A Study of Semantic Search in SemSearch 2011

Xitong Liu
Department of Electrical and Computer Engineering
University of Delaware
Newark, DE 19716, USA
xliu@ece.udel.edu

Cong-Lei Yao
HP Labs China
HP Labs
Beijing, China
conglei.yao@hp.com

Hui Fang
Department of Electrical and Computer Engineering
University of Delaware
Newark, DE 19716, USA
hfang@ece.udel.edu

ABSTRACT

We describe the joint efforts of University of Delaware and HP Labs China in the participation of both the Entity Search Track and List Search Track in SemSearch Challenge 2011. In the Entity Search track we focus on how to apply existing Information Retrieval techniques to retrieve the related entities from semi-structured RDF corpus and how to improve the performance using some heuristics. In the List Search track we studied how to apply Natural Language Processing techniques to analyze the query and extract the type requirement which can be used to filter out the irrelevant entities from initial retrieval.

1. INTRODUCTION

We participated both tracks in SemSearch Challenge 2011. The goal of the first track, Entity Search Track, is to retrieve a list of relevant entities given by keyword query. It can be classified into the category of entity search, on which more and more research efforts have been put in community recently, including the *expert finding* in Enterprise Track [4, 8, 1] and *related entity finding* (REF) in the Entity Track [2, 3]. One main difference between them is that for both expert finding and related entity finding entities are retrieved from unstructured data set while for Entity Search Track the entities are retrieved from semi-structured data set.

The goal of second track, List Search Track, is to find resources which belong to a particular set of entities. This is very similar to the Entity List Completion (ELC) pilot task in Entity Track 2010 [3]. One major difference is that for ELC task besides entity description, some example entities are already given in the query and the goal is to *complete* the set of entities by retrieving other related entities in the same set while for the List Search Track only the entity description is given, which makes the task more challenging.

Obviously we can formulate the problems of both track as the problem of keyword search over semi-structured data set. To solve this problem, we plan to leverage the existing Information Retrieval (IR) techniques which have been shown to be effective on keyword search over unstructured data set. Hence there lies two challenges before the problem solution: (1) how to apply the IR techniques over the semantic semi-structured date set, (2) how to model the relevance between the keyword query and entities. To address the first challenge, we process the semi-structured date set into unstructured document collection on which existing IR techniques can be applied. For the second challenge, we utilize some semantic features of the entities in the data set to connect the query and entity.

For the Entity Search Track, we first build the unstructured entity profile document collection based on the semi-structured data set and apply axiomatic retrieval functions to search over the profile document collection using the given keyword query. The entities are ranked based on the rank of their profile documents. For the List Search Track, we first apply some Natural Language Processing techniques to extract both the content requirement and type requirement. Content requirement is used to retrieve an initial list of entity candidates, and type requirement is used to filter out irrelevant entities then.

The paper is organized as follows. We describe the processing of data set in Section 2, introduce the method of Entity Search Track in Section 3 and the method of List Search Track in Section 4, and conclude in Section 5.

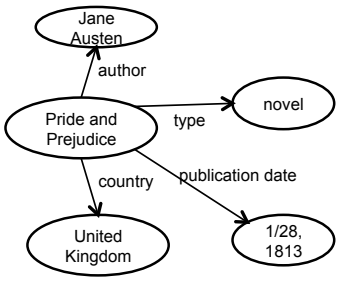
2. DATA SET PROCESSING

The standard data collection for SemSearch Challenge 2011 is Billion Triple Challenge 2009 dataset [1], which consists of a set of RDF graph collection represented in N-Quads format [2]:

```
<subject> <predicate> <object> <context> .
```

Basically each entity is represented by one node in the RDF graph, and many attributes are connected with the entity, as shown in Figure 1(a). Each attribute is a pair of attribute name and value. In this example, we have one entity "Pride and Prejudice" with four attributes. To represent the RDF graph in plain text, N-Quads store each attribute pair in one record. More specifically, the entity name is taken as the

[1]http://vmlion25.deri.ie/
[2]http://sw.deri.org/2008/07/n-quads/



(a) RDF Graph of Pride and Prejudice

<Pride and Prejudice> <author> <Jane Austen> <LINK> .
<Pride and Prejudice> <country> <United Kingdom> <LINK> .
<Pride and Prejudice> <type> <novel> <LINK> .
<Pride and Prejudice> <publication date> <1/28, 1813> <LINK> .

(b) N-Quads of Pride and Prejudice

Figure 1: Example of RDF Graph and N-Quads records

subject field, attribute name as the predicate field, attribute value as the object field. Moreover, the fourth field, i.e. the context field, is the URI of the RDF graph that provides the context information of entity. Therefore, each entity is represented by a couple of N-Quads records, all of which share the same subject field, as shown in Figure 1(b).

In order to represent the entity in unstructured document, we follow our method in the participation of Semantic Search 2010 Workshop [6] by merging all the attributes of an entity (both the attribute name and value) together as "bag-of-words". The reason is that each attribute is kind of description of the entity covering certain aspect, merging all the attributes together would build a *profile* document, which can cover many aspects for the entity. The profile document construction can be done by simply merging all the predicate fields and object fields of N-Quads records which share the same subject field. The context fields are abandoned because they can not provide any useful information of the entity besides the origins of the RDF graphs.

We notice that there are two different formats for the predicate fields:

1. `<http://dbpedia.org/property/abstract>`, in which the attribute type is located after the last / of URI, i.e. `abstract`.
2. `<http://www.aktors.org/ontology/portal#year-of>`, in which the type is located in the string after `#`, i.e. `year-of`.

We only extract the last part in the predicate fields and discard the remaining parts because we think the last part already provides enough semantic information about the attribute name. We use Indri [3] to build index over the unstructured profile document collection.

3. ENTITY SEARCH TRACK

3.1 Problem Formulation

The task of Entity Search Track is to rank the entity according to the relevance to a given keyword query. Since we have already processes the data set into unstructured document collection, we can use the relevance between the entity profile document and the keyword query to model the relevance between the entity and keyword query, the problem of entity search over the semi-structured data set is essentially equivalent as document retrieval over the unstructured document collection.

3.2 Submitted Runs

For this track, we submitted three runs:

UDelAX

The retrieval function we use is the Axiomatic retrieval function [5] because the results of our system in Semantic Search 2010 Workshop show that it outperforms other two retrieval functions [6]. The scoring function $S(Q, D)$ is shown as below:

$$S(Q,D) = \sum_{t \in Q \cap D} c(t,Q) \times \frac{c(t,D)}{c(t,D) + s + \frac{c(t,D)\cdot|D|}{avdl}} \times \left(\frac{N+1}{df(t)}\right)^{0.35}, \quad (1)$$

where $c(t, D)$ is the count of term t in document D, $c(t, Q)$ is the count in query Q, N is the number of documents in the collection, $|D|$ is the document length, $avdl$ is the average document length of the collection, $df(t)$ is the number of documents containing term t and s is a parameter which is set to 0.05.

UDelProx

Based on the Axiomatic retrieval function, we also applied query proximity to favor the document which query terms occur more closer than others because we think that documents with higher query term proximity indicate higher relevance. We leverage the facility provided by the Indri Query Language [4] to apply proximity limitation over the query terms in the following format: `#odN( QUERY )` which means terms must appear in an ordered window and there must be at most $N - 1$ terms between any of two adjacent query terms. N is set to 15 as the results over the query set of Semantic Search 2010 Workshop show that it can lead to optimal performance.

UDelVO

Based on the study of query set, we found that they are

[3]http://www.lemurproject.org/indri/

[4]http://www.lemurproject.org/lemur/IndriQueryLanguage.php

Run	MAP	Rprec	P@10
UDelAX	0.1996	0.1928	0.2180
UDelProx	**0.2167**	**0.2391**	**0.2600**
UDelVO	0.1858	0.1885	0.1940

Table 1: Performance of three submitted runs in Entity Search Track

essentially the entity name in free text. Although most the entity are represented as URIs, the entity name are sometimes embedded in the URI. For example, the entity "IBM" may be represented as `http://dbpedia.org/page/IBM`, and it may be a relevant result for query "IBM". Therefore, we assume that if the URI have any vocabulary overlap (VO) with the keyword query, it's an strong evidence that it's more likely relevant to the query. Thus based on the initial retrieval results by Axiomatic retrieval function, we promote all the entities with vocabulary overlap with the query by ranking them higher than others.

3.3 Performance Evaluation

We evaluate the performance of the three submitted runs using the judgement file released by the organizers. We report three measurements: MAP (Mean Average Precision), Rprec (Precision at R where R is the number of relevant documents for a given query) and P@10 (Precision at rank 10), as shown in Table 1. According to the results, **UDelProx** performs best among all three runs, which shows that incorporating term proximity can improve the performance. Moreover, **UDelVO** performs worse than the other two runs, which shows the assumption that relevant entities have vocabulary overlap with the keyword query does not hold for all relevant entities.

4. LIST SEARCH TRACK

4.1 Problem Formulation

The task of List Search Track is to find entities which belong to a particular set of entities. Actually the query is the description of entity set. Some sample queries are:

```
ten ancient Greek city
astronauts who walked on the Moon
```

The difference between the two tracks are that the Entity Search Track is to find the entities which match the query (for example, entities relevant to the entity name: MIT) and the List Search Track is to to find the set (or a collection) of entities which match the criterion of the query (for example, all the ten entities which are ancient Greek cities). The key challenge for List Search Track is how to utilize the semantic meta data to find the entities which *match the criterion*.

Based on the study of query set, we find that almost all the queries contain two different parts: *type requirement* Q_t which specifies the type of the entity and *content requirement* Q_c which describes some other common attributes of the entities in the set. For example, consider the query: "astronauts who walked on the Moon", the "astronauts" is the type requirement which says that the relevant entities should be "astronauts", rather than "doctors", "politicians" or something else, while the "who walked on the Moon" asks for the set of astronauts which share the same attribute: they have walked on the Moon. To tackle the problem of this track, we first need to identify both the type requirement and content requirement.

We found that all the 50 queries are expressed in natural language rather than keywords as in Entity Search Task. Therefore, some Natural Language Processing (NLP) techniques can be applied to analyze the query. To address this challenge, we apply the Illinois Chunker [7] over the queries and find that it can successfully separate the type requirement from content requirement in most cases by chunking the whole sentence into several parts and the type requirement is in one chunk denoted by part of speech tag NP which stands for noun phase. For example, the query "astronauts who walked on the Moon" is chunked as below:

```
[NP Astronauts ] [NP who ] [VP landed ] [PP on ] [NP the Moon ] .
```

Based on the analysis, we find that the type requirement for almost all the queries are in the first NP chunk. Therefore, we extract the first NP chunk as the type requirement of the query and the remaining parts as the content requirement.

Intuitively, a relevant entity should meet both the content requirement and type requirement. Since the content requirement represents some attributes of the entity, we can use the content requirement as query to retrieve a list of entities by applying some IR techniques such as the axiomatic function mentioned in Section 3. After that, we can extract the entities which match the type requirement from the initial retrieved entity list.

By manually selecting some relevant entities and analyzing the structure of RDF graph, we find that some attributes specify the type information of the entity, and we use them to identify whether the entity meet the type requirement or not. We manually choose three common attributes which specify the entity type: "type", "description" and "sameas" and check whether the type requirement is mentioned in any of the attributes. If so, we label it as relevant, otherwise irrelevant. We keep the relevant entities and remove the irrelevant ones. The rank of the entities is kept unchanged.

4.2 Submitted Runs

For this track, we also submitted three runs:

UDelRun1

We use the Axiomatic retrieval function [5] for initial retrieval and use the type requirement automatically extracted by the Illinois Chunker to filter out irrelevant entities.

UDelRun2

Although the type requirement can be automatically extracted by the chunker, sometimes there are some *vocabulary gap* between the type requirement in the query and type attributes in the semi-structured data set. For example, one query is "members of u2" and the automatic extracted type

Run	MAP	Rprec	P@10
UDelRun1	0.0999	0.1283	0.1740
UDelRun2	**0.1285**	**0.1497**	**0.1800**
UDelRun3	0.1079	0.1446	**0.1800**

Table 2: Performance of three submitted runs in List Search Track

requirement is "member". Actually, the type attributes of the relevant entity may be "person" rather than "member". Therefore, our method will fail for such queries. To solve this problem, we manually expand the type requirement by adding some conceptually relevant terms. For the previous example, we will add "person" to the type requirement.

UDelRun3

We applied the model-based relevance feedback [9] to expand the original content requirement query and retrieve a list of entities. Manual expanded type requirement is applied to filter out irrelevant entities then. The only difference between **UDelRun2** and **UDelRun3** is the initial retrieval function.

4.3 Performance Evaluation

The performance of the three submitted runs for the list search track are evaluated in three measurements: MAP, Rprec and P@10, as shown in Table 2. **UDelRun2** performs better than **UDelRun1** as expected, since the manual expanded type requirement can cover more aspects than the automatic extracted one. What's more, the model-based feedback does not show any advantage than axiomatic retrieval function in this case, as **UDelRun2** also performs better than **UDelRun3**.

5. CONCLUSIONS

In this paper we describe our methods for the participation of SemSearch Challenge 2011. The main focus is to study how to apply IR techniques to solve the entity search problem over the semi-structured data set. In the List Search Track, some NLP techniques are also applied to better understand the different query requirement.

In our future work, we plan to study how to automatically expand the type requirement based on some conception hierarchy.

6. REFERENCES

[1] P. Bailey, N. Craswell, A. P. de Vries, and I. Soborof. Overview of the TREC 2007 Enterprise Track. In *Proceedings of Text Retrieval Conference*, 2007.

[2] K. Balog, A. P. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld. Overview of the TREC 2009 Entity Track. In *Proceedings of Text Retrieval Conference*, 2009.

[3] K. Balog, P. Serdyukov, and A. P. de Vries. Overview of the TREC 2010 Entity Track. In *Proceedings of Text Retrieval Conference*, 2010.

[4] N. Craswell, A. P. de Vries, and I. Soboroff. Overview of the TREC 2005 Enterprise Track. In *Proceedings of Text Retrieval Conference*, 2005.

[5] H. Fang and C. Zhai. An Exploration of Axiomatic Approaches to Information Retrieval. In *Proceedings of SIGIR*, 2005.

[6] X. Liu and H. Fang. A Study of Entity Search in Semantic Search Workshop. In *Proceedings of Semantic Search Workshop, World Wide Web Conference*, 2010.

[7] M. Mavronicolas and D. Roth. Sequential Consistency and Linearizability: Read/Write Objects. In *In Proceedings of the 29th Annual Allerton Conference on Communication, Control and Computing*, pages 683–692, 1991.

[8] I. Soborof, A. P. de Vries, and N. Craswell. Overview of the TREC 2006 Enterprise Track. In *Proceedings of Text Retrieval Conference*, 2006.

[9] C. Zhai and J. Lafferty. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *Proceedings of CIKM-01*, 2001.