SSQ: Querying Entities Using Shallow Semantics

Xiaonan Li
Department of Computer Science and Engineering
University of Texas at Arlington
xiaonan.li@mavs.uta.edu

Chengkai Li
Department of Computer Science and Engineering
University of Texas at Arlington
cli@uta.edu

Cong Yu
Yahoo! Research New York
congyu@yahoo-inc.com

ABSTRACT

In [3], we proposed Shallow Semantic Query (SSQ) for structured querying of named entities from Web text. Complex semantic constraints on entities can be neatly expressed in SSQ as a junction of predicates centered on entity variables. Such complex entity queries are not supported by current entity search systems. Our position-based ranking method achieved better precision on SSQ query results than state-of-the-art models adapted for SSQ. We also developed an Entity-Centric Index (ECI) and the corresponding Entity-Centric Retrieval (ECR) algorithm, which enable more efficient processing of SSQ queries than the document-centric approach extended from existing entity search systems. In this paper, we give an overview of the semantics of SSQ queries, the ranking framework, and the new index and query processing method. We present the system architecture and the implementation of the major components. We also elaborate on a system demonstration scenario. Our online demo is available at `http://idir.uta.edu/ssq`.

1. INTRODUCTION

Information discovery on the Web has so far been dominated by the keyword-based and document-centric paradigm. Although commercial Web search engines have been very successful in helping people find useful Web pages, there is still a major gap between user information needs and search results. Users have to digest the contents of returned Web pages to obtain answers to questions in their minds.

Named entity search, e.g., finding all Silicon Valley companies, is one of the fastest increasing user information needs. In response to this user demand, entity search was proposed recently to fill the gap between user information needs and search results. Specifically, an entity search system returns named entities, instead of Web pages, to Web users directly, saving users from having to digest page contents. However, existing entity search systems [1, 2, 4] can only express and handle simple queries that have only one semantic constraint. This limitation significantly undermines their advantage over traditional search engines. For example, to find all Silicon Valley companies, a keyword query on "silicon valley company" using a Web search engine may return a page containing a list of such companies, thus making the need for entity search less urgent. Meanwhile, a huge body of real world scenarios, in which users have multiple semantic constraints inter-weaved on desired entities, have so far been left out of the consideration and capability of both existing entity search systems and Web search engines.

Example 1 (Motivating Example): Consider a business analyst investigating the development of Silicon Valley. Particularly, she is interested in: *Finding companies and their founders, where the companies are in Silicon Valley and the founders are Stanford graduates.*

With traditional search engines, our analyst may start with a search on "Silicon Valley company" and scan through the potentially long list of result articles to, hopefully, fetch a list of companies that are likely to be in Silicon Valley. She then similarly issues another search on "Stanford graduate" to find a list of people graduated from Stanford University. She then manually combines entities in these two lists and, by multiple additional searches, check if a company was founded by a person, for each pair of person and company. Alternatively, she can also go through the list of companies and, for each company, find its founders and check if Stanford is their alma mater by multiple search queries. Both are painful options and require the user to break down the task into a time-consuming, error-prone iterative procedure of searching, reading and re-searching. Using current entity search systems, the analyst also has to issue multiple queries and assemble the query results (several lists of entities) manually to reach final answers.

Inspired by the growing demands for searching entities with complex constraints (such as Example 1), we developed an entity-centric structured query mechanism, Shallow Semantic Query (SSQ) [3], to facilitate structured entity queries over Web text. Developing a quality SSQ system is not a trivial undertaking. *First*, as a new query mechanism, the semantics of SSQ has to be formalized. *Second*, we must tackle the challenge of ranking SSQ query results. Given that SSQ is essentially a search system over text, a vast amount of false query answers are unavoidable. Moreover, in SSQ queries the users often expect a large number of true answers, therefore achieving good precision-at-k for small k is not sufficient. We proposed a ranking method that exploits position-based features to improve ranking precision in the long range. *Third*, like any online search systems, query efficiency has a key impact on user experiences with SSQ. To support efficient processing of SSQ queries, we designed an Entity-Centric Index (ECI) and the corresponding Entity-Centric Retrieval (ECR) algorithm. Comprehensive experiments verified that both our ranking method and query processing technique are effective, out-performing state-of-the-art techniques adapted for SSQ [3]. Our online demo is available at `http://idir.uta.edu/ssq`.

Chakrabarti et al. [1] studied optimization of proximity scoring function for ranking entities. EntityRank [2] presented Impression Model for the same purpose. However, their ranking models do not aim at good ranking precision in the long range as our method does. Recently, the Content Query Language [4] was proposed for entity search, with many language features incorporated. However, it was still intended to handle simple entity search queries (one semantic constraint per query). All these existing systems adopted document-centric approach for indexing and query processing, which is less efficient than the entity-centric approach according to our study [3].

2. SHALLOW SEMANTIC QUERY

In this section, we give a brief overview of the semantics of SSQ queries, our ranking framework, and the entity-centric index and query processing methods. The interested reader is referred to [3] for details.

2.1 Overview

Example 2 illustrates the SSQ query for Example 1. The query syntax is modeled after SQL. There are three elementary concepts within this SSQ query. First, the query centers on two *entity variables*, x and y. Variable x is bound to all entities belonging to type PERSON and y to all entities belonging to type COMPANY. Second, for each variable, the query specifies a *selection predicate* as the criterion on the selection of entities. For example, a desired PERSON x should be a Stanford graduate (x:["Stanford" "graduate"]). Third, a *relation predicate* specifies the relation between x and y (y was founded by x).

Example 2 (SSQ Query):

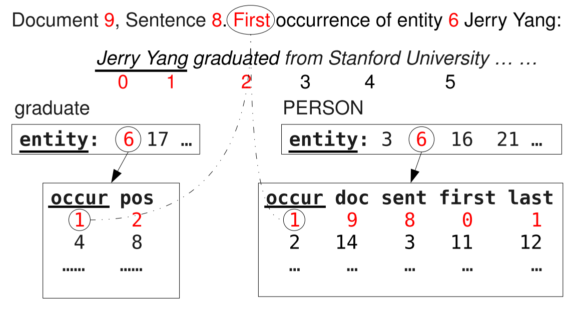
```
SELECT x, y
FROM   PERSON x, COMPANY y
WHERE  x:["Stanford" "graduate"]    // Predicate P1
  AND  y:["Silicon Valley"]         // Predicate P2
  AND  x,y:["found"]                // Predicate P3
```

More predicates can be added to the above query, e.g., (x:["Russian"]). Generally, an SSQ query may have an arbitrary number of variables and predicates. Note that the semantics of the query is indicated by keyword phrases associated with entity variables. There is no explicit schema behind SSQ queries.

A predicate in an SSQ query requires some typed entities to co-occur with a set of keyword phrases [1] in the same context. In this paper, we use sentences as the default contexts. In Example 2, predicate P1 requires a PERSON entity to co-occur with "Stanford" and "graduate" in one sentence. Such sentences are considered *evidences* for this predicate. Suppose the following three evidences are found:

S1: *Stanford University graduate Jerry Yang ...*
S2: *...a senior manager at Yahoo! in Silicon Valley.*
S3: *Jerry Yang co-founded Yahoo!.*

[1]A single keyword is considered a phrase of length 1.


Document 9, Sentence 8. First occurrence of entity 6 Jerry Yang:
Jerry Yang graduated from Stanford University … …
0 1 2 3 4 5
graduate
entity: 6 17 …
PERSON
entity: 3 6 16 21 …
occur pos
1 2
4 8
occur doc sent first last
1 9 8 0 1
2 14 3 11 12

(a) Term-Entity posting list **(b) Entity-Doc posting list**

Figure 1: Entity-Centric Index

Jerry Yang satisfies P1 by evidence S1, Yahoo! satisfies P2 by evidence S2, and they together satisfy P3 by S3. Assembling the information together, SSQ composes an answer ⟨Jerry Yang, Yahoo!⟩ as it satisfies all the query predicates.

2.2 Ranking

Given an SSQ query answer, in our ranking framework, a score for each predicate in the query is evaluated, based on the supporting evidences. The final score of the answer is computed from the predicate scores. The scores for different predicates are independent from each other. The intuition can be explained as follows. In Example 2, whether a PERSON is a Stanford graduate (P1) is independent from whether she founded any COMPANY (P3) and certainly irrelevant to whether a COMPANY is in Silicon Valley (P2).

Following predicate independence, our ranking framework uses a Bounded Cumulative Model (BCM) that integrates three position-based features in scoring individual predicates and computes the final score by the product of the predicate scores. Given a query with a set of predicates P, the BCM scores an answer t for predicate $p \in P$ as

$$p(t) = \sum_{o \in O}(f(o,p)[1 - \prod_{s \in \phi(t,p,o)} (1 - prox(t,p,s)credit(o,p,s))])$$

where O is the set of all possible ordering patterns of the phrases and variables in p; $\phi(t,p,o)$ is the set of t's evidences for p, in which the entities in t and the phrases of p follow the ordering pattern o. The three features in BCM are: $f(o,p)$, the frequency that pattern o occurs among the evidences for p; $prox(t,p,s)$, the proximity of entities in t and the phrases of p, in evidence s; $credit(o,p,s)$, the measure of how likely an evidence s following pattern o is a valid evidence.

2.3 Index and Query Processing

We now introduce our Entity-Centric Index (ECI) and Entity-Centric Retrieval (ECR) algorithm that are designed to support efficient processing of SSQ queries.

ECI uses posting lists as the physical data structure for organizing the $\{term, document, entity\}$ three-dimension information space. It contains two kinds posting lists, term-entity posting lists (TEPL) and entity-document posting lists (EDPL). A TEPL is built for each unique term w, enlisting all entities co-occurring with w by their IDs. Each such entity is associated with a list of entries recording co-occurrence information with two attributes, *occur* (entity occurrence identifier) and *pos* (w's position). In Figure 1(a), "graduate" co-occurs with entity 6, 17, etc. It occurs at position 2 of the sentence where entity 6, Jerry Yang, appears for the first time ($occur$=1). An EDPL is built for each type d, enlisting all entities of type d by their IDs. Each entity in the EDPL is associated with a complete list of occurrence information. In Figure 1(b), entity 3, 6 and 16 all belong to PERSON. The first occurrence of entity 6 is found in document 9 sentence 8, spanning from position 0 to 1.

Intuitively, a three-way merge join of two TEPLs (for "Stanford" and "graduate", respectively) and one EDPL (PERSON), by joining on $\langle entity, occur \rangle$, can retrieve all the evidences for predicate P1 in Example 2. However, it is inefficient to perform a multi-way merge join for every predicate in a multi-predicate SSQ query and the performance degrades quickly as more predicates are added. The design of our ECR algorithm leverages the fact that all the terms associated with the same variable, no matter in which predicate, are required to co-occur with the same entity. By predicate P1 and P3, three terms ("Stanford", "graduate" and "found") are associated with variable x. Therefore, if a PERSON is not enlisted in the TEPL of some of the terms, no evidence of the entity can be found for the corresponding predicate. For example, if Steve Jobs is not in the TEPL of "Stanford", there is certainly no sentence in which he co-occurs with "Stanford", thus P1 cannot be satisfied by Steve Jobs. There is no need to retrieve the co-occurrence evidence of Steve Jobs and "graduate". Furthermore, it is also unnecessary to retrieve the evidences containing Steve Jobs and some COMPANY for predicate P3. Therefore, some unnecessary processing can be avoided, making ECR more efficient than the straightforward approach of doing multi-way join for every predicate. ECR is shown to require an order of magnitude less disk I/Os than the document-centric approach adapted from the techniques in existing systems.

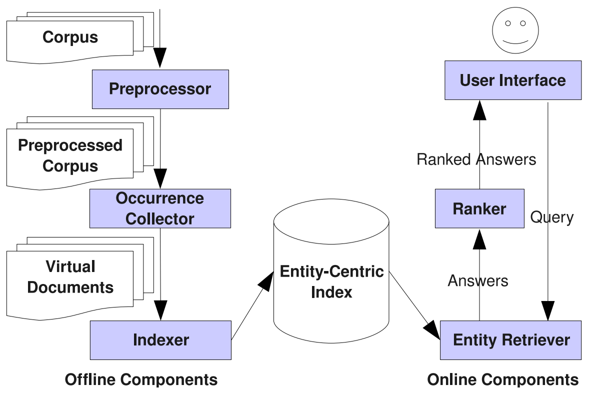


Figure 2: System Architecture

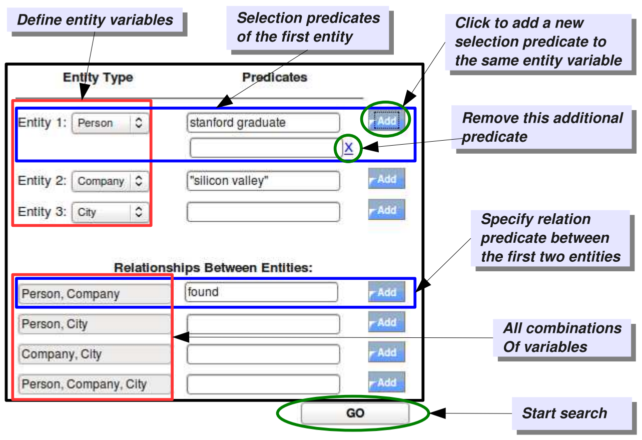

Define entity variables
Selection predicates of the first entity
Click to add a new selection predicate to the same entity variable
Remove this additional predicate
Specify relation predicate between the first two entities
All combinations Of variables
Start search
Entity Type
Predicates
Entity 1: Person
stanford graduate
Add
X
Entity 2: Company
"silicon valley"
Add
Entity 3: City
Add
Relationships Between Entities:
Person, Company
found
Add
Person, City
Add
Company, City
Add
Person, Company, City
Add
GO

Figure 3: Query Interface

3. SYSTEM ARCHITECTURE

Our SSQ system is implemented in Apache Lucene [2]. We use the 2008-07-24 snapshot of Wikipedia [3] as both the corpus for finding evidences and entity repository. As shown in Figure 2, the system consists of the offline components (*Preprocessor*, *Occurrence Collector*, and *Indexer*) and the online components (*Entity Retriever*, *Ranker*, and *User Interface*).

3.1 Offline Processing

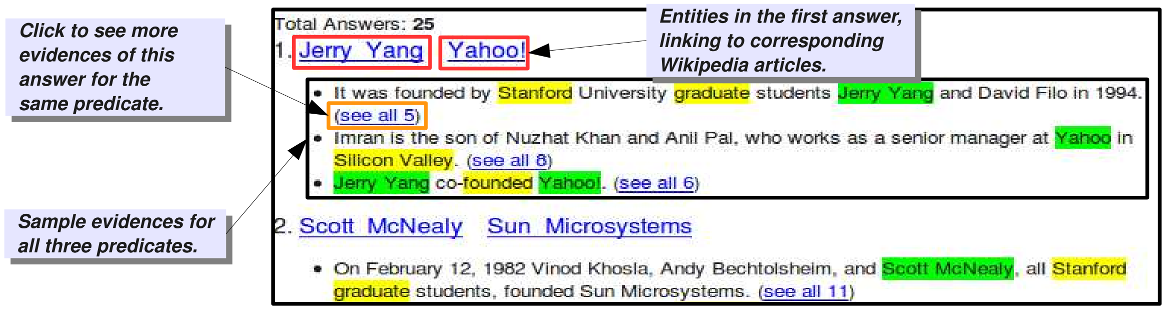
The **Preprocessor** prepares Wikipedia articles for further processing. For each article, it removes section titles, tables, infoboxes, references, etc., retaining only the main textual content. Then it uses a sentence detector to segment the text into sentences. Finally, punctuation marks are removed and words are stemmed. The original Wikiepdia dump is about 18GB. The size is reduced to about 6GB after preprocessing. In our system, each Wikipedia article is treated as the entry of a named enitity. In the content of a Wikipedia article, the hyperlinks to other Wikipedia articles are considered occurrences of the corresponding entities. The **Occurrence Collector** builds, for each entity, a virtual document that contains all the sentences containing its occurrences. A sentence containing multiple entities is thus included in the virtual documents of all these entities. The **Indexer** constructs an Entity-Centric Index from the virtual documents of all the entities in a way similar to building inverted index over normal documents.

Our corpus contains around 2.4 million Wikipedia articles. We defined 10 entity types and assigned about 0.75 million entities (articles) into these types based on their categories in Wikipedia. For example, if an entity has a category named *English novel*, it is assigned to the type NOVEL. This simple approach turns out accurate enough to demonstrate the effectiveness of SSQ. For all the typed entities, a total of about 100 million occurrences are collected. Entities not belonging to these types are assigned to a special type called UNTYPED. All the entities are indexed in our system, resulting in an index of 6.4GB, comparable to the size of the preprocessed corpus.

3.2 Online Processing

The **Entity Retriever** implements our Entity-Centric Retrieval algorithm. Given an SSQ query, it first retrieves the evidences for all predicates. Then, it composes query answers by examing whether a tupe of entities has evidences for all the predicates. Evidences supporting entities that appear in query answers are retained. The result of Entity Retriever is sent to **Ranker** to compute the ranking scores of all the answers, based on our Bounded Cumulative Model. The **User Interface** assists users in composing SSQ queries

[2]http://lucene.apache.org
[3]http://download.wikimedia.org


Click to see more evidences of this answer for the same predicate.
Entities in the first answer, linking to corresponding Wikipedia articles.
Sample evidences for all three predicates.

Total Answers: 25

1. Jerry Yang Yahoo!

- It was founded by Stanford University graduate students Jerry Yang and David Filo in 1994. (see all 5)
- Imran is the son of Nuzhat Khan and Anil Pal, who works as a senior manager at Yahoo in Silicon Valley. (see all 8)
- Jerry Yang co-founded Yahoo!. (see all 6)

2. Scott McNealy Sun Microsystems

- On February 12, 1982 Vinod Khosla, Andy Bechtolsheim, and Scott McNealy, all Stanford graduate students, founded Sun Microsystems. (see all 11)

Figure 4: Result Presentation

and displays ranked answers in HTML pages. Figure 3 is a snapshot of the query interface. It is divided into two parts. The upper part specifies entity variables and their selection predicates. The bottom part specifies relation predicates between the variables. Figure 4 shows a snapshot of the result HTML page. It consists a ranked list of answers returned by Ranker.

4. DEMONSTRATION

This section depicts our demonstration system in a concrete scenario. We show how to issue the query in Example 2 using the query interface and interpret the query result. For an online demo, please see `http://idir.uta.edu/ssq`.

4.1 Query Formulation

With the query interface in Figure 3, we are going to define two entity variables, one as PERSON and the other as COMPANY, and specify three predicates.

1. The first variable x is defined by selecting its desired type, PERSON, using the first combo box. Variable x has one selection predicate, P1. It is specified by entering two keywords, "Stanford" and "graduate", in the input box next to the selected type.

2. The second variable y and its selection predicate P2 are specified similarly using the second combo box and the corresponding input box.

3. More selection predicates can be added to the same variable by clicking the **Add** button beside each predicate. In this case, another blank input box is appended below the existing predicate. Keywords or phrases can thus be entered. Additional predicates may be removed later by clicking the **X** mark beside it.

4. To explore a bit more of the interface, we temporarily define a third variable CITY z. We shall now observe four lines in the bottom part of the interface. Each line corresponds to one combination of the three variables. The combinations are xy, xz, yz, xyz. For example, the first line is for combination xy (PERSON and COMPANY).

5. The input box next to each combination is for specifying a relation predicate for entities in that combination. According to P3 of Example 2, "found" in entered in the input box next to the first combination xy. More relation predicates can be added to xy by clicking the corresponding **Add** button.

6. To correctly specify our example query, the third variable z shall be removed by de-selecting its type CITY, upon which action, all combinations involving z in the bottom part will disappear.

7. By clicking the **GO** button at the bottom of the interface, the query is sent to SSQ server for processing.

4.2 Result Interpretation

Upon successful query processing, the result is returned as an HTML page to the user. Figure 4 shows a snapshot of the result page of Example 2, and we give a detailed interpretation of this result.

The first line of the result page indicates that there are 25 answers in total. However, due to space limitations, only the top-2 answers are shown in the snapshot. The best scored answer is ⟨Jerry Yang, Yahoo!⟩. Jerry Yang is instantiated from the first variable (x) in Figure 3 and Yahoo! the second variable (y). Each entity appears as a hyperlink to the corresponding Wikipedia article. For instance, Yahoo! is linked to a Wikipedia article with the same title.

A snippet of the first answer (the inner black box) is provided for the user to get a quick sense of the answer's validity, without reading the Wikipedia articles. For this query, a snippet consists of three sample evidences retrieved by Entity Retriever, each for a different predicate in the query. Entities and keyword phrases are highlighted for clarity. Since there can be more evidences for the same predicate, a link to all evidences of the same answer with regard to the same predicate is provided at the end of each sample evidence.

5. REFERENCES

[1] S. Chakrabarti, K. Puniyani, and S. Das. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *WWW*, 2006.

[2] T. Cheng, X. Yan, and K. C.-C. Chang. EntityRank: searching entities directly and holistically. In *VLDB*, 2007.

[3] X. Li, C. LI, and C. Yu. Structured querying of annotation-rich web text with shallow semantics. Technical report, Department of Computer Science and Engineering, University of Texas at Arlington. `http://ranger.uta.edu/%7Ecli/ssqreport.pdf`.

[4] M. Zhou, T. Cheng, and K. C.-C. Chang. Data-oriented content query system: searching for data into text on the web. In *WSDM*, 2010.