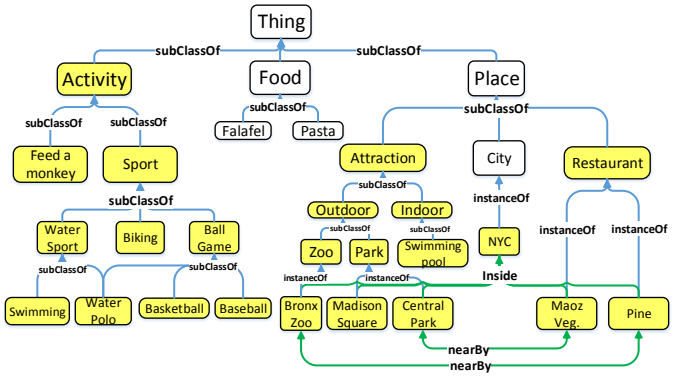

Figure 1: Sample ontology

Ann can post a question on a forum, which is more likely to yield detailed answers relevant to her question. However, she would again receive a number of (wide-ranging) text-based results, which she would then have to manually examine and aggregate, to extract the desired information.

We present OASSIS (Ontology ASSISted crowd mining), a prototype system which broadens traditional crowd-based data-sourcing by enabling users to declaratively specify their information needs, and obtain, using the crowd, *concise, relevant answers* that represent *frequent, significant patterns*.

Note that to answer Ann's question, one has to combine general, ontological knowledge (e.g., the locations of NYC attractions and restaurants) with personal, perhaps unrecorded information about people's habits (e.g., which are the most common combinations matching Ann's needs). This challenge arises in many other contexts where personal knowledge is required. E.g., a dietician may wish to study the culinary preferences in some population, focusing on food dishes rich in fiber. While nutritional facts can be recorded in a knowledge base, relevant habits of people may not be. Similarly, a medical researcher may wish to study the usage of particular ingredients for self-treatments of bodily symptoms, which also involves documented and undocumented data.

In order to retrieve and analyze such mixed data, we allow users to declaratively formulate their data needs in OASSIS-QL, a new query language that extends the query language SPARQL [9] with crowd mining features. OASSIS's friendly UI assists users in constructing queries quickly and easily, and then the system executes them while exploiting two types of data sources: an *ontology* and *crowd-provided*

data. The ontology is used to find candidate data patterns based on general knowledge, e.g., combinations of nearby attractions and restaurants. Then, the system automatically generates and poses questions to the crowd through a dedicated crowdsourcing platform, to identify which patterns are common or preferable. The question generation algorithm dynamically decides what to ask next by analyzing the collected data, to minimize the number of posed questions.

The ontology can be chosen out of many publicly available large knowledge bases.Consider the sample ontology in Figure 1, which illustrates facts by labeled nodes and edges. E.g., the fact "Maoz Veg. is nearby Central Park" is modeled by a "nearby" edge between the relevant nodes. Such an ontology can be used by the system for finding combinations of restaurants and attractions relevant to Ann's query.

Next, the system would mine the crowd in order to discover which combinations are indeed common, as well as related tips and advice. For instance, it may generate a *concrete question* asking crowd members whether and how often do they bike in Central Park. The system may pose such a question to several crowd members, aggregate the answers, and make further inferences based on *semantic dependencies* that we derive from the ontology. E.g., if bike riding in Central Park is infrequent, so must be sub-types of it, such as mountain-bike riding. To speed up data collection, the system may also ask *specialization* questions, where the crowd is asked to specify a relevant habit, related to the one in question. For example, it may ask "What else do you do in Central Park?" to find other prominent activities that are done together with biking in Central Park.

A full description of the formal model underlying `OASSIS`, as well as its query generation algorithm, are given in [2]. We briefly overview these in Section 2, to illustrate the operation of our prototype system. We will demonstrate `OASSIS` using the travel domain and culinary domain examples mentioned above, to show its applicability for various data domains where the crowd serves as a main source of knowledge. We will show how users describe their information needs in `OASSIS-QL` using a dedicated UI (and based on an ontology), and how the relevant data is mined from the crowd to answer the query. Audience members will be invited to actively participate in both query formulation and question answering. See details in Section 3.

Related Work. Crowd data-sourcing has attracted much research interest in the past few years (e.g., [6, 8, 10]). In this paper, we focus on the topic of crowd mining, which we recently introduced in [3]. We extend [3] in two ways: first, we allow users to define their information needs using a declarative query language; and second, we take advantage of semantic relations in order to compute a comprehensive yet concise set of data patterns. For the latter, we employ techniques developed in our previous work [1], but query-driven crowd mining is a new development. `OASSIS-QL` combines capabilities from SPARQL [9] for processing the RDF ontology, with ideas from data mining query languages such as DMQL [4], and enhances them to obtain a query language for crowd mining.

2. SYSTEM OVERVIEW

We next discuss the query language `OASSIS-QL` and our crowd mining algorithm, and detail the system architecture.

```
1  SELECT FACT-SETS
2  WHERE
3      {$w subClassOf* Attraction.
4       $x instanceOf $w.
5       $x inside       NYC.
6       $x hasLabel    "child-friendly".
7       $y subClassOf* Activity.
8       $z instanceOf  Restaurant.
9       $z nearBy      $x}
10 SATISFYING
11      {$y+ doAt    $x.
12       [] eatAt   $z.
13       MORE}
14     WITH SUPPORT = 0.25
```

Figure 2: Sample `OASSIS-QL` Query

2.1 Query Language

An `OASSIS-QL` query has three parts: 1) a `SELECT` clause which defines the output of the query; 2) a `WHERE` clause which is evaluated against the ontology; and 3) a `SATISFYING` clause, which defines the data pattern to be mined from the crowd. An example of a query $\mathcal{Q}$ is given in Figure 2, which expresses the scenario presented earlier: "Find a popular combination of activities in a child-friendly attraction in NYC, and a good restaurant nearby (plus other relevant advice)". The answer to $\mathcal{Q}$, in natural language, would include, e.g., "Go biking in Central Park and eat at Maoz Veg. (tip: rent the bikes at the Boathouse)", "Play ball games in Central Park and eat at Maoz Veg." and "Feed a monkey at the Bronx Zoo and eat at Pine Restaurant".

We use $\mathcal{Q}$ to illustrate the semantics of `OASSIS-QL`; full details of the language can be found in [2].

The `SELECT` clause (line 1) specifies that the output should be *significant fact-sets*, i.e., set of facts that co-occur frequently, in RDF format. (We note that the language further allows to project the query results only on certain facts or variables.) The `WHERE` clause (lines 2-9) defines a SPARQL-like selection query on the ontology. In short, it consists of a fact-set (here each fact is given in a separate line) over the ontology elements such as NYC or Activity. The facts contain variables (`$w`, `$x`, ...), and the selection returns all the variable bindings such that the resulting fact-set exists in the ontology. The SPARQL syntax also allows paths of relations to be specified. E.g., `subclassOf*` (line 3) defines a path of length zero or more of `subclassOf` relations connecting the elements.

The `SATISFYING` clause (lines 10-14) defines the data patterns (fact-sets) to be mined from the crowd along with a *support threshold*, which sets the minimal frequency of significant fact-sets (line 14). For example, the fact-set `[] eatAt $z. $y doAt $x` (we ignore, for now, `MORE` and the + next to `$y` in Figure 2) combined with the binding φ`($z) = Maoz_Veg.`, φ`($y) = Sport`, and φ`($x) = Central_Park` corresponds to the concrete crowd question "How often do you eat at Maoz Veg. while also doing a sport in Central Park?". `$y+` specifies, intuitively, that we are also interested in multiple values (in this case multiple activities) that co-occur together, e.g., Biking and Baseball. Finally, the `MORE` keyword is used as syntactic sugar for a set of zero or more unrestricted facts, which captures related advice. To set the threshold, one can convert the user answers to support values. For example, we can configure the UI to let crowd members choose their habit frequency out of *Never*,

Rarely, *Sometimes*, *Often* or *Very Often* with imputed support values 0, 0.25, 0.5, 0.75, and 1 resp. Answers from different crowd members to the same question are *averaged*, and thus, a threshold of, e.g., 0.25 would imply that significant fact-sets occur at least rarely for the average user.

The output of `OASSIS-QL` is defined to be *semantically concise*, meaning that only the most specific among the significant fact-sets are returned. The notion of "more specific" is made precise using the ontology subclassOf and instanceOf relations, based on which a semantic subsumption partial order is defined over fact-sets. For example, Ball_Game doAt Central_Park is more specific than Sport doAt Central_Park since Ball_Game is a sub-class of Sport. It is also more specific than Ball_Game doAt Park since Central_Park is an instance of Park. Hence, if Ball_Game doAt Central_Park is in the query results, the more general Ball_Game doAt Park, Sport doAt Central_Park, Sport doAt Park, etc. should be excluded, even though they must also be frequent.

2.2 Crowd Mining Algorithm

We sketch the main principles guiding the algorithm for dynamic question selection, described in detail in the full version [2]. The algorithm is used to evaluate the `SATISFYING` clause by posing questions to crowd members. It aims to minimize the number of questions while providing a pleasant user experience, based, respectively, on theoretical results [2] and on preliminary user studies we have performed. In particular, several factors guide the order in which questions are asked. First, it is natural to start by asking general questions that identify the person's areas of interest. For example, it natural to ask someone if they play sports before asking whether or not they play basketball. Second, if a person indicates low support for a fact-set, then questions about more specific fact-sets do not need to be asked. For example, if a person never goes to Central Park, it is useless (and very irritating) to ask them whether they play basketball in Central Park. Third, users prefer to answer a sequence of related questions rather than unrelated ones. For example, if a person indicates that they frequently play sports in a park then a natural follow-up question would be "Do you play sports in Central Park?" rather than "Do you feed monkeys at the Bronx Zoo?"

Recall that the ontology is used to define a partial order of fact-sets. This order is traversed "top-down" in our evaluation algorithm, generating a queue of questions for each person. When a person indicates that a binding is not frequent, bindings that are more specific are eliminated from the person's queue since they cannot be frequent for that person. Questions may also be removed from a person's queue if a binding is "resolved" as either relevant or irrelevant to the query result by other crowd members.

Answering a query involves posing questions to many crowd members. First, we want to guarantee that the answer is significant in the population and not just for a single member. In our implementation, we ask 5 people per binding and average their answers. More generally, one can use any function to determine the number of users to be asked and how to aggregate their answers, including existing techniques for error estimations, spam filtering and outlier detection, to ensure high-quality results [2]. Second, several users may be needed to answer the many questions posed during query evaluation. Finally, multiple users can work in parallel and speed up the evaluation process.

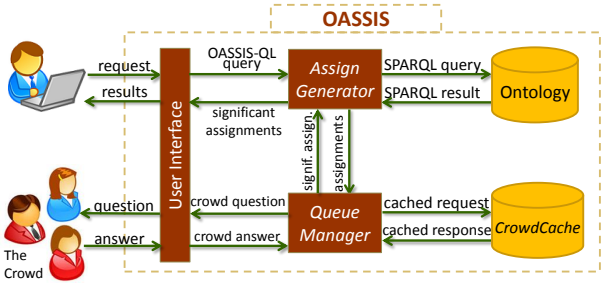

Figure 3: `OASSIS` architecture

So far, we have only discussed *concrete* questions. However, to speed up query processing, `OASSIS` can leverage answers to *specialization* questions. For example, if it is known that sports are often played in Central Park and we wish to determine which particular sports are played, the crowd could be asked "What type of sport do you do in Central Park and how frequently?". They could then choose, e.g., Water Sport, Biking or Ball Game, along with the frequency indications as before. Since many ontologies have a large fan-out per node, and a user typically only does a few things frequently, this information can be used to find significant assignments more quickly. It has been observed in previous work that interleaving both types of questions is beneficial: while specialization questions allow us to quickly identify prominent, significant fact-sets, people cannot spontaneously recall all of their data; whereas concrete questions allow us to dig deeper into their memory [3].

2.3 Implementation

`OASSIS` is implemented in Python 2.7 and uses a MySQL 5.6 database. External libraries used include RDFLIB for handling RDF data and NetowrkX for constructing the fact-sets partial order.[1] The system architecture is depicted in Figure 3. The user submits requests via a user-friendly query builder with auto-completion suggestions, through which a query $\mathcal{Q}$ can be quickly and accurately constructed (Figure 4). This UI allows the user to choose categories of interest (e.g., outdoor places one goes to), desired properties (e.g., selecting only child-friendly places), connections between categories (e.g., an outdoor place that is nearby a restaurant) and so on. An advanced user can construct more sophisticated queries via a text-based `OASSIS-QL` editor, also with auto-completion capabilities, to take advantage of the full expressive power of the query language. $\mathcal{Q}$ is then passed to the *AssignGenerator* module, which computes assignments to the query variables based on the ontology and passes them to the *QueueManager* module. *QueueManager* generates relevant questions for each crowd member. We explain the interaction with the system in the next section. The crowd's input is used to compute answers to $\mathcal{Q}$, which are translated by the `OASSIS` engine into a user-friendly form and returned to the user incrementally. The web-based front-end for the demo is developed in PHP 5.3.

`OASSIS` uses two repositories: *Ontology* and *CrowdCache*. *Ontology* stores the ontology; the one used for the demo is constructed from WordNet [7], YAGO [5], and data obtained

[1] `www.rdflib.net` and `networkx.github.io`.

Figure 4: Query builder screen

Figure 5: A question to the crowd

via the Foursquare API[2]. In general, any third-party RDF ontology can be used. *CrowdCache* is a database that stores the computed variable assignments along with the answers collected from the crowd for each assignment.

3. DEMONSTRATION

We demonstrate `OASSIS` and its UI for two different domains, mentioned earlier, where both ontological and human knowledge are required: the travel domain and culinary domain. We show how users can describe their information needs using the `OASSIS-QL` user-friendly query builder, execute the formulated query and browse the results. To assist in computing the query answer, the crowd is engaged to contribute information via a social questions-game where they are asked query-relevant questions about their habits, the frequency in which they do certain activities and combinations thereof. Users are awarded stars (bronze, silver and gold) as they answer questions, and can use them as virtual money either to pose queries to the system or to view suggestions computed in response to pervious queries.

To interact with the crowd, questions are phrased in pseudo-natural language using the ontology terms (see Figure 5). For each term type in the ontology, an appropriate phrase/template is kept and instantiated at run time with the concrete term being asked about. The same templates are used for presenting the query results in a pseudo-natural language.

As a preparatory step to the demonstration, we will initialize the system with a few sample queries, and collect data from our local crowd through the game mentioned above. This repository will later be extended with data provided by conference participants.

We will use three screens displaying different aspects of `OASSIS`. The first will be used to show how `OASSIS-QL` queries are formulated, the second will demonstrate how the crowd is engaged to answer questions relevant to the query, and the third will run in administrator mode to show the underlying operation of the system. We will start the demonstration by explaining the system, its interface and the goal, and showing the queries already fed to the system. We will then let our audience play the game from their own laptop/tablet/smartphone or using one of our laptops, by accessing the `OASSIS` web crowd interface. In parallel, we will show, on an administrator screen, the current state of the system and the data: what `OASSIS-QL` queries were recently posed to the system, which answers were collected for which crowd questions, what fact-sets have already been found to be (in)significant and which questions will be posed to the crowd next, highlighting how the underlying algorithms operate. Finally, we will ask one of the attendees who recently contributed to the system (by being part of the crowd) to use the query builder and pose some new query to the system via the `OASSIS-QL` user interface. We will examine the questions that `OASSIS` consequently generates and poses to the crowd, and again reveal what is happening under the hood on the administrator screen.

[2]Foursquare API. `developer.foursquare.com/`.

Acknowledgements. This work has been partially funded by the European Research Council under the FP7, ERC grant MoDaS, agreement 291071, by the NSF grant III-1302212 and by the Israel Ministry of Science.

4. REFERENCES

[1] A. Amarilli, Y. Amsterdamer, and T. Milo. On the complexity of mining itemsets from the crowd using taxonomies. In *ICDT*, 2014.

[2] Y. Amsterdamer, S. B. Davidson, T. Milo, S. Novgorodov, and A. Somech. OASSIS: query driven crowd mining. In *SIGMOD*, 2014.

[3] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart. Crowd mining. In *SIGMOD*, 2013.

[4] J. Han, Y. Fu, W. Wang, K. Koperski, O. Zaiane, et al. DMQL: A data mining query language for relational databases. In *SIGMOD*, volume 96, 1996.

[5] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artif. Intell.*, 194, 2013.

[6] A. Marcus, D. Karger, S. Madden, R. Miller, and S. Oh. Counting with the crowd. In *VLDB*, 2012.

[7] G. A. Miller. WordNet: a lexical database for English. *Comm. ACM*, 38(11), 1995.

[8] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM*, 2012.

[9] E. Prud'Hommeaux, A. Seaborne, et al. SPARQL query language for RDF. *W3C rec.*, 15, 2008.

[10] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*, 2013.