Top-K Diversification for Path Queries in Knowledge Graphs

Christian Aebeloe[1], Vinay Setty[1,2], Gabriela Montoya[1], and Katja Hose[1]

[1] Aalborg University, Denmark
{caebel,vinay,gmontoya,khose}@cs.aau.dk
[2] University of Stavanger, Norway
vsetty@acm.org

Abstract. To explore the relationships between entities in RDF graphs, property path queries were introduced in SPARQL 1.1. However, existing RDF engines return only reachability of the entities ignoring the intermediate nodes in the path. If the paths are output, they are too many, which makes it difficult for users to find the most relevant paths. To address this issue, we propose a generalized top-k ranking technique that balances the trade-off between relevance and diversity. We propose a shortest path based relevance scoring in combination with several path similarity measures for diversification. With preliminary experiments and examples, we show that our diversification strategies provide more novel paths as our technique prioritizes diversity over path length.

1 Introduction

Knowledge Graphs (KGs) have become a popular way to represent and query world knowledge. KGs such as YAGO [6] and DBPedia [3] are extensively used for tasks, such as question answering, knowledge exploration, and reasoning. RDF[3] and SPARQL[4] are standards for representing and querying KGs, recommended by the W3C and widely adopted by the Semantic Web community.

SPARQL 1.1[5] introduces property path queries to check only the transitive reachability of entities via paths of specific properties while omitting the actual paths connecting the entities. For example, a property path query with the clause "`Alan_Turing linksTo* Princeton_University`" checks if `Princeton_University` is transitively reachable from `Alan_Turing` via any number of `linksTo` property relationships but omits the intermediate entities in the path. To address this issue, we have proposed an additional operator $\rightarrow$ that returns the full paths rather than only performing reachability checks [1]. Since enumerating all the paths may be overwhelming, we need ranking techniques to select the top-k most informative paths for the users.

Related work on ranking and diversifying property path query results is currently very limited as supporting reachability checks already is a very challenging task. Some recent works rank the paths between entities based on their lengths and frequency of

[3] https://www.w3.org/RDF/
[4] https://www.w3.org/TR/rdf-sparql-query/
[5] https://www.w3.org/TR/sparql11-query/

resources [8]. However, they only support queries involving a property path with limited length rather than involving arbitrary length property paths. Therefore, we need a general top-k ranking technique supporting arbitrary property path queries.

Selecting paths based on top-k shortest path ranking is a good strategy in general, but shortest paths may have several redundant resources (entities and properties). Therefore, it is also essential to apply *diversification* techniques to avoid redundancy, while still minimizing the path lengths. However, existing techniques do not balance the trade-off between path lengths and diversity. Path diversification using the Jaccard similarity measure has been proposed before which works well in some cases [8]. But Jaccard measure treats paths as sets and ignores the order of entities. To the best of our knowledge, there are no diversification measures considering the semantics of paths.

To address these issues, our contributions in this paper are: (1) we formulate a generalized top-k ranking technique for property paths that balances the trade-off between path lengths and diversity. (2) we then propose the Levenshtein similarity measure for respecting path semantics. (3) we perform preliminary evaluations and compare the effectiveness of the two diversification strategies using an evaluation metric to measure novelty and average path lengths.

2 Diversification Metrics for Property Paths

Given a property path query Q and the set of all relevant paths R, our goal is to select $S \subseteq R$ to maximize the objective given below:

$$DivP(R) = \arg\max_{S \subseteq R} \sum_{P_i \in S} (1-\lambda) \cdot Rel(P_i, R) + \lambda \cdot (1 - Sim(P_i, S)) \quad s.t. |S| = k \tag{1}$$

Where $Rel(P_i, R)$ is a function quantifying the relevance of a path P_i to a query Q with result set R. Each path P contains a set of entities and property edges connecting them, which we call "resources", and each path has a path length represented as $|P|$, which corresponds to the total number of resources in the path.

To avoid paths with redundant resources, the objective function in Equation 1 aims at balancing relevance (Rel) and similarity (Sim) among paths in S. The trade-off between these two scores is balanced by a user-specified diversification parameter $0 \leq \lambda \leq 1$. If $\lambda = 0$, we rank the paths purely according to their lengths. On the other hand, if $\lambda = 1$, the k most dissimilar paths according to some similarity measure irrespective of their path lengths are chosen. Computing a solution for $DivP(R)$ is known to be NP-hard [4]. However, since the objective follows the submodularity property, there are known greedy heuristics for solving it approximately [1, 7].

In this paper, we quantify the relevance relative to the shortest path in the result set. A path is more relevant if its path length is closer to the shortest path's length. We compute the normalized relevance score as $Rel(P_i, R) = (\min_{P_j \in R} |P_j|)/|P_i|$. We can also use other scoring models such as weighted path lengths and informativeness measures [8]. In this paper, we explore several Sim functions that we can use for diversification.

Jaccard similarity: The simplest way to quantify the overlap in paths is to treat them as sets and compute the Jaccard similarity value. Jaccard similarity is defined as:

$$Sim_J(P_i, S) = \max_{P_j \in S, P_j \neq P_i} \frac{|P_i \cap P_j|}{|P_i \cup P_j|} \tag{2}$$

Since Jaccard similarity captures the overlap in paths by treating them as sets, it ignores the order of resources in the paths. Intuitively, Sim_J cannot distinguish between two paths with identical resources but connected in a different order. To address this issue, we need a similarity measure that preserves the order of sequences.

Levenshtein similarity: To quantify the path sequence similarity, we adapt the Levenshtein distance [5] (LD), which is largely used to quantify the distance between two strings as the minimal number of insertions, deletions, and substitutions of one character for another that will transform one string into the other. For diversification of property paths, in Equation 1, we normalize the similarity value computed from LD as follows:

$$Sim_L(P_i, S) = \max_{P_j \in S, P_j \neq P_i} 1 - \frac{LD(P_i, P_j)}{max(|P_i|, |P_j|)} \tag{3}$$

3 Evaluation

For evaluation we use YAGO [6], which contains 1+ billion triples. To evaluate the effectiveness of the aforementioned diversification metrics, we define an evaluation metrics inspired by the novelty metric used by Arnaou et al. [2]. Evaluations are performed on sets of top-k paths of R, $[P_1, ..., P_k]$, sorted according to Equation 1 in descending order. We define an evaluation metric Nov to capture the *novelty* based on the fraction of unseen elements of each path in the top-k results:

$$Nov(P_i, [P_1, ..., P_k]) = \frac{|\mathcal{F}(P_i) \setminus \bigcup_{1 \leq j < i} \mathcal{F}(P_j)|}{|\mathcal{F}(P_i)|} \tag{4}$$

Where $\mathcal{F}(P_i)$ is a function for computing the set of elements in the path P_i. Here we observe that the set of elements of a path can be at the granularity of *resources* or *triples*. The former treats each unseen individual resource such as entities and properties in the path as a novel element, while the latter considers novel elements at the granularity of triples. Therefore, we define two variants of Nov–Nov_R and Nov_T which replace $\mathcal{F}(P_i)$ in Equation 4 with $resources(P_i)$ and $triples(P_i)$ respectively, representing set of resources and triples in the path P_i respectively. Intuitively, Nov_T captures the novelty of a path respecting semantics of paths.

Table 1. Mean novelty and path length values for top-10 paths

	Nov_R		Nov_T		Path length	
λ	**Jaccard**	**Levenshtein**	**Jaccard**	**Levenshtein**	**Jaccard**	**Levenshtein**
0.0	0.54	0.54	0.93	0.93	2.00	2.00
0.2	0.62	0.59	0.97	0.95	2.00	2.00
0.4	0.65	0.60	0.97	0.96	2.18	2.18
0.6	0.69	0.66	0.99	0.98	2.83	2.87
0.8	0.82	0.71	0.98	0.99	4.97	3.57
1.0	0.81	0.69	0.96	0.98	5.95	3.92

We perform evaluations on 10 randomly chosen queries. Table 1 shows the mean novelty metrics and path lengths, when varying the λ-values from 0 to 1.0, with $k = 10$. When $\lambda = 0$, we observe that the novelty and the path lenghts are identical for both approaches. This is expected, as the paths are ranked solely based on the path length and diversification plays no role.

Since Nov_R measures the fraction of unseen resources in the ranked paths, it is natural to observe that Jaccard consistently achieves higher Nov_R values than Levenshtein. However, when we consider Nov_T, which measures the fraction of unseen triples, Levenshtein performs better with higher values of λ (> 0.6). For lower values of λ, Jaccard still performs slightly better than Levenshtein. This could be attributed to the fact that in shorter paths, there are fewer triples and hence the impact of Levenshtein is not significant. However, this needs further analysis which we leave as a future work.

In Table 1, we can also observe that until $\lambda = 0.4$ the path lenghts are identical for both Jaccard and Levenshtein. For $\lambda > 0.6$, Levenshtein, prefers shorter paths with higher novelty w.r.t Nov_T, while Jaccard prefers longer paths because it treats shorter paths with same resources but connected in different order as similar. Since we are ranking and diversifying paths this behavior is undesirable. This demonstrates that Levenshtein is able to achieve the trade-off between path lengths and diversity.

Due to lack of space we omit results with different values of k and specific examples, but they can be found on our website, http://qweb.cs.aau.dk/jedi/.

4 Conclusion

In this paper, we addressed the issue of ranking paths in KGs. We proposed a generalized top-k diversification technique that is customizable with different relevance and similarity functions. We proposed a new similarity measure *Levenshtein*, which respects the order of nodes in the paths. We conducted preliminary experiments using novelty metrics and path lengths and show that Levenshtein provides better balancing in trade-off between path lengths and diversification than Jaccard. However, Levenshtein does not always outperform Jaccard and there is a need for further analysis in this regard. Our diversification technique has been integrated into JEDI [1], which can be used to observe the impact of diversification when evaluating queries over different KGs.

Acknowledgment This research was partially funded by the Danish Council for Independent Research (DFF) under grant agreement no. DFF-4093-00301 and Aalborg University's Talent Management Programme.

References

1. C. Aebeloe, G. Montoya, V. Setty, and K. Hose. Discovering Diversified Paths in Knowledge Bases. *PVLDB*, 11(12), 2018.
2. H. Arnaout and S. Elbassuoni. Result Diversity for RDF Search. In *KDIR*, pages 249–256, 2016.
3. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, pages 722–735. Springer, 2007.
4. J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR'98*, pages 335–336, 1998.
5. V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
6. F. Mahdisoltani, J. Biega, and F. Suchanek. YAGO3: A knowledge base from multilingual wikipedias. In *CIDR'14*, 2014.
7. G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, 14(1):265–294, 1978.
8. G. Pirrò. Explaining and suggesting relatedness in knowledge graphs. In *ISWC*, pages 622–639, 2015.