CrowdQ: Crowdsourced Query Understanding

Gianluca Demartini[†] Beth Trushkowsky Tim Kraska[◊] Michael J. Franklin

AMPLab, UC Berkeley
{trush,franklin}@cs.berkeley.edu

U. of Fribourg–Switzerland[†]
gianluca.demartini@unifr.ch

Brown University[◊]
kraskat@cs.brown.edu

ABSTRACT

Work in hybrid human-machine query processing has thus far focused on the data: gathering, cleaning, and sorting. In this paper, we address a missed opportunity to use crowdsourcing to understand the query itself. We propose a novel hybrid human-machine approach that leverages the crowd to gain knowledge of query structure and entity relationships. The proposed system exploits a combination of query log mining, natural language processing (NLP), and crowdsourcing to generate query templates that can be used to answer whole classes of different questions rather than focusing on just a specific question and answer.

1. INTRODUCTION

Structured queries are beneficial because they allow for query optimization as well as provide a clear expectation for answers in the result set. However, users typically specify their information needs as unstructured keyword queries [8], particularly in search engines. Efforts in deep-web search and crowdsourced query processing in databases aim to bridge this gap. Examples include CrowdDB [5] and Qurk [6] in traditional database systems, and [3], [1], and [4] in the context of search engines.

However, these existing hybrid human-machine approaches have so far been focused on the *data*. In other words, the crowd has been used to gather, clean, and sort data in response to an individual information request; the data for the query is subsequently stored for future use. We believe there is a missed opportunity to use crowdsourcing to instead target *query semantics*. This opportunity is particularly advantageous in the context of search engines, where unstructured keyword queries are common. Understanding query semantics allows queries with similar structure to be grouped, and this structure can be utilized to automatically provide direct answers. Thus we can go beyond merely caching the results of one crowdsourced query at a time, and instead amortize the cost of using the crowd over many queries. Knowledge of query structure, in particular entity relationships, can also enable the decomposition and optimization of complex queries. Furthermore, a fundamental understanding of a user's information need can be used to improve answers, refine the query, as well as suggest queries based on its similarity to other queries.

6th Biennial Conference on Innovative Data Systems Research (CIDR '13)
January 6-9, 2013, Asilomar, California, USA.

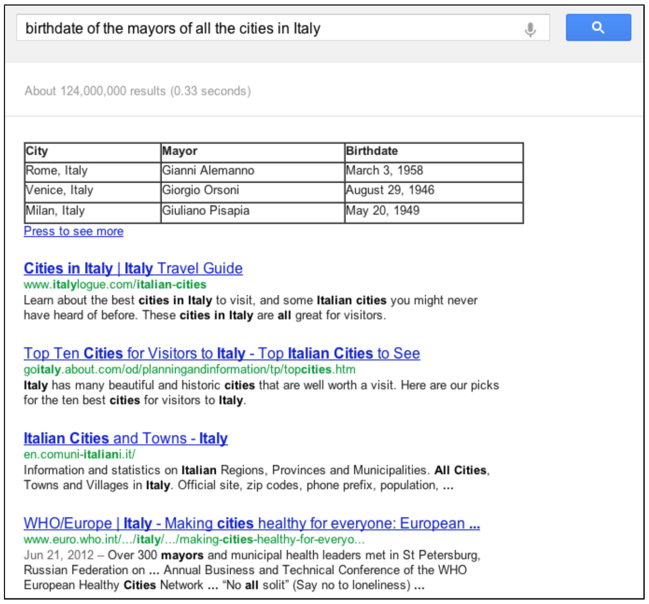
Query pattern
what is the average rainfall for ENTITY
what is the longest river in ENTITY
what is the largest city in ENTITY
who is married to ENTITY
movies being filmed in ENTITY

Table 1: Query patterns obtained by running an entity extractor over AOL query logs.

In this paper we propose a novel system called *CrowdQ* (Crowd-supported Query-answering) which uses the crowd to understand the structure of keyword queries in order to construct a repository of query templates. We can prepopulate the repository by mining for patterns in query logs and crowdsourcing their structure, as well as grow the repository over time. The template repository will aid in automatically answering future queries that are particular instances of existing templates. To answer a query instance, CrowdQ will find its match in the template repository and first attempt to provide an answer by querying a structured knowledge base (e.g., Freebase). Of course, answers not found in the knowledge base can be sent to the crowd. We describe next use cases that demonstrate the types of queries that CrowdQ will tackle through the use of templates.

Understanding query patterns is a technique already used to provide direct answers in search engine result pages for common queries (e.g., "current weather in San Francisco"). Approaches for translating natural language to structured queries exist (e.g., [7, 9]). However, existing techniques have not realized the full potential for query templates: to provide answers for all instances of a simple query pattern, including the uncommon ones, and to compose simple patterns to form complex queries. Table 1 shows examples of common patterns found when we analyzed AOL query logs using an entity extractor. Note that these examples represent both uncommon queries in addition to queries for which only some instances may be directly answered by search engines.

Having query templates for these simple queries, i.e., those that seek a single fact/attribute about a particular entity, becomes even more useful when such queries are composed to create more complex queries. Imagine a user wants to find information about the average age of politicians in Europe. She might try querying for 'birth dates of the mayors of all capital cities in Europe'. Unfortunately, this query cannot be answered as-is by a search engine. The user will have to first query for 'countries in Europe' to find a list of country names, then for 'capital city of Italy', 'capital city of Germany', etc., then for 'mayor of Rome, Italy', and, finally, for



birthdate of the mayors of all the cities in Italy

About 124,000,000 results (0.33 seconds)

City	Mayor	Birthdate
Rome, Italy	Gianni Alemanno	March 3, 1958
Venice, Italy	Giorgio Orsoni	August 29, 1946
Milan, Italy	Giuliano Pisapia	May 20, 1949

Press to see more

Cities in Italy | Italy Travel Guide
www.**italy**logue.com/**italian-cities**
Learn about the best **cities in Italy** to visit, and some **Italian cities** you might never have heard of before. These **cities in Italy** are **all** great for visitors.

Top Ten **Cities** for Visitors to **Italy** - Top **Italian Cities** to See
go**italy**.about.com/od/planningandinformation/tp/top**cities**.htm
Italy has many beautiful and historic **cities** that are well worth a visit. Here are our picks for the ten best **cities** for visitors to **Italy**.

Italian Cities and Towns - **Italy**
en.comuni-**italiani**.it/
Information and statistics on **Italian** Regions, Provinces and Municipalities. **All Cities**, Towns and Villages in **Italy**. Official site, zip codes, phone prefix, population, ...

WHO/Europe | **Italy** - Making **cities** healthy for everyone: European ...
www.euro.who.int/.../**italy**/.../making-**cities**-healthy-for-everyo...
Jun 21, 2012 – Over 300 **mayors** and municipal health leaders met in St Petersburg, Russian Federation on ... Annual Business and Technical Conference of the WHO European Healthy **Cities** Network ... "No **all** solit" (Say no to loneliness) ...

Figure 1: Search engine result page visualizing the structured results of a complex keyword query.

'birth date of Gianni Alemanno' (the mayor of Rome).

While most of these individual queries can be answered by a modern search engine with only one or even no clicks thanks to advances in web mining, it is still not possible to get a direct answer for the original information need by posing a single keyword query. The user has to issue many queries in a *cascaded* fashion where the answer of one query is used to compose the next one. By understanding the relationships between the entities in the original query, CrowdQ will be able to decompose the complex query into the structured subqueries that can be answered by matching them to existing query templates. We envision a direct answer to this user's complex information need being displayed on the search engine's result page, like the mockup in Figure 1.

The rest of this paper is organized as follows. In Section 2 we highlight the challenges that need to be addressed to answer complex keyword queries. Then in Section 3 we describe the proposed CrowdQ system architecture that will answer these queries using query decomposition and semantic search against a structured repository; we particularly focus on defining the novel steps involved in crowdsourcing query structure and semantics. We conclude in Section 4.

2. RESEARCH CHALLENGES

The goal of building query templates to obtain answers to complex information needs presents several challenges.

The first challenge is to take a keyword query about a specific entity (e.g., 'birthdate of Barack Obama') and transform it into the representation of an entire class of queries about the same entity type (e.g., 'birthdate of [person]'). This process requires query parsing and annotation; common annotation procedures include Part-of-Speech (POS) tagging and Named-Entity Recognition (NER).

The next challenge is to extract the semantics of the query by identifying the principal components that reflect the information need, i.e., the relationships between entities. We propose using paid crowdsourcing to aid in query decomposition to understand the relationship between the query's different elements.

Another challenge is identifying the appropriate answer *type*, e.g., date, name, picture, number, etc., for each complex query. This is necessary because we aim to (automatically) combine the answers to simple queries into the final answer that addresses the original complex information need by running a join query with aggregation operators over a structured data repository. We plan to leverage the crowd again to guide the choice of answer type.

Once query semantics information has been extracted, the structured query must be composed and stored so it can be reused to answer future complex queries that share similar structure without the need to further involve (and thus pay) the crowd. We will design query template matching techniques to decide whether for an given input query there exists a template that can be used to automatically decompose and answer the complex query.

In summary, the CrowdQ system needs to:

- Identify the key entities in the query (e.g., based on NLP techniques)
- Identify the relations between the key entities (e.g., by means of crowdsourcing)
- Compose the structured query and generate a query template (i.e., DB/IR)
- Store and index the generated query templates to match future queries (i.e., DB/IR)
- Integrate different structured datasets (e.g., data.nytimes.com, geonames.org) to guarantee data quality, freshness, coverage, and consistency (i.e., DB)

We describe how to address these challenges next.

3. CROWDQ

3.1 System Architecture

The proposed system architecture for CrowdQ, depicted in Figure 2, is composed of an off-line pre-processing component and an on-line query processing component. The off-line pre-processing pipeline takes as input a search engine query log and analyzes each query to populate a query template repository, with help from the crowd as described in Section 2. The on-line component takes as input a query from a web search user and decides which type of answer to return (similarly to how it is currently done for vertical search selection [2]). If the query is classified as a complex analytic search, then it is matched against the repository of structured query templates. Answers for the matching structured queries are obtained by running them against a structured fact repository and results are aggregated into the final search engine result page (SERP) (see Figure 1).

3.2 A Hybrid Pipeline for Structured Query Template Extraction

The goal of CrowdQ is to understand the structure behind complex search queries by decomposing them with the steps depicted in Figure 3. We now describe in more detail the off-line query template generation that leverages paid crowdsourcing. The steps involved in the query template generation and answering are the following.

Query Annotation and Entity Extraction.

The process begins with the input of a keyword query $q = \{k_1, ..., k_n\}$ where k_i is a keyword. The first step consists of automatically creating annotations for q as a list of

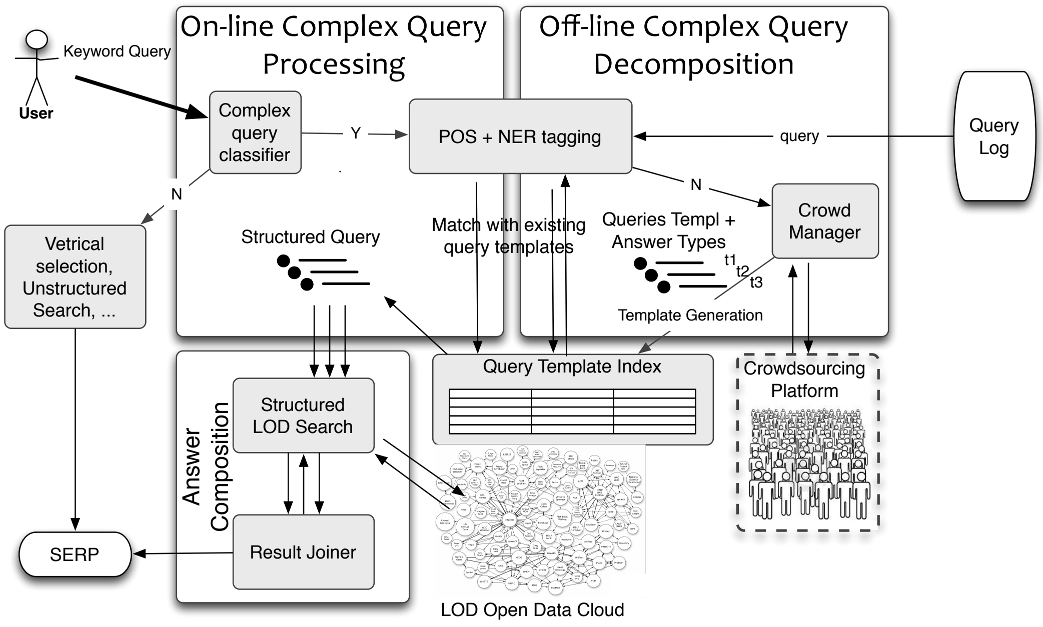
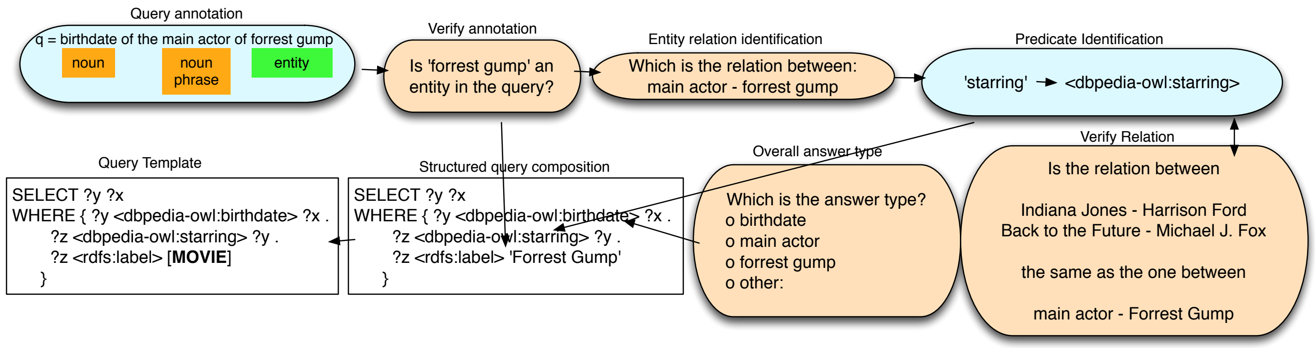


Figure 2: CrowdQ Architecture. The off-line processing starts at the top-right with a search log as input. The on-line processing starts at top-left with a user keyword query.



Query annotation
q = birthdate of the main actor of forrest gump
noun
noun phrase
entity
Verify annotation
Is 'forrest gump' an entity in the query?
Entity relation identification
Which is the relation between: main actor - forrest gump
Predicate Identification
'starring' → <dbpedia-owl:starring>
Verify Relation
Is the relation between
Indiana Jones - Harrison Ford
Back to the Future - Michael J. Fox
the same as the one between
main actor - Forrest Gump
Overall answer type
Which is the answer type?
o birthdate
o main actor
o forrest gump
o other:
Structured query composition
SELECT ?y ?x
WHERE { ?y <dbpedia-owl:birthdate> ?x .
?z <dbpedia-owl:starring> ?y .
?z <rdfs:label> 'Forrest Gump'
}
Query Template
SELECT ?y ?x
WHERE { ?y <dbpedia-owl:birthdate> ?x .
?z <dbpedia-owl:starring> ?y .
?z <rdfs:label> [MOVIE]
}

Figure 3: A hybrid pipeline for query decomposition and query template creation. Light blue steps are performed automatically. Orange steps are crowdsourced.

POS tags as well as named entity tags. This step yields two query representations $q_{POS} = \{POS_1, ..., POS_n\}$ as well as $q_E = \{E_1, ..., E_n\}$. While NLP techniques to perform such annotation on natural language exist, they often rely on the grammatical structure of text. It may be necessary to adapt them to obtain high-quality annotation of keyword queries. One possible approach to improve annotation quality is to use crowdsourcing for low confidence cases. The first task the crowd would perform is validation of the automatically generated POS and entity annotations (the *verify step*, see Figure 3). If the crowd deems the annotation incorrect, then a crowdsourced *fix step* is added. Workers will have to indicate the entities mentioned in the keyword query, which will be used in the subsequent step of the pipeline. Initial experiments we performed have shown that the crowd can effectively identify the main entities involved in complex search queries (see Table 2).

In the next step, the generated query representations are used both to create the query templates as well as to create the task that will be presented to workers on the crowdsourcing platform. Specifically, different annotations will be presented to the worker, with priority given to entity annotations which are assumed to be more significant than the POS annotations for the specific task. For example, a query with its entity and POS annotations may look like:

noun		noun phrase		entity
birth date	of the	**main actor**	of	**forrest gump**

Entity Relation Detection.

Once the query has been correctly annotated and entities have been identified, the next challenge is determining the relationships between different objects in the query. For example, the system needs to understand that 'birth date' is an attribute of 'actor' and that 'actor' is related to the 'movie' class of which 'Forrest Gump' is an instance. Automatic approaches that estimate probabilities for all possible relationships between query objects may have low efficacy due to poor language structure present in keyword queries.

Preliminary experiments have shown that crowd workers can effectively identify the key entities involved in a query but have difficulty determining entity relationships without additional support. Table 2 shows entities and their relevant attributes identified by the crowd for the mayor's birth

query: Birth date of the mayor of the capital city of Italy	
Key Entities	Main Attributes
mayor (7)	birth date (7), city (1)
city (5)	capital (2), mayor (2), title (1)
country (2)	capital (2), mayor (1), name (1)

Table 2: Entities and attributes identified by (n) out of 10 workers on Amazon Mechanical Turk for one query.

date query. While the crowd can effectively identify the key entities involved in the query, it was difficult for workers to identify the entities' key attributes and relationships (e.g., determining that mayor-city and city-country are correct relationships but mayor-country is not).

To address this challenge, we propose presenting workers with a list of questions that directly asks for the relationship between a pair of entities. All possible pairs can be presented to an individual worker because a limited number of elements is expected to be present in a keyword query. Workers can either name the relationship or state that the two elements are not related each other.

Overall Answer Type Identification.

The next step in the pipeline which can be tackled with crowdsourcing is the identification of the answer type. The answer type could be either mentioned in the query (and thus workers will simply have to select the relevant keyword, e.g., 'birth date') or the worker will have to determine the implicit answer type.

Predicate Selection.

For each identified relationship, a Resource Description Framework (RDF) predicate needs to be selected. This selection can be accomplished, for example, based on text similarity between the relationship and candidate predicates in the RDF store. However, automatic systems can fail on this task because the RDF store may contain synonyms as well as cryptic predicate names. To improve accuracy, an additional *verify step* can be done for RDF predicate selection (see Figure 3): for each selected RDF predicate, the crowd is shown example relationships from the knowledge repository and asked if the relationship is the same as that between the entities in q.

For example, for the crowd-identified relation 'starring' between 'Forrest Gump' and 'main actor' the predicate `<dbp-prop:starring>` is selected. The examples ('Indiana Jones' - 'Harrison Ford') and ('Back to the Future' - 'Michael J. Fox') are shown to the crowd to verify that the relation is the same and thus that the predicate is correct.

Structured Query Composition.

At this point in the pipeline, a structured SPARQL query is composed:

```
SELECT ?y ?x
WHERE { ?y <dbpedia-owl:birthdate> ?x .
        ?z <dbpedia-owl:starring> ?y .
        ?z <rdfs:label> 'Forrest Gump'
      }
```

This query run against the BTC09[1] dataset returns the following answers:

```
<http://dbpedia.org/resource/Robin_Wright_Penn> 1966-04-08
<http://dbpedia.org/resource/Tom_Hanks> 1956-07-09
<http://dbpedia.org/resource/Sally_Field> 1946-11-06
<http://dbpedia.org/resource/Gary_Sinise> 1955-03-17
<http://dbpedia.org/resource/Mykelti_Williamson> 1960-03-04
```

The final phase in creating the query template consists of replacing specific entity instances in the structured query with wildcards. For example, instead of 'Forrest Gump' we will consider a generic entity type [movie]. Thus, the generated template can match any raw keyword query of the same POS/NER annotation structure as q.

Note that some of the crowdsourcing tasks, e.g., the selection of the correct predicate for entity relationships, could be *gamified* (presented as a challenging game) and thus completed by the crowd at no cost rather than being done by paid crowdsourcing.

4. CONCLUSION

In this paper we introduced CrowdQ, a novel hybrid human-machine system for answering complex keyword queries by leveraging the crowd to understand query semantics. By building a repository of structured query templates with the help of both algorithms and people, answers to new queries can be automatically provided by matching them to templates and taking advantage of structured knowledge bases. We described the challenges in realizing this vision, as well as our initial results and future plans for addressing them. The proposed system architecture combines techniques from areas such as DB, NLP, IR, Data Mining, and human intelligence to understand the need behind a user keyword query.

5. ACKNOWLEDGEMENTS

This research is supported in part by the Swiss NSF grant IZK0Z2_142896, by NSF CISE Expeditions award CCF-1139158, gifts from Amazon Web Services, Google, SAP, Blue Goji, Cisco, Cloudera, Ericsson, General Electric, Hewlett Packard, Huawei, Intel, Microsoft, NetApp, Oracle, Quanta, Splunk, VMware and by DARPA (contract #FA8650-11-C-7136).

6. REFERENCES

[1] O. Alonso and M. Lease. Crowdsourcing for information retrieval: principles, methods, and applications. In *SIGIR*, 2011.

[2] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of evidence for vertical selection. In *SIGIR*, 2009.

[3] M. S. Bernstein et al. Direct answers for search queries in the long tail. In *CHI*, 2012.

[4] M. J. Cafarella et al. Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1), 2008.

[5] M. J. Franklin et al. CrowdDB: Answering Queries with Crowdsourcing. In *SIGMOD*, 2011.

[6] A. Marcus, E. Wu, S. Madden, and R. Miller. Crowdsourced Databases: Query Processing with People. In *CIDR*, 2011.

[7] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over RDF data. In *WWW*, 2012.

[8] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR*, pages 159–166, 2007.

[9] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural language questions for the web of data. In *EMNLP-CoNLL*, 2012.

[1]The dataset consists of more than one billion RDF triples and is available at: `http://vmlion25.deri.ie/`