Ranking of Resulting Objects and Snippet Generation for Falcons

Pokhar M Jat
Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar, India

Manoj K Jain
Mohan Lal Sukhadia University, Udaipur, India

Deepak Sengar
Dhirubhai Ambani Institute of Information and Communication Technology,
Gandhinagar, India

ABSTRACT
Semantic web search engine Falcons support keyword based search for linked objects by using comprehensive virtual document which it creates for each object. In our work we are suggesting idea of using Selectivity Estimation of triple patterns for ranking of resulting objects and generating snippet for the keyword query for Falcons Semantic web search engine. Selectivity of a triple pattern is the fraction of triple satisfying the keyword query. Our work ranks resulting objects considering their relevance to the keyword query. For each resulting object for a searched keyword query, Object-rank is calculated by calculating the query-relevant-index for each RDF triple related to the object. For each resulting object, a query relevant structured snippet is provided to show the associated literals and linked objects matched with the query. Snippet generation is also done by query-relevant index of RDF triples related to the resulting object.

General Terms
Searching Semantic Web, Resource Ranking on Semantic Web.

Keywords
Semantic Web, Falcons, Snippet Generation, Resource Ranking.

1. INTRODUCTION
In this paper, we focus on ranking of resulting objects for a keyword query search for Falcons semantic web search engine and also on the generation of snippet for each object. Semantic search ([4], [11], [12]) promises to provide more accurate results Ranking of searched results is a fundamental and crucial subtask of query execution for search engines. In Falcons, the system constructs for each object a comprehensive virtual document ([1], [8], [10]) consisting of textual descriptions extracted from its concise RDF description [1]. We focus on the use of idea of selectivity estimation of RDF triple patterns [7] for ranking of resulting objects. Selectivity of a triple pattern [5] based on the keyword query can be calculated easily which in turn will make object's ranking easier to calculate. The objects' ranking goal is to find the efficient order of representation of results for the searched keyword query.

For each resulting object, the system generates a query-relevant structured snippet [1] to show the associated literals and linked objects matched with the keyword query. The snippet can help the user quickly determine its relevance or even directly obtain knowledge. The triples with the higher selectivity value, which is easy to calculate and is based on searched keyword query, is most likely to be a part of snippet.

The problem we are going to tackle in this paper is best explained by a simple example. Consider, hypothetically, searching a keyword based query (e.g. "Chris Bizer" "Tom Heath") on Falcons semantic web search engine and 3 objects are returned as results. The next task is to rank these 3 objects and generate snippet for each of these 3 objects. Again hypothetically if each of these 3 objects contain 3 triples. Then we can calculate query relevant index of three triples by using idea of selectivity for each of the three objects. Knowing the query relevant index of all the triples of an object, we can calculate object-rank for that particular object. One with the highest object-rank will be ranked first, the next lower will be ranked second and so on. For generating snippet for each object, triple (or sentence) with highest query relevant index will be selected first, and then the next lower will be second and so on(depending upon how many threads should be suitable for a snippet)

2. PRELIMINARIES
For Falcons semantic web keyword based search engine, the system constructs for each object a virtual document. Then an inverted index is built from terms in virtual documents to object. For constructing virtual document RDF Sentence are used. An RDF graph [7] g can be decomposed into a unique set of RDF sentences, denoted by Sent(g) Some important factors are describes as follows:

2.1 A. RDF Sentence
Two RDF Triples are called b-connected if they contain common blank nodes. In an RDF graph, RDF sentence [2] is a set of b-connected RDF triples. See also [3].

In an RDF graph g as a set of RDF Triples, an RDF sentence $s \subseteq g$ satisfies the following conditions:

- $\forall$ ti, tj ϵ s,,ti, tj are b-connected

  $\forall$ ti ϵ s , tj ϵ g \ s,,ti, tj are not b-connected
- Where ti, tj are RDF Triples

We can define

Sub(s) = {s|$\exists$ <s, p, o> ϵ s},

Pred(s) = {p|$\exists$ <s, p, o> ϵ s},

Obj(s) = {o|$\exists$ <s, p, o> ϵ s},

Where, Sub(s) is subject of RDF sentence s, and

Pred(s) is predicate of RDF sentence s, and

Obj(s) is object of RDF sentence s.

2.2 Virtual Documents

To implement keyword-based search for web pages, an inverted index from terms to web page URIs is built. For Semantic Web, an object identified by a URI has no such content except for the URI itself. So its virtual document is made that includes its data so that an inverted index from terms in virtual documents to objects can be built along with other important data that it possesses which is used to rank the object.

2.3 Constructing Virtual Documents with RDF Sentences

Let G be the universal RDF graph combined from all the RDF documents in the data set. Let U and L be the sets of all URIs and all literals, respectively. For an object o, all the RDF sentences decomposed from G that describe o form its concise RDF description (RDFDesc(o)) . We can also define the sets of entities(DescEnt(o)) and literals(DescLit(o)) which are related to object o.

- If Object o is a Part of Subject of Sentence(or RDF Triples): RDFDesc(o) = {s|s ϵ Sent(G) ∧ o ϵ Subj(s)}, DescEnt(o) = {u ϵ U| ∃ s(s ϵ RDFDesc(o) ∧ u ϵ (Pred(s) U Obj(s))) }.
- If Resulting Object o is a Part of Object of Sentences (or RDF Triples): RDFDesc(o) = {s|s ϵ Sent(G) ∧ o ϵ Obj(s)}, DescEnt(o) = {u ϵ U| ∃ s(s ϵ RDFDesc(o) ∧ u ϵ (Pred(s) U Subj(s))) }

  RDFDesc [1] gives the set of all possible RDF sentences(or triples) for object o. RDFDesc of 1 and 2 together give all RDF sentences(or triples) related to object o.

  DescEnt [1] of 1 and 2 together give URIs of objects related to object o
- 3.Literals which could be Related to o: DescLit(o) = {l ϵ L| ∃ s(s ϵ RDFDesc(o) ∧ l ϵ Obj(s)) }

  Using 1, 2 and 3, virtual document ([1], [8], [11]) of an object o is calculated

3. RANKING RESULTING OBJECTS USING SELECTIVITY OF TRIPLE PATTERNS

For each resulting object, we have RDF Triples related to the object. An RDF triple has three parts, subject, predicate and object respectively. Selectivity of triple pattern is calculated based on keyword query.

Selectivity of a triple pattern is calculated as follows:

If an RDF triple pattern is represented as t, and subject(s), predicate(t), and object(o) are parts of RDF triple,

Sel(t): Selectivity of triple pattern t based on keyword query,

Sel(s): Selectivity of subject s of RDF triple based on keyword query,

Sel(p): Selectivity of predicate p of RDF triple based on keyword query,

Sel(o): Selectivity of object o of RDF triple based on keyword query.

Sel(obj): Selectivity of resulting object obj based on keyword query,

Sel(sent) : Selectivity of RDF sentence based on the keyword query.

Sel(sB) : Selectivity of subject of an RDF Triple which is a blank node.

Sel(oB) : Selectivity of object of an RDF Triple which is a blank node.

3.1 Relation Between Sel(t), Sel(s), Sel(p), and Sel(o):

See [5], Sel(t) = Sel(s) * Sel(p) * Sel(o)

Thus, the expected selectivity of t, Sel(t), is modeled as the multiplication of the expected selectivity of subject Sel(s), predicate Sel(p), and object Sel(o).The selectivity function returns a value between 0 and 1,thus, it is basically a normalization to [0,1] of the estimated selectivity.

3.2 Selectivity Estimation:

In this section we calculate selectivities of subject, predicate and object of triple pattern based on keyword query.

1.*Subject Selectivity Estimation(Sel(s)) Based on Keword Query: There are three possible cases.*

- Subject of triple matches with the keyword query(or part of the keyword query), then

  Sel(s) = 1
- •Subject of triple does not match with the keyword query(or part of the keyword query), then

  Sel(s) = 1/|R|

  Where |R| is total number of resources
- If subject of triple is a blank node,then

If a blank node ([6], [9]) appears at the subject position of an RDF Triple, then this blank node must have been also an object of a previous RDF Triple. So we can calculate selectivity of a blank node at the subject position as follows.

If t1 is an RDF triple which has a blank node say b at the position of subject, and t2 is the RDF triple(previous) having b at its object position.

Sel(sB) = average of selectivities of object of triple t1 and subject of triple t2.

Or Sel(sB) = (Sel(t1.o) + Sel(t2.s)) / 2 ;

Where, sB represents a blank node at the subject position, and Sel(t1.o) is selectivity of object for the RDF triple t1, and Sel(t2.s) is the selectivity of subject for the RDF Triple t2.

3.2.1 Predicate Selectivity Estimation(Sel(p)) Based on Keword Query: There are two possible cases.

- predicate of triple matches with the keyword query(or part of the keyword query), then

  Sel(p) = 1
- Predicate of triple does not match with the keyword query(or part of the keyword query), then

Sel(p) = |Tp|/|T|, see [5]

Where |Tp| corresponds to the number of triples matching predicate p, and |T| is total number of triples. This is the fraction of triples which matches pattern p.

3.2.2 Object (of RDF Triple) Selectivity Estimation(Sel(o)) Based on Keword Query: There are four possible cases.

- Object of triple matches with the keyword query(or part of the keyword query), then

  Sel(o) = 1/|R|

- Object of triple does not match with the keyword query(or part of the keyword query), then

  Sel(o) = 1/|R|

  Where |R| is total number of resources.

- Object of triple does not match with the keyword query(or part of the keyword query) and it is a literal, then

  Sel(o) = 1/|T|

  Where |T| represents total number of RDF triples

- If object of triple is a blank node,then

If a blank node([6], [9]) appears at the object position of an RDF Triple, then this blank node must have been also a subject of some following(next) RDF Triple. So we can calculate selectivity of a blank node at the object position as follows:

If t1 is an RDF triple which has a blank node say b at the position of object, and t2 is the RDF triple(following) having b at its subject position, then

Sel(oB) = average of selectivities of subject of triple t1 and object of triple t2.

Or, Sel(oB) = (Sel(t1.s) + Sel(t2.o)) / 2

Where, oB represents a blank node at object position of an RDF triple, and

Sel(t1.s) is selectivity of subject for the RDF triple t1, and

Sel(t2.o) is the selectivity of object for the RDF Triple t2

3.2.3 Selectivity of Resulting ObjectsBased on Keyword Query:

After calculating selectivity of all triples based on keyword query related to a resulting object, we can calculate selectivity of object based on keyword query as follows

Sel(obj) = sum of selectivities of all triples related to the object obj.

$$\text{Or,} \quad Sel(obj) = \sum_{i=1}^{n} Sel(ti)$$

where, n is total no. of related triples to resulting object obj, and ti is a related triple of resulting object obj.

3.2.4 Ranking of Resulting objects:

After calculating selectivities of all resulting objects based on the keyword query, we rank them as follows: There are two possible cases.

- No. Of Related Triples are same (or almost same) for Resulting Objects: Resulting object with the highest selectivity will be ranked first, next lower will be ranked second and so on.

- No. of Related Triples are Different for Different Resulting Objects: here if we apply rule of case 1, then object ranking might not be perfect. To avoid it and to rank in a fine manner, we introduce the concept of a Virtual default triple

Virtual Default Triple: It is an assumed (which does not exist in real) related triple to a resulting object, and its selectivity would be the selectivity which has highest frequency for related triples to the resulting object.

So if two resulting objects have different no. of related triples, we make them equal by adding virtual default triple to the object having less number of related RDF triples, and then selectivity of object (after adding virtual default RDF triple) is recalculated and then they are compared for ranking of objects.

If there are two resulting objects for a keyword query, first is obj1 having m no. of related triples, and the other is obj2 having n no. of related triples and m<n, then

Since m < n, selectivity of obj1 will be recalculated as follows:

Sel(obj1) = sum of selectivities of all real related triple for obj1 + (n-m)*selectivity of virtual default triple.

$$\text{Or,} \quad Sel(obj1) = \sum_{i=1}^{n} Sel(ti) + (n-m) * Sel(VDefT),$$

where, Sel(VDefT) is selectivity of virtual default triple,

n is total no. of real related triples to resulting object obj, and

ti is a real related triple of resulting object obj1

Now comparision is done between Sel(obj1) and Sel(obj2), one with higher value will be ranked first and the next lower will be ranked second . If there are more resulting objects for a searched keyword query with different number of related triples the same approach will be followed.

4. SNIPPET GENERATION

In this section we will describe, how snippet would be generated of a resulting object for a searched keyword query by using selectivity of RDF sentences.

1.Selectivity of an RDF Sentence for a Resulting Object Based on Keyword Query : For each RDF sentence [2] related to a single resulting object, we can calculate its selectivity(based on the keyword query) as follows:

An RDF sentence [2] is made by connecting RDF triples via blank nodes ([6], [9]). So by using selectivities of RDF triples which make an RDF sentence, we can calculate selectivity of that RDF sentence:

Sel(sent) = Sum of selectivities of all RDF triples which form RDF sentence sent.

$$\text{Or,} \quad Sel(sent) = \sum_{j=1}^{m} Sel(tj)$$

where, m is the number of triples which make an RDF sentence for a single resulting object, and

tj is one of the RDF triple of an RDF sentence of a resulting object.

2.Ranking of RDF Sentences for Generating Snippet for a Resulting Object: First we will define a thread. A thread could be a RDF Triple if and only if it does not contain any blank node or a thread could be an RDF Sentence. A thread would be an RDF sentence when blank nodes exist in RDF Triples for that RDF sentence. Secondly we should decide how many threads should be there in a snippet of a resulting object for a searched query. This could be decided by considering following points in mind.

- User should be able to see sufficient number of resulting objects on the resulting page.
- Searched keyword query could possibly be answered in the snippet itself

For our case we are considering no. of threads should be 3 for a snippet.

For each RDF Sentence of a resulting object, selectivity is calculated. Then, Selectvities of RDF Sentences for that object are compared. Now we select m no. of RDF sentences with highest selectivities where m is the number of threads which would make the snippet for a resulting object. Value of m is 3 for our case, so we will select 3 RDF sentences (threads) with highest selectivities which would make snippet for the resulting object.

5. CONCLUSION

In this paper, we have presented the idea of ranking of resulting objects and snippet generation for a searched keyword query for Falcons keyword-based semantic web search engine. To meet the challenge idea of selectivity has been used. Selectivities have been calculated based on the searched keyword queries.

Firstly, objects are searched for a keyword query, then they are ranked. RDF Sentences (or RDF Triples) related to each object are used for ranking of the resulting object. Selectivity of an RDF triple is fraction of the triple satisfying the searched keyword query. Based on selectivities of all related RDF Triples, selectivity of the resulting object has been calculated. Selectivities of all resulting objects for a searched keyword query are compared to rank them, one with higher selectivity will be ranked higher. For snippet of a resulting object, threads (RDF Triples or RDF sentences) which have higher values of Selectivities than the other threads have been selected.

6. REFERENCES

[1] Cheng Gong & Qu Yuzhong, Searching Linked Objects with Falcons: Approach, Implementation and Evaluation. In International Journal on Semantic Web and Information System, 5(3), 50-71, July-September 2009

[2] X. Zhang, G. Chang & Y. Qu (2007). Ontology Summarization based on RDF Sentence Graph. In C. Williamson, M. E. Zurko, P. Patel-Schneider, & P.Shenoy (Eds.), Proceedings of the 16th International Conference on World Wide Web (pp. 707-716). New York,NY,USA: ACM

[3] G. Tummarello, C. Morbidoini, R. Bachmann-Gmur, & O. Erling (2007). RDF Sync: Efficient Remote Synchronization of RDF Models. In K. Aberer et al. (Eds.), Proceedings of the 6th Semantic web Conference and the 2nd Asian Semantic web Conference (Vol. 4825, pp. 537 - 551). Berlin/Heidelberg: Springer.

[4] T. Tran, P. Cimiano, S.Rudolph, & R. Studer (2007). Ontologu Based Interpretation of Keywords for Semantic Search. In K. Aberer et al. (Eds), Proceedings of the 6th Semantic web Conference and the 2nd Asian Semantic web Conference (Vol. 4825, pp. 523 - 536). Berlin/Heidelberg: Springer.

[5] A. Bernstein, C. Kiefer, and M. Stocker. OptARQ: A SPARQL Optimization Approach based on Triple Pattern Selectivity Estimation(Section 2). Technical Report IFI-2007.02, Department of Informatics, University of Zurich, 2007..

[6] M. Arenas, M. Consens, and A. Mallea. Revisiting Blank Nodes in RDF to Avoid the Semantic Mismatch with SPARQL. (Section 2)

[7] G. Antoniou, and F. van Harmelen. Semantic web Primer, 2nd edition. Chapter 3 & 4.

[8] C. Waters (1999). Information Retrieval and Virtual Document. Journal of the American Society for Information Science, 50(11), 1028-1029.

[9] W3C, Resource Description Framework(RDF): Concepts and Abstract Syntax, Section 6.6 Blank Nodes.

[10] Y. Qu, W. Hu, & G. Chang (2006). Constructing Virtual Documents for Ontology Matching. In L. Carr, D. D. Roure, A. Iyengar, C. Goble, & M.Dahlin (Eds.), Proceedings of the 15th International Conference on World Wide Web (pp. 23-31). New York, NY, USA:ACM.

[11] Q. Zhou, C. Wang, M. Xiong, H. Wang, & Y. Yu (2007). SPARK: adapting keyword query to semantic search. In K. Aberer et al. (Eds.), Proceedings of 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (Vol. 4825, pp. 694-707), Berlin/Heidelberg: Springer.

[12] H. Wang, K. Zhang, Q. Liu., T. Tran, & Y. Yu. (2008). Q2Semantic: a Lightweight Keyword Based Interface to Semantic Search. In S. Bechhofer, M. Hausewirth, J. Hauffmann, & M. Koubarakis (Eds.), Proceedings of 5th European Semantic Web Conference (Vol. 5021, pp. 584-598). Berlin/Heidelberg: Springer