Harnessing Relationships for Domain-specific Subgraph Extraction: A Recommendation Use Case

Sarasi Lalithsena
Kno.e.sis Center
Wright State University
Dayton, OH, USA
sarasi@knoesis.org

Pavan Kapanipathi
IBM T.J. Watson Research Center
Yorktown, NY, USA
kapanipa@us.ibm.com

Amit Sheth
Kno.e.sis Center
Wright State University
Dayton, OH, USA
amit@knoesis.org

Abstract*—Applications on the Web such as search engines and recommendation systems are increasingly adapting semantic approaches by leveraging knowledge graphs. While some applications require processing of the whole knowledge graph, most are domain-specific and require only a relevant subset of it. For example, a movie or a book recommendation system would require a subgraph that comprises knowledge relevant to the specific domain. In such scenarios, processing the whole knowledge graph, particularly the commonly used, large, and openly available knowledge graphs on the Web, is computationally intensive and the irrelevant portion may negatively impact the performance of the application. This necessitates the identification and extraction of relevant subgraphs that adequately captures entities and their relationships for a given application domain and/or task. In this work, we present an approach to identify a minimal domain-specific subgraph by utilizing statistic and semantic-based metrics. Our approach highlights the importance of relationships as first-class elements to capture the domain specificity of a subgraph. We demonstrate the applicability of this approach for a recommendation use case on two domains, i.e. movie and book. Our evaluation demonstrates a reduction of 80% to 90% of the knowledge graph with orders of magnitude decrease in time for computation without compromising accuracy.

***Keywords*-Domain-specific knowledge graph, Relationship ranking, Recommendation**

I. Introduction

In recent years, knowledge graphs (KGs) available on the Web such as DBpedia, Freebase, and Yago have been increasingly used to incorporate semantics into various applications such as recommendation systems [1], document similarity [2], named entity disambiguation [3], and social media analysis [4]. These KGs are large and often grow rapidly in size. For example, english DBpedia [5] version has more than 4.5 million entities and over 1 billion facts. The use of the complete KG for the above applications can be computationally intensive. Furthermore many of the these applications are domain-specific, hence may not require the complete knowledge available in the KG. For example, movie recommendations harness knowledge relevant only to movies and social media analysis tools such as Twitris run analysis on specific topics like natural disasters and elections. A key challenge is to extract the minimal domain-specific subgraph that can reduce the computation without compromising the accuracy of the application.

Existing applications extract relevant subgraphs by navigating a predefined number of hops from a set of given entities that represent a domain. Most of these applications set the predefined number of hops between 2 and 4 [1], [2], [6]–[8]. There are several limitations to this strategy: (1) the mean shortest path between entities in a KG such as DBpedia is around 5-hops [9], and navigating 4-hops from a set of entities can cover a significant part of the graph. Fig. 1 shows the percentage of unique entities reached by navigating up to 3-hops starting from 3,072 movie entities in the MovieLens[1] dataset which is a popular dataset for movie recommendation. The subgraph extracted by navigating 3-hops encompasses 66% of the DBpedia entities. In particular, it contains 2.9 million entities and 4.9 billion paths up to length 3 starting from movies. Processing of such a large subgraph can get computationally intensive, and (2) given a domain of interest, not all entities connected via a predefined number of hops are necessarily relevant. For example, two movies connected because their directors are known for *action movies* is important for a movie recommendation system, whereas two movies connected because the directors of these movies *died in the same city* is irrelevant. Hence, extracting a subgraph with a predefined number of hops can encompass paths that do not contribute to the accuracy.

In this work, we address the task of extracting a domain-specific subgraph from a large KG. Specifically, we reduce the volume of a large graph while keeping its value with respect to a given domain. Our approach identifies entities and relationships that are strongly associated to the domain of interest. In order to determine the associations, our methodology treats relationships as first-class elements because relationships induce semantics between entities and, hence, can play a significant role in identifying domain-specific knowledge [10]. The practical usage of our approach is demonstrated with a recommendation system use case.

[1]http://grouplens.org/datasets/movielens/

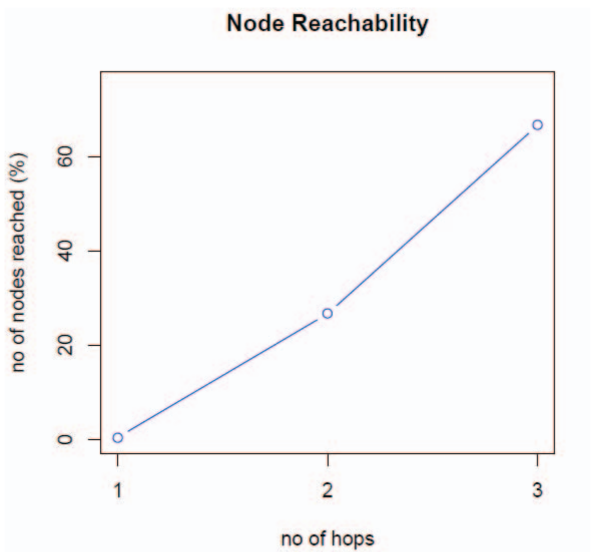

Figure 1. Percentage of nodes reached via n-hops.

We harness the domain-specific subgraphs extracted by our approach using DBpedia for recommending entities of two domains, i.e., movie and book. In the evaluation, we show that the use of a domain-specific subgraph can, (1) reduce the graph by over 80% without compromising the accuracy by identifying domain-specific entities and relationships; (2) decrease the computation time by more than tenfold in comparison to the existing methods that extract subgraphs.

The rest of the paper is organized as follows. Section II describes our approach, followed by evaluation in Section III. In Section IV, we detail the related work. Section V concludes with suggestions for future work.

II. Approach

In order to extract a domain-specific subgraph from a large KG, the primary input is *the domain*. For example, the domain for a movie or a book recommendation system is *movie* and *book* respectively. The entities of the given domain are identified as *in-domain entities* i.e. movies and books. These in-domain entities can be leveraged to initiate and expand the subgraph. Our approach uses types of the entities in the KG as the domain and entities of the given type as in-domain entities. For instance, for a movie recommendation system that utilizes DBpedia as a KG, the input to our approach would be type `dbo:Film`, and entities of type `dbo:Film` are the in-domain entities.

Starting with the in-domain entities, the extraction of the domain-specific subgraph now translates to expanding the in-domain entities with relationships and other entities that are specific to the domain of interest. Existing applications perform this expansion with a simple n-hop navigation with a pre-defined value for n [1], [7], [8]. This would end up creating a subgraph with knowledge that is not specific to the domain. In order to extract a domain-specific subgraph, this n-hop navigation should only pick the facts that are specific to the domain. For an illustrative example, consider expanding the entity `dbpedia:Titanic` to a movie domain-specific subgraph on DBpedia. Fig. 2(a) shows the subgraph extracted by navigating 2-hops from the entity `dbpedia:Titanic`. If we want to extract a movie domain-specific subgraph from Fig. 2(a) which consists of set of entities and relationships specific to the movie domain, facts such as `dbpedia:Kapuskasing` is the `dbprop:birthplace` of `dbpedia:James Cameron` and `dbpedia:Jim Threapleton` is the `dbprop:spouse` of `dbpedia:Kate Winslet` have less significance compared to facts such as `dbpedia:James Cameron` and `dbpedia:Kate Winslet` have won the `dbpedia:Academy Awards`.

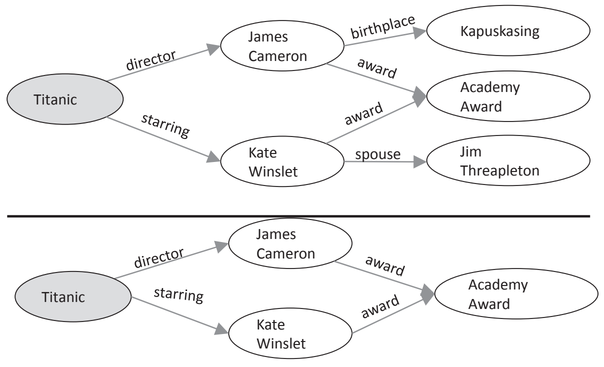

Figure 2. (a) 2-hop expansion subgraph; (b) domain-specific subgraph.

Inspired by this observation, our approach focuses on assessing the domain-specificity of a fact. The specificity of a fact with respect to a domain can intuitively be assessed using the relationships. For example, identifying that relationships `dbprop:starring`, `dbprop:director`, and `dbprop:award` are more specific to the movie domain whereas `dbprop:spouse` and `dbprop:deathdate` are less specific would help to generate the domain-specific subgraph in Fig. 2(b). Therefore, in our approach we consider relationships as the first-class elements in identifying the domain-specific subgraph and propose measures to capture the domain specificity of relationships.

To get to the crux of the approach, we formally define a KG and a domain-specific subgraph as follows:

Definition 1: A *knowledge graph* $KG = (V, E, P)$ is a graph-based data model where V is a set of entities, E is a set of labeled edges, where $E \subseteq V \times P \times V$ and P is a set of relationships.

Even though edges are directed in a KG, we assume that all semantic relations can be considered to have semantically inverse relations. Hence, our KG is undirected.

Definition 2: A *domain-specific subgraph* for domain d with a set of in-domain entities D is $DSG = (V_d, E_d, P_d)$,

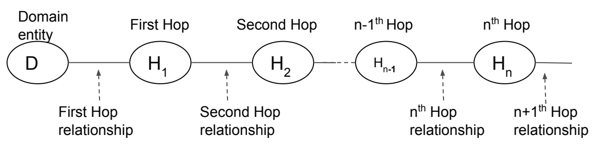

Figure 3. Hops and relationships.

where P_d is a set of domain-specific relationships with each relationship having a domain specificity score w_d, $V_d \subseteq V$ and $E_d \subseteq E$ that can be reached by navigating KG starting with D and using only P_d.

The primary focus of this work is to determine the domain specificity scores that can be utilized to restrict the graph to a domain-specific subgraph. Section II-A describes our novel measures to determine domain specificity scores w_d.

A. Domain Specificity Measures

Association between a relationship and a domain determines the domain specificity of the relationship. Since the in-domain entities represent the domain in a KG, we consider the association between a relationship and the in-domain entities as the domain specificity of the relationship. For example, relationship `dbprop:starring` is strongly associated with the movie domain because it appears specifically with the in-domain entities, whereas, relationship `dbprop:country` is generic and can be associated with many non-domain entities of types such as `dbo:Person`, `dbo:Organization` and `dbo:Place`. Therefore, we consider that the domain specificity of `dbprop:starring` is higher than the domain specificity of `dbprop:country` with respect to the movie domain.

Our approach focuses on expanding the graph starting from in-domain entities by identifying domain-specific relationships. Determining the domain specificity of the first hop relationships (relationships that connect in-domain entities to the first hop entities as shown in Fig. 3) from the in-domain entities is trivial because we can find the direct association of the relationship with the in-domain entities. For example, in Fig. 4(a), `dbprop:director` and `dbprop:country` are directly connected to in-domain entity (m_1) and its association can be determined based on its frequency of occurrence with in-domain entities versus all entities. On the other hand, measuring the domain specificity of relationships that are not directly connected to in-domain entities is non-trivial. For example, in Fig. 4(a) it is not straightforward to find the association of relationships such as `dbprop:award`, `dbprop:spouse`, and `dbprop:president` with in-domain entities because they may require more information about their connectedness to the in-domain entities. For simplicity, in the rest of the paper we refer to relationships that connect in-domain entities to their first-hop entities as *direct relationships* and the relationships that are more than one-hop away from in-domain entities as *indirect relationships*.

We identify two characteristics in the KG that can be harnessed to measure the association of relationships to in-domain entities: (1) type and (2) path. We use these as ways to obtain the connectedness of relationships to in-domain entities. In Sections II-A1 and II-A2, we detail the intuition and formalizations of measures with these characteristics.

1) Type-based scoring: For this measure, we consider the KG as entities of certain types connected via relationships. For instance, `dbpedia:Titanic` is an entity of type `dbo:Film` connected to `dbpedia:James Cameron`, an entity of type `dbo:Director` via the relationship `dbprop:director`. We utilize the association between types to determine the domain specificity of the n^{th} hop relationship. For example, as shown in Fig. 4(a), in order to calculate the domain specificity of relationships `dbprop:award` and `dbprop:spouse` that are 2^{nd} hop relationships, we utilize the strength of association between the type `dbo:Film` and `dbo:Director`. If entities of the type `dbo:Director` are strongly associated with the entities of type `dbo:Film` and the relationship `dbprop:award` is strongly associated with the type `dbo:Director`, then there is a higher likelihood that the `dbprop:award` relationship has high association to the movie domain. To capture this intuition in calculating the domain specificity of an indirect n^{th} hop relationship we utilize two factors:

(1) The strength of association between the domain entity type t_d and the entity type at the $n - 1^{th}$ hop t_{n-1} is determined by calculating the association between each pair of directly connected intermediate types leading to the $n - 1^{th}$ hop from the in-domain entity type. The strength of association between directly connected entity types t_i and t_j is determined using Eq. 1.

$$d\_typerel(t_i, t_j) = \frac{edgecount_{t_i,t_j}}{edgecount_{t_i} \times edgecount_{t_j}} \quad (1)$$

where $edgecount_{t_i,t_j}$ is the number of edges that connect entities of type t_i and t_j, and $edgecount_{t_j}$ is the number of edges with entities of type t_j.
The strength of the association between t_d and t_{n-1} connected via an ordered set of intermediate entity types $T = t_1, \ldots, t_{n-2}$ can be determined using Eq. 2

$$ind\_typerel(t_d, t_{n-1}, n) = \prod_{k=1}^{n-1} d\_typerel(t_{k-1}, t_k) \quad (2)$$

where $t_0 = t_d$. When there are multiple ordered sets of intermediate entity types that connect t_d and t_{n-1}, we calculate $ind\_typerel(t_d, t_{n-1}, n)$ for each ordered set and take the maximum value.

(2) The strength of association between entity types and their direct relationships is determined using Eq. 3,

$$proprel(p, t) = \frac{edgecount_{p,t}}{edgecount_p} \quad (3)$$

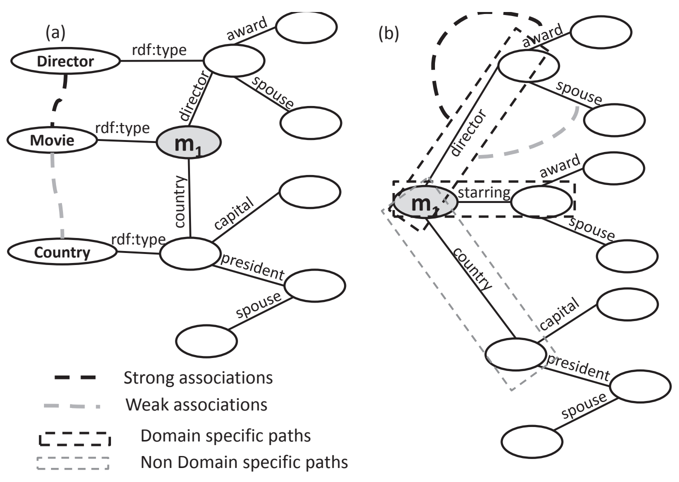

Figure 4. (a) Type-based connectedness; (b) Path-based connectedness m_1 is a movie (in-domain) entity.

where t can be any type, p is a direct relationship of t, $edgecount_{p,t}$ is the number of edges with relationship p directly connected to entities of type t, and $edgecount_p$ is the total number of edges with relationship p.

For indirect relationships, type-based scoring combines both $proprel(p, t)$ and $ind\_typerel(t_d, t_{n-1}, n)$ to compute a score for each n-hop relationship p, $propscore(p, n)$, for a given domain entity type t_d, as given in Eq 4.

$$propscore(p, n) = \sum_{t_{n-1_j} \in C} ind\_typerel(t_d, t_{n-1_j}, n) \times proprel(p, t_{n-1_j}) \tag{4}$$

where C is the set of all entity types at the $n-1^{th}$ hop that have a direct relationship with relationship p. When the same relationship appears at hops i and j, where $i < j$, we take the $propscore$ at the i^{th} hop. Type-based scoring uses Eq. 3 to find the domain specificity of direct relationships, where p is a direct relationship of t_d.

2) Path-based scoring: While type-based scoring utilizes the types of *intermediate entities* to determine the domain specificity, path-based scoring utilizes the domain specificity of *intermediate relationships*. The intuition is, if `dbprop:director` is already detected as a domain-specific relationship in Fig. 4(b), then `dbprop:award`, which is an indirect relationship connected via the intermediate domain-specific relationship `dbprop:director`, has a higher possibility to be a domain-specific relationship.

In order to determine the domain specificity of a relationship, we introduce an iterative approach that determines the domain specificity of intermediate relationships along the path from the in-domain entities. In the first iteration, it ranks direct relationships connected to the in-domain entities as given in Fig. 5(a) and selects the `top-K` relationships as domain-specific relationships. The `top-2` relationships in the example are `dbprop:director` and `dbprop:starring`. In the next iteration, as shown in Fig. 5(b), it uses the domain-specific relationships identified at the first-hop to traverse to the next hop and score the second-hop relationships such as `dbprop:knownFor`, `dbprop:award`, and `dbprop:writer`. In summary, this methodology determines the domain specificity of an n^{th} hop relationship based on its association with the ordered set of domain specific relationships from the in-domain entities up to $n-1^{th}$ hops. The ordered set of domain-specific relationships are termed as *domain-specific paths*. According to Fig. 5(b), `m1 starring x award y` is a domain-specific path formed using the ordered set of relationships `starring` and `award`. The `x` and `y` are variables and can be replaced with any entity and `m1` is any in-domain entity. In other words, the intermediate entities are of no concern in representing a domain-specific path. These are utilized to determine the domain specificity of relationships that are connected to a subgraph at the next hop.

In this approach the domain specificity of an n^{th} hop relationship depends on its association to the domain-specific path obtained from domain entities to $n-1^{th}$ hop. To measure the association, we adopt Pointwise Mutual Information (PMI) [11], which is a well-known association measure. PMI quantifies the association of two items appearing together using the co-occurrence of the two items. Here, the two items are the n^{th} hop relationship and the domain-specific paths until the $n-1^{th}$ hop. Hence, we define the path-based measure $propscore(p, n)$ of an n^{th} hop relationship p as,

$$propscore(p, n) = PMI(p, dsp_{n-1}) \tag{5}$$

$$PMI(p, dsp_{n-1}) = log \frac{Prob(p, dsp_{n-1})}{Prob(p) \times Prob(dsp_{n-1})} \tag{6}$$

where $dsp_{n-1} \in DSP_{n-1}$, DSP_{n-1} is the set of domain-specific paths up to the $n-1^{th}$ hop.

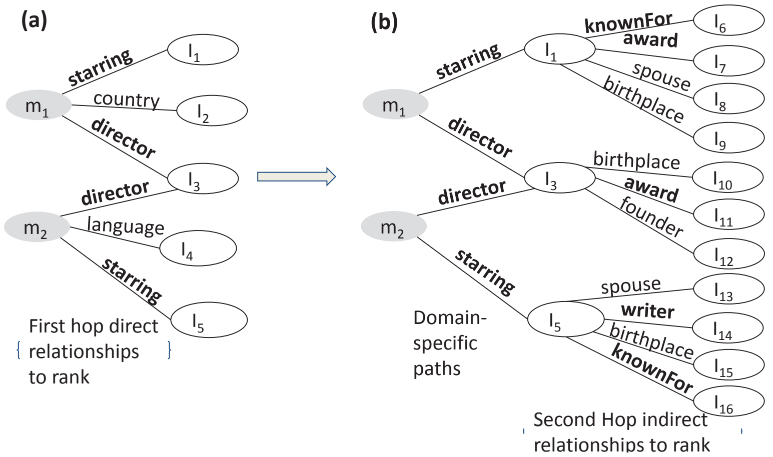

Figure 5. Iterative scoring of relationships (a) Iteration 1: scoring of direct (1-hop) relationship; (b) Iteration 2: scoring of indirect (2-hop) relationships; *m_1 and m_2 are movie (in-domain) entities.*

$$Prob(p, dsp_{n-1}) = \frac{Path(dsp_{n-1}, p)}{\sum_{p \in P} Path(dsp_{n-1}, p)} \tag{7}$$

where $Path(dsp_{n-1}, p)$ is the number of paths of length n with a domain-specific path dsp of length $n-1$ connected to p. The probability $Prob(p)$ is the fraction of edges that contain p. $Prob(dsp_{n-1})$ is the fraction of paths of length $n-1$ that have the domain-specific path dsp. If there are multiple domain-specific paths dsp_{n-1} that can reach p, we calculate the $propscore(p, n)$ for each dsp_{n-1} and take the maximum value.

PMI is known for its sensitivity to low frequency values. Therefore, we normalize the $PMI(p, dsp_{n-1})$ as proposed in [12].

III. Evaluation

To show the effectiveness of our approach, we use the domain-specific subgraph for a recommendation use case. Recent developments in recommendation algorithms [1], [7], [13], [14] recognize the importance of incorporating KGs as background knowledge. Our evaluation methodology adapts an existing recommendation algorithm as outlined in Section III-A1 and compares its performance with subgraphs generated from: (1) n-hop expansion and (2) an n-hop domain-specific subgraph. We perform the evaluation for two domains, movie and book, by using different evaluation metrics as detailed in Section III-B.

A. Evaluation Setup

1) Recommendation algorithm: We adapt an existing recommendation algorithm [13] for our evaluation. This uses KGs from Linked Open Data to calculate the similarity between the items. The similarity function used by [13] considers the entities reached via 1-hop, which restricts the recommendations of movies connected via 1-hop to the user selected movies. For a fair comparison, we replace the similarity function with a measure proposed by [15] which measures the similarity of two entities x and y connected via n number of hops. This similarity function is given below.

$$sim(x, y) = \sum_{n=1}^{r} \frac{1}{2n^2} \times \sum_{path \in Paths(x,y)} \sum_{e \in edges(path)} w(e)$$

where r is the maximum number of hops between two entities, $Paths(x, y)$ returns all the paths between two entities within n-hops, $edges(path)$ returns all the edges in the $path$, and $w(e)$ is the weight of the edge. We consider all the edges to have an equal weight.

2) Baseline - n-hop expansion subgraph: The baseline is the recommendation algorithm presented in Section III-A1 with the input KG being a simple n-hop expansion of DBpedia. We experiment with n = 2, 3.

3) Our approach - Domain-specific subgraph: While the recommendation algorithm remains same as the baseline, the input KG is created by ranking domain-specific relationships in DBpedia using type-based and path-based measures.

4) Datasets: We used popular datasets from two domains: (1) movie and (2) book. For movie domain, we used the MovieLens dataset that consists of 1,000,209 ratings for 3,883 movies by 6,040 users and for book domain we used the DBbook[2] dataset that has 72,372 ratings for 8,170 items by 6181 users. Since our recommendation algorithm is a content-based approach, users who have rated very few items significantly impacts the recommendation performance. To deal with this, we eliminated users who have less than 20 ratings as suggested in [13]. After this filtering step, the MovieLens dataset contains 5,886 users with approximately 0.9M ratings, and the DBbook dataset contains 17,802

[2]http://challenges.2014.eswc-conferences.org/index.php/RecSys#DBbook_dataset

ratings by 812 users. For each user, we take 60% of ratings as the training data and the rest 40% as the testing data.

B. *Evaluation Metrics:*

We evaluate our approach on three aspects.

- Graph reduction: Graph reduction measures the reduction of domain-specific subgraph DSG_n compared to the n-hop expansion subgraph in terms of the number of nodes, relationships, and also the number of reachable paths within n-hops starting with the in-domain entities.
- Impact on accuracy: During the process of reducing the graph, it is important to make sure that the graph reduction does not occur at the cost of the accuracy of the recommendation algorithm. To assess this, we use two measures. First we calculate the standard measure of `precision@n` as used in [13]. `precision@n`, in comparison to other metrics such as recall, is the most suitable evaluation metric for recommender systems, particularly where the number of recommended items are pre-obtained [16]. However, `precision@n` only measures binary relevancy of the items up to n^{th} rank but does not capture whether domain-specific subgraph DSG_n replaces any highly relevant items selected using n-hop expansion subgraph with relatively low relevant items. To quantify this, we leverage the ratings provided by users for each item in the gold standard datasets. Ratings provided by the users will give an indicator whether domain-specific subgraph DSG_n replaces any highly relevant items from n-hop expansion subgraph. We take the average of the ratings of the user from the gold standard dataset. Then, we calculate the deviation of the rating for each relevant top-n items from the average rating and finally take the mean of the deviation for all relevant items. A higher positive deviation value reflects better results. The measure `ratingdev` for a given user u is formalized as,

$$ratingdev(u) = \frac{\sum_{r \in R} itemrating_r - avgrating_u}{|R|}$$

  where R is the top n relevant items for user u, $itemrating_r$ is the rating given by the user u for item r, and $avgrating_u$ is the average rating for user u.
- Impact on runtime performance: Graph reduction should be able to reduce the time taken to run the recommendation algorithm. The most time consuming module in the recommendation algorithm is calculating the similarity of all item pairs as it requires traversing the whole KG. Hence, we compare the runtime performance required for this step using the domain-specific subgraph DSG_n and the n-hop expansion subgraph. We use the Neo4j graph database to navigate the graph and calculate the item similarities for recommendation.

C. *Evaluation Results*

Using the aforementioned metrics we evaluate the quality of domain-specific subgraph extracted by our approach and present the results in this section.

1) Graph reduction: Tables I, II, and III show the number of nodes, relationships, and paths reached via n-hop expansion subgraph and $DSG_{n(k_1,.,k_n)}$ with the `top-K` relationships at each hop. They also include the reduction percentage from n-hop expansion subgraph to DSG_n in parenthesis. Tables I and II show the reduction statistics for the domain-specific subgraphs created with 2-hops for the movie and book domains respectively.

Domain-specific subgraphs created with the `top-15` relationships at each hop reduce the n-hop expansion subgraph by over 80% to 90% (3^{rd} row in Tables I and II) for both the domains. As we use more relationships to extract the domain-specific subgraph, the reduction percentage is decreased. The graph reduction percentage from both type-based and path-based measures are almost similar except for movie 2-hop graph with `top-25` at each hop in which type-based reduction is less than the path-based reduction.

Table III shows graph reductions for domain-specific subgraphs created with 3-hops for both the domains.

Domain-specific subgraphs were able to reduce the graph by 90% - 96%. This is slightly higher than the reduction from 2-hop graphs. An important aspect to note here is the significant increase in the number of reachable paths when the hop size changes from 2 to 3.

2) Impact on accuracy:

precision@n: We calculate the `precision@n` for different ranks and average it over all the users. Fig. 6 shows the `precision@n` for all the users from the MovieLens dataset. We calculate the precision for both the 2-hop expansion subgraph and DSG_2. Like graph reduction, we experiment with different `top-Ks` in selecting the DSG_2.

For the movie domain, DSG_2 selected with the `top-15` relationships at each hop via path-based scoring technique has the best performance. It performs better than the baseline. This indicates that merely selecting the n-hop subgraph can also negatively impact the results. It increased the precision by 2% for `top-5` recommendations. As we increase the number of relationships it decreases the performance and converges with the baseline at `top-35` relationships. Type-based scoring also improves the baseline.

For the rest of the evaluation, we pick the domain-specific subgraph created with `top-15`, `25`, and `15` at each hop which performed best out of the various `top-K` values and present the results. Fig. 7 shows the `precision@n` over all users for the recommendation algorithm with DSG_3 for the movie domain. Fig. 8 shows the same results for the book domain with DSG_2 and DSG_3.

For the movie domain DSG_3 created with both type and path-based measures, performs equally well and outperforms the baseline. It increased the precision by 6.7% for the top

	Path-based			Type-based		
	Relations	**Nodes**	**Paths**	**Relations**	**Nodes**	**Paths**
2-hop	349	1.07M	108.4M	349	1.07M	108.4M
$DSG2_{(15,15)}$	15(95.7%)	0.08M(92.0%)	5.08M(95.3%)	14(95.9%)	0.13M(87.6%)	17M(83.9%)
$DSG2_{(25,25)}$	25(92.8%)	0.13M(87.3%)	17.4M(83.8%)	24(93.1%)	0.63M(40.9%)	61.6M(43.19%)
$DSG2_{(35,35)}$	35(90%)	0.64M(40.7%)	61.64M(43.1%)	32(90.8%)	0.64M(40.7%)	61.62M(43.18%)

Table I
GRAPH REDUCTION STATISTICS FOR 2-HOP EXPANSION SUBGRAPH AND DSG_2 ON MOVIE DOMAIN; M DENOTES MILLIONS

	Path-based			Type-based		
	Relations	**Nodes**	**Paths**	**Relations**	**Nodes**	**Paths**
2-hop	424	1.2M	793.4M	424	1.2M	793.4M
$DSG2_{(15,15)}$	15 (96.5%)	0.09M(92.8%)	159.6M(79.9%)	15(96.5%)	0.09M(92.8%)	159.7M(80%)
$DSG2_{(25,25)}$	25 (94.1%)	0.62M(49.1%)	465.6M(41.5%)	25(94.1%)	0.62M(49%)	464.4M(41.68%)
$DSG2_{(50,50)}$	50 (88.2%)	0.68M(44.8%)	484.4M(39.2%)	38(91.0%)	0.63M(48.5%)	464.6M(41.66%)

Table II
GRAPH REDUCTION STATISTICS FOR 2-HOP EXPANSION SUBGRAPH AND DSG_2 ON BOOK DOMAIN; M DENOTES MILLIONS

	Path-based			Type-based		
	Relations	**Nodes**	**Paths**	**Relations**	**Nodes**	**Paths**
Movie-3-hop	636	2.86M	4885.3M	636	2.86M	4885.3M
$Movie - DSG3_{15,25,15}$	30(95.3%)	0.19M(93.2%)	48.8M(98.9%)	24(96.2%)	0.26M(90.9%)	105.5M(97.83%)
$Book$−3-hop	641	3.2M	13852.8M	641	3.2M	13852.8M
$Book - DSG3_{15,25,15}$	31(95.2%)	0.18M(94.2%)	1082.6M(92.18%)	21(96.7%)	0.12M(96%)	1062.5M(92.33%)

Table III
GRAPH REDUCTION STATISTICS FOR 3-HOP EXPANSION SUBGRAPH AND DSG_3 ON MOVIE AND BOOK DOMAINS; M DENOTES MILLIONS

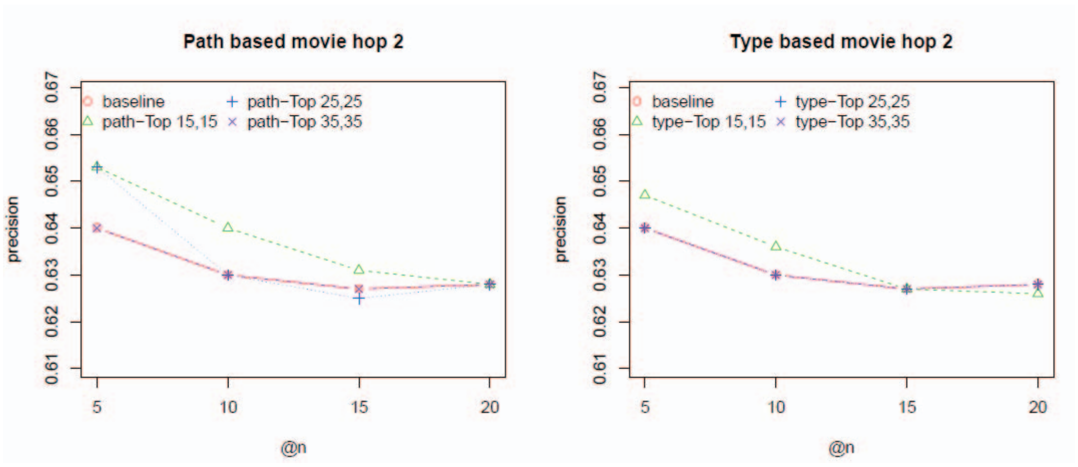

Figure 6. Precision for 2-hop expansion subgraph and movie DSG_2 on the movie domain.

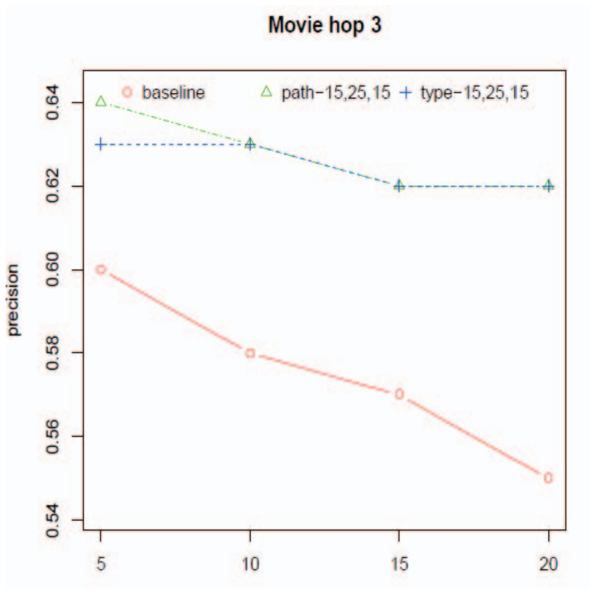

Figure 7. Precision for 3-hop expansion subgraph and movie DSG_3 on movie domain.

	2-hop		3-hop	
	Baseline	$DSG_{215,15}$	**Baseline**	$DSG_{315,25,15}$
5	0.822	0.823	0.807	0.823
10	0.814	0.816	0.806	0.815
15	0.810	0.811	0.806	0.811
20	0.806	0.807	0.805	0.806

Table IV
DEVIATION FROM AVERAGE RATING FOR MOVIE

5 ratings. For the book domain, path-based scoring measure performs slightly better than the type-based scoring measure and is on par with the baseline. However, for the domain-specific subgraph of 3-hops, type-based scoring measure underperforms in comparison to the baseline.

ratingdev: As $precision@n$, we calculate the $ratingdev$ for different ranks and average it over all the users. We pick the path-based measures to present the results for $ratingdev$ as it shows better performance in comparison to the type-based measures. Table IV shows the deviation from the average ratings for the movie domain with the domain-specific subgraphs $DSG_{215,15}$ and $DSG_{315,25,15}$ in comparison to the n-hop expansion graph. Table V shows the same results for the book domain.

For both the domains, deviation from the average rating performs equally well or outperforms the baseline except for the 1^{st} rank for the book 2-hop graph. This indicates that our approach did not replace highly relevant items from the $n-$hop expansion subgraph and that it was also able to identify more relevant items using the DSG. For example, the top-5 items recommended by the algorithm using the n-hop expansion subgraph to a user in the MovieLens dataset

	2-hop		3-hop	
	Baseline	$DSG_{215,15}$	**Baseline**	$DSG_{315,25,15}$
1	0.592	0.584	0.533	0.558
2	0.599	0.604	0.571	0.579
3	0.601	0.614	0.569	0.579
4	0.606	0.617	0.595	0.595
5	0.610	0.620	0.596	0.6

Table V
DEVIATION FROM AVERAGE RATING FOR BOOK

only have 3 relevant items with ratings 3, 3, and 2. But using DSG, the user was given 5 relevant items with ratings 4, 4, 5, 3, and 4. $ratingdev$ was increased by 2.5% and 3.7% for the movie and book domains respectively at their highest ranks (top-5 for movies and top-1 for book) for the 3-hop domain-specific subgraph. This shows our approach particularly performs well as the hop increases.

3) Impact on runtime performance: Table VI shows the time taken to calculate the item similarity, which is the most time consuming component of recommendation system.

Results indicate a tenfold decrease in the amount of time taken to run the algorithm with a DSG in all cases. The reduction significance is higher as we increase the number of hops to cover the subgraph. As given in the Table VI, for the 3-hop subgraphs time is reduced from hours to minutes.

To summarize, the evaluation results suggest that the DSG decreases the size of the graph by an average of 80% to 90% and still performs comparable to and in some cases better than the original subgraph. Graph reduction results in a tenfold reduction in time to calculate the item similarity.

In some cases, path-based scoring measures perform better than type-based scoring. The quality of type-based scoring depends on the type assignments of the entities. Some entities are not assigned to the most specific class, and instead are assigned to a more generic class in the schema. For example, DBpedia contains only 6,591 entities of type actor while there are 80,837 unique entities which are linked to movies via the `starring` relationship.

IV. RELATED WORK

Many applications use a subgraph to incorporate the background knowledge from generic KGs like DBpedia. Some of the prominent types of applications are: (1) recommendation, and (2) named entity disambiguation. In this section, we discuss usage of the KG in each of these applications and the applicability of domain-specific subgraph extraction.

Recommendation systems mainly use KGs to capture the item relatedness in recommending items. Content-based recommendation systems [1], [7], [8], [13], [14] extract DBpedia subgraphs from a 2-hop expansion of in-domain entities with a set of manually selected relationships for the movie and book domains. A hybrid recommendation algorithm [1] extracts the DBpedia subgraph within 3-hops of movie entities to capture the relationships between the

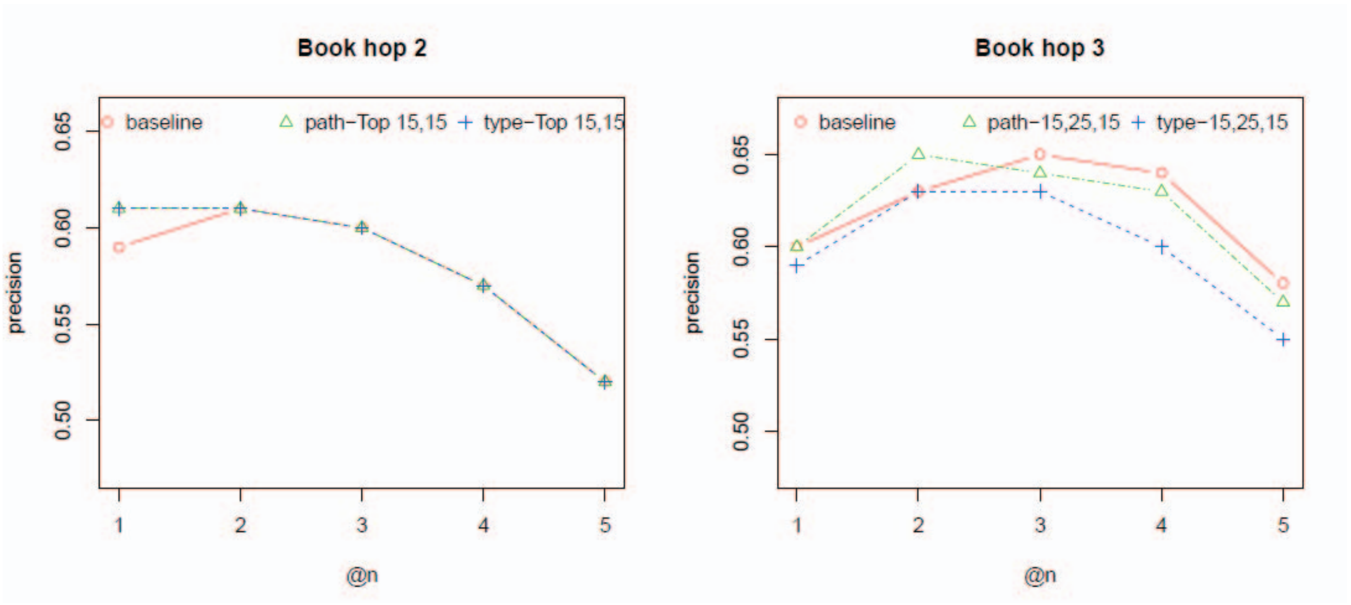

Figure 8. Precision for 2 and 3-hop expansion subgraph and DSG on book domain.

	Movie			Book		
	n-hop Expansion	DSG		n-hop Expansion	DSG	
		Path Rank	Type Rank		Path Rank	Type Rank
2-Hop	72 s	5s	11.2 s	10.15 m	1.3 m	1.4 m
3-Hop	2h 35 m	76 s	3.2 m	7 h 40 m	15.2 m	27 m

Table VI
TIME PERFORMANCE IMPROVEMENT; S - SECONDS, M - MINUTES AND H - HOURS

movies rated by the user and the movies to be recommended. Subgraphs extracted from n-hop expansion still contains a significant portion of KG with irrelevant entities and relationships. Manual selection of relationships for a given domain can help to capture the domain semantics, but this is a labor and time intensive process given the number of relationships in KGs. For instance, DBpedia has more than 1,000 relationships and only 65% of relationships have domain and range defined that help to capture the domain semantics. Also, recommendation systems rank the paths to identify the relevant paths. However, this ranking depends on the two items being compared rather than the semantics of the domain of the two items. In this work, we capture the domain semantics in extracting the subgraph for a given domain. However, the proposed techniques can be complementary to ranking relevant paths.

Recommendation is a well-suited use case for domain-specific subgraph extraction as it considers the item relatedness among in-domain entities. Even though we focus on recommendation, it is not limited to recommendation. Named entity disambiguation links the entity mention in a text to an entities in a KG. Recent approaches [3], [6] use KGs as a way to generate the context for each ambiguous entity mention. AGDISTIS [3] extracts a DBpedia subgraph with up to 3-hop of all candidates and then uses centrality-based measures to link the sense. A recent approach [6] disambiguates entities by using path-based relatedness measures among all candidate senses in the DBpedia subgraph. Even though existing techniques work well with generic entities, a domain-specific subgraph would benefit the named entity disambiguation in specialized domains like medicine and finance due to the special domain characteristics [17].

In addition to the above mentioned approaches for semantic relatedness, [18] leverages the information content of the relationships and paths for semantic relatedness. Some approaches [19], [20] also employ the schema information to further improve the information content-based methods. While existing semantic relatedness techniques consider the features of the two items being compared, we restrict the features based on a given domain. In fact, [6] emphasizes the importance of identifying relevant paths between two entities to improve the semantic relatedness measures.

V. Conclusion and Future work

In this paper, we presented a novel approach to extract the domain-specific subgraph from a large, generic knowledge graph by ranking relationships to capture their domain specificity. We experimented with two measures that utilize intermediate types and intermediate relationships to determine the domain specificity of relationships. By restricting the KG to only domain-specific relationships and entities, we demonstrated its effectiveness for a recommendation use case on two domains, movie and book. Both our domain specificity measures were able to reduce the graph size by more than 80% which led to a tenfold decrease in computation time of the recommendation algorithm. The results also showed that there was no compromise in the accuracy of recommendations but rather found more accurate results.

In the future, we will explore subgraph extraction for other possible applications like semantic similarity which contains cross-domain entities. We also plan to explore the domain specificity of the hierarchical structures, such as the Wikipedia category, and merge both hierarchical and domain relationships to extract the domain-specific subgraph.

Acknowledgment

This research was supported by NSF awards EAR-1520870 "Hazards SEES: Social and Physical Sensing Enabled Decision Support for Disaster Management and Response" and IIP-1542911 "Market Driven Innovations and Scaling up of Twitris- A System for Collective Social Intelligence". Any opinions, findings, and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi, "Top-n recommendations from implicit feedback leveraging linked open data," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 85–92.

[2] M. Schuhmacher and S. P. Ponzetto, "Knowledge-based graph document modeling," in *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 2014, pp. 543–552.

[3] R. Usbeck, A.-C. N. Ngomo, M. Röder, D. Gerber, S. A. Coelho, S. Auer, and A. Both, "Agdistis-graph-based disambiguation of named entities using linked data," in *The Semantic Web–ISWC 2014*. Springer, 2014, pp. 457–471.

[4] A. S. Jadhav, H. Purohit, P. Kapanipathi, P. Anantharam, A. H. Ranabahu, V. Nguyen, P. N. Mendes, A. G. Smith, M. Cooney, and A. P. Sheth, "Twitris 2.0: Semantically empowered system for understanding perceptions from social data," 2010.

[5] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[6] I. Hulpuş, N. Prangnawarat, and C. Hayes, "Path-based semantic relatedness on linked data and its use to word and entity disambiguation," in *The Semantic Web–ISWC 2015*. Springer, 2015, pp. 442–457.

[7] C. Musto, P. Basile, P. Lops, M. de Gemmis, and G. Semeraro, "Linked open data-enabled strategies for top-n recommendations," *CBRecSys 2014*, p. 49, 2014.

[8] A. Passant, "dbrecmusic recommendations using dbpedia," in *The Semantic Web–ISWC 2010*. Springer, 2010, pp. 209–224.

[9] "Dbpedia network dataset – KONECT," May 2015. [Online]. Available: http://konect.uni-koblenz.de/networks/dbpedia-all

[10] A. Sheth, I. B. Arpinar, and V. Kashyap, "Relationships at the heart of semantic web: Modeling, discovering, and exploiting complex semantic relationships," in *Enhancing the Power of the Internet*. Springer, 2004, pp. 63–94.

[11] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational linguistics*, vol. 16, no. 1, pp. 22–29, 1990.

[12] G. Bouma, "Normalized (pointwise) mutual information in collocation extraction," *Proceedings of GSCL*, pp. 31–40, 2009.

[13] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker, "Linked open data to support content-based recommender systems," in *Proceedings of the 8th International Conference on Semantic Systems*. ACM, 2012, pp. 1–8.

[14] G. Piao and J. G. Breslin, "Measuring semantic distance for linked open data-enabled recommender systems," in *The 31st ACM/SIGAPP Symposium on Applied Computing*, 2016.

[15] J. P. Leal, "Using proximity to compute semantic relatedness in rdf graphs," *Computer Science and Information Systems*, vol. 10, no. 4, pp. 1727–1746, 2013.

[16] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender systems handbook*. Springer, 2011, pp. 257–297.

[17] S. Zwicklbauer, C. Seifert, and M. Granitzer, "From general to specialized domain: Analyzing three crucial problems of biomedical entity disambiguation," in *Database and Expert Systems Applications*. Springer, 2015, pp. 76–93.

[18] G. Pirrò, "Explaining and suggesting relatedness in knowledge graphs," in *The Semantic Web–ISWC 2015*. Springer, 2015, pp. 622–639.

[19] K. Anyanwu, A. Maduko, and A. Sheth, "Semrank: ranking complex relationship search results on the semantic web," in *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005, pp. 117–127.

[20] B. Aleman-Meza, C. Halaschek-Weiner, C. Ramakrishnan, A. P. Sheth *et al.*, "Ranking complex relationships on the semantic web," *Internet Computing, IEEE*, vol. 9, no. 3, pp. 37–44, 2005.