Augmenting LOD-Based Recommender Systems Using Graph Centrality Measures

Bart van Rossum and Flavius Frasincar$^{[0000-0002-8031-758X]}$

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands
{vanrossum, frasincar}@ese.eur.nl

Abstract. In this paper we investigate the incorporation of graph-based features into LOD path-based recommender systems, an approach that so far has received little attention. More specifically, we propose two normalisation procedures that adjust user-item path counts by the degree centrality of the nodes connecting them. Evaluation on the MovieLens 1M dataset shows that the linear normalisation approach yields a significant increase in recommendation accuracy as compared to the default case, especially in settings where the most popular movies are omitted. These results serve as a fruitful base for further incorporation of graph measures into recommender systems, and might help in establishing the recommendation diversity that has recently gained much attention.

Keywords: Top-N recommendations · Linked Open Data · information network schema · random forest

1 Introduction

Recommender systems (RS) are algorithms aimed at presenting the user with items of which it is most likely that (s)he will enjoy them [11]. Over the past years, the rise of the Semantic Web has enabled a boost in their performance [12]. The structured knowledge representations available in the Linked Open Data (LOD) cloud allows for the construction of hybrid recommendation systems that leverage both user preferences as well as information on product features [4]. Combining the best of both worlds, these hybrid systems have been shown to consistently outperform other state-of-the-art recommendation systems [13].

One of the most promising ways of incorporating data from the LOD is by constructing a knowledge graph linking all users, items, and relevant semantic information, and including path-based features in hybrid recommendation systems. In other words, the number of connections in a knowledge graph between a user and an item determines the likelihood of this item being recommended to the user, an idea with strong intuitive appeal. This approach has been successfully employed in leveraging semantic information from the popular LOD dataset DBpedia [3] in the domain of books, films, and music [14, 17].

While these semantic path-based features have greatly increased the accuracy of recommender systems, they do not nearly capture all relevant information embedded in a knowledge graph. One can imagine that the structure of this graph contains information that is potentially relevant to recommendation systems. The clustering of items and users or the relative importance of items within this graph, for example, are definitely factors of interest when recommending items to a user. To this day, however, this dimension of the knowledge graph has remained largely unexplored. While some authors have proposed including graph-based features in their RS, unifying graph-based features with the successful path-based ones remains an open challenge [13, 16].

In this paper we propose a simple, yet effective way of unifying the previously mentioned two types of features. By normalising path-based features based on the centrality of the nodes along each path, we aim to improve the performance of hybrid recommendation systems. The rationale behind this proposal is as follows: paths between a user and an item along nodes with a high popularity in the network are unlikely to capture the unique preferences of this user, and should therefore contribute less to the score of this item. A user liking "The Matrix", for example, tells us more about the popularity of this movie then about the particular user. Since centrality measures offer a diverse set of ways of measuring the popularity of a node in a network, these are very much suited in performing this correction.

An additional benefit of this approach is its potential to diversify the set of items recommended to a user. The goal of diversification has been deemed more and more important recently, yet is usually in conflict with the goal of high recommendation accuracy [12, 15]. The proposed use of normalised path-based features, however, can contribute to higher diversity whilst retaining or increasing accuracy, since items with a low centrality measure are more likely to be both recommended and unknown to the user.

This paper is structured as follows. Section 2 discusses related work, while Sect. 3 elaborates on the proposed methodology. Section 4 introduces the evaluation procedure and the obtained results, while Sect. 5 gives our conclusion and future work.

2 Related Work

The use of semantic path-based features in recommender systems is not novel, and has been successfully applied by the authors of [14]. In the book, film, and music domain, they leveraged data from `DBpedia` to train an ontology-based recommender system, enabling them to compare various learning-to-rank algorithms. This is where their approach differs from ours: since we are only interested in the added value of graph-based normalisation, we only employ random forests to train our recommender system. Moreover, since we are merely interested in the normalisation procedure, we limit the number of possible paths by defining a network schema in which only certain path types are allowed.

In this respect, our approach closely resembles the one used in [17], which pioneered the use of network schemas in path-based recommender systems. Moreover, the evaluation strategy employed by these authors is identical to the one we use. Whereas they also explore the benefits of feature selection, this aspect remains unexplored in this paper, due to our focus on normalisation.

An alternative approach that already captures more information about the graph structure is proposed by the authors of [16]. They propose `entity2rec`, a feature-construction algorithm that learns property-specific vector representations of users and items by simulating random walks through the knowledge graph. Subsequently they construct user-item relatedness scores based on these representations, feeding this into a recommender system. A downside of this approach is the need to tune several hyperparameters, and the extent to which it employs information from the graph is still very limited.

A second paper that utilizes graph-based features is [13], the authors of which construct item features that are directly based on graph centrality measures. Together with the aforementioned path-based measures, these are loaded into several types of recommender systems. Their results show that directly using centrality measures as input variables has little added value, since they mainly serve as an additional popularity measure and are not incorporated into the other features yet. Our approach differs from the one used in [13] as we aim to include graph centrality measures directly into the used path measures, thus exploiting the interaction effects between them.

In this paper we aim to use centrality measures by discounting popular node counts on the considered paths, thereby bridging the gap between path-based and graph-based features. As we exploit the structure of the information network to a further extent than has been done before, we hope this information will prove relevant to the recommender system and increase recommendation accuracy. Since it is unknown which type of normalisation is most suited for this purpose, we evaluate multiple procedures in various settings. In short, the main contribution of this paper is to build on the current semantic path-based recommender systems literature by controlling path-based features for graph-based measures. Similar to the aforementioned papers, we evaluate the effectiveness of this procedure on the MovieLens 1M dataset.

3 Methodology

In order to construct path-based features, we require a way of modeling the knowledge of the movie domain and user preferences. This is done in an information network, described in the following. Next, the definition of a meta path and the normalisation procedure are given, as well as the algorithm used to train the ranking function.

3.1 Information Network

An information network is defined as an undirected graph $G = (V, A)$, where V represents the set of vertices, or nodes, in the network, and A the set of edges between them. In our case, $V = I \cup U \cup E$, where I equals the set of items (movies), U the set of users, and E the set of entities. The latter set consists of all actors, directors, subjects, and entities otherwise linked to the movies. G is undirected as each edge $a \in A$ is assumed to be symmetric, reflecting the fact that we are mainly interested in the association between two nodes rather than any causal links between them. Moreover, we define the type function $t : A \to T$, representing that each edge a is of type $t(a) \in T$. Table 1 displays the elements of T, indicating that all edges link either items and users, or items and entities.

Table 1. Set of edge-types T

Connected sets	Type	k
I & U	`likes`	1
I & E	`starring`	2
I & E	`director`	3
I & E	`cinematography`	4
I & E	`producer`	5
I & E	`editing`	6
I & E	`writer`	7
I & E	`musicComposer`	8
I & E	`narrator`	9
I & E	`basedOn`	10
I & E	`subject`	11

In order to get an impression of what such a graph looks like, Fig. 1 shows a small example information network. Consisting of only 11 nodes and containing only the first three types of edges (`likes`, `starring`, and `director`), it is a heavily simplified version of the actual network we will work with. Nonetheless, it clearly displays the way in which the sets U, I, and E are linked. Moreover, it provides some intuition on how paths between items offer an indication of their similarity. We can infer Inception and The Prestige to be similar items, for example, since they are connected by multiple paths of different types.

3.2 Meta Paths and Path-based Features

In order to assess the degree of similarity between items, we introduce the notion of a meta path. A meta path specifies the type of edges that connect nodes of a certain type. Using the notation introduced above, each meta path P of length l can be defined as:

$$P = V_1 \xrightarrow{t_1} V_2 \xrightarrow{t_2} \dots \xrightarrow{t_{l-1}} V_l \tag{1}$$

where $V_i \in V,\ i = 1, \dots, l$ and $t_i \in T,\ i = 1, \dots, l-1$. In other words, a meta path specifies which type of semantic link exists between two types of nodes. Note that there may exist multiple instances of each meta path in our information network G. Every instance p of P can be defined as a sequence of

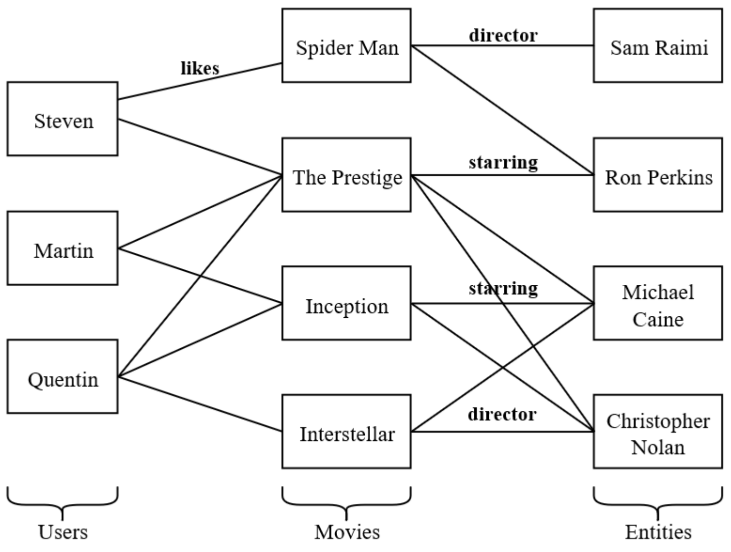

Fig. 1. Example information network

nodes and edges $(v_1, a_1, v_2, a_2, \dots, a_{l-1}, v_l)$ where v_i is of type $V_i,\ i = 1, \dots, l$ and $t(a_i) = t_i,\ i = 1, \dots, l-1$.

In the following, we focus on symmetric meta paths of length $l = 3$ between items, i.e., all meta paths of the type $P = V_1 \xrightarrow{t} V_2 \xrightarrow{t} V_3$ for which $V_1, V_3 \in I$. These paths link all movies that have something in common, e.g., movies that are directed by the same director when we set $V_2 \in E$ and $t =$ `director`. Note that this allows for $|T|$ different meta paths. In the following, we will refer to $P(k),\ k = 1, \dots, |T|$, as the meta path with t equal to the k-th element of T (as displayed in Table 1).

For each user-item pair $(u, i), u \in U, i \in I$, we can now construct a feature vector $\mathbf{x}_{u,i} \in \mathbb{R}^{|T|}$ as follows:

$$\mathbf{x}_{u,i}(k) = \sum_{j \in I_u^+} \#path_{i,j}(k), \quad k = 1, \dots, |T| \tag{2}$$

where I_u^+ is the set of items liked by user u and $\#path_{i,j}(k)$ indicates the number of instances of $P(k)$ linking i and j. Intuitively, when elements in this vector are of larger magnitude, we expect the user u to be more interested in item i, as there exist more meta path instances linking it to items that he has previously liked.

Table 2. Example of path-finding procedure

i	k	Path
Interstellar	1	Interstellar $\xrightarrow{\texttt{likedBy}}$ Quentin $\xrightarrow{\texttt{likes}}$ The Prestige
Interstellar	1	Interstellar $\xrightarrow{\texttt{likedBy}}$ Quentin $\xrightarrow{\texttt{likes}}$ Inception
Interstellar	2	Interstellar $\xrightarrow{\texttt{starring}}$ Michael Caine $\xrightarrow{\texttt{starringIn}}$ The Prestige
Interstellar	3	Interstellar $\xrightarrow{\texttt{directedBy}}$ Christopher Nolan $\xrightarrow{\texttt{directing}}$ The Prestige
Spider Man	1	Spider Man $\xrightarrow{\texttt{likedBy}}$ Steven $\xrightarrow{\texttt{likes}}$ The Prestige
Spider Man	2	Spider Man $\xrightarrow{\texttt{starring}}$ Ron Perkins $\xrightarrow{\texttt{starringIn}}$ The Prestige

To further clarify this procedure, we now demonstrate the feature construction for one of the users of Fig. 1. More specifically, we evaluate $\mathbf{x}_{u,i}$ for $u =$ Martin and $i \in \{$Interstellar, Spider Man$\}$. Since $I^+_{\text{Martin}} = \{$The Prestige, Inception$\}$, this boils down to finding all the paths displayed in Table 2. Note that while the edge types are in fact symmetric, we have used asymmetric naming here for the sake of readability. Summing paths over each path type yields the following user-item features: $\mathbf{x}_{\text{Martin,Interstellar}} = \langle 2, 1, 1\rangle$ and $\mathbf{x}_{\text{Martin,Spider Man}} = \langle 1, 1, 0\rangle$. Based on these two features only, one would expect Martin to prefer Interstellar over Spider Man.

3.3 Centrality Measure and Normalisation Approaches

The features described in the previous section can be considered as the default case, path-based features which are not yet normalised in any way. In the following we describe a procedure that normalises the path count using the centrality of the nodes along the path. To this extent, we compute the degree centrality C_D of all nodes. As the normalisation is done separately for each meta path, i.e., for each element of the feature vector, we define this measure with respect to networks where only instances of a certain meta path are allowed. In other words, in computing this measure we only consider edges of one specific type, depending on the meta path of interest.

The degree centrality measures the number of nodes in the network a particular node is connected to. For any node $v \in V$ and meta path P it is defined as follows:

$$C_D(v, P) = deg(v, P) \tag{3}$$

where $deg(v, P)$ is the degree of node v in a network where only meta paths of type P are allowed. $C_D(i, P(1))$ with $i \in I$, for example, would measure the number of users that have liked item i, whereas $C_D(i, P(2))$ would reflect the number of actors starring in i.

Having introduced this centrality measure, we can define the actual normalisation procedure. In brief, every path between a user and an item is now inversely weighted by the relative degree centrality of the nodes along that path. We propose two slightly different procedures, both of which will be evaluated.

Firstly, one can consider a so-called 'compounding' procedure, which means that the normalised user-item feature $\mathbf{x}^c_{u,i}$ is now defined as follows:

$$\mathbf{x}_{u,i}(k) = \sum_{j \in I_u^+} \#path^*_{i,j}(k), \quad k = 1, \ldots, |T| \tag{4}$$

where

$$\#path^*_{i,j}(k) = \sum_{p \in P(k): i,s,j \in p} \frac{\overline{C}_D(i,p)}{C_D(i,p)} \times \frac{\overline{C}_D(s,p)}{C_D(s,p)} \times \frac{\overline{C}_D(j,p)}{C_D(j,p))} \tag{5}$$

in which $\overline{C}_D(i, P)$ is the average of C_D taken over all nodes of the same type as i. If $i \in I$, for example, this average is taken w.r.t. all movies in the network (not only on the considered paths, but only for the considered edges of type k). Note that i and j are necessarily of the same type, while s can be both from U or from E. From (6) it becomes clear that paths along nodes with relatively high centrality measures will now contribute less to the feature vector than in (2), since they are multiplied with a weighting factor that is below one.

Secondly, one can consider a normalisation procedure that still discounts popular nodes, yet is less sensitive to outliers. We will refer to it as the 'mean' procedure, which discounts using the arithmetic mean of the relative centrality measures along a path:

$$\#path^*_{i,j}(k) = \sum_{p \in P(k): i,s,j \in p} \frac{1}{3} \left(\frac{\overline{C}_D(i,p)}{C_D(i,p)} + \frac{\overline{C}_D(s,p)}{C_D(s,p)} + \frac{\overline{C}_D(j,p)}{C_D(j,p)} \right). \tag{6}$$

As before, we will demonstrate these procedures by considering their effect on two specific meta paths from Table 2 where $k = 2$, i.e., the paths Interstellar $\xrightarrow{\texttt{starring}}$ Michael Caine $\xrightarrow{\texttt{starringIn}}$ The Prestige and Spider Man $\xrightarrow{\texttt{starring}}$ Ron Perkins $\xrightarrow{\texttt{starringIn}}$ The Prestige. Table 3 displays the results of applying the two normalisation procedures to these specific paths, where the node degrees are computed based on Fig. 1 and the average node degrees are assumed to be 2 for all node types (for the considered edges of type k).

Table 3. Example of normalisation procedures

Node	C_D	$\overline{C}_D/C_D$	Node	C_D	$\overline{C}_D/C_D$
Interstellar	1	2	Spider Man	1	2
Michael Caine	3	2/3	Ron Perkins	2	1
The Prestige	2	1	The Prestige	2	1
Compound: $2 \times 2/3 \times 1 = 4/3$			Compound: $2 \times 1 \times 1 = 2$		
Mean: $1/3 \times (2 + 2/3 + 1) = 11/9$			Mean: $1/3 \times (2 + 1 + 1) = 4/3$		

Whereas in the example in Sect. 3.2 all paths contributed equally to the total, we see that the discounting creates variance in the path weights. First of all, both paths get a score above one due to the fact that (in this example) they star fewer actors than an average movie. Second, we find that the path via Ron Perkins receives a higher value than the one via the more popular actor Michael Caine. This clearly shows the rationale behind our procedure: when a user likes less-known entities, this gives us a clearer indication of his true preferences then when he likes more popular entities. Third, this example confirms our statement that the compounding procedure is more sensitive than the mean one: in both cases, it yields scores which in magnitude deviate most from one.

3.4 Training the Ranking Function

The aim of the recommender system is to recommend the most relevant items to a user, based on the knowledge present in G. This is done using a scoring function $f : \mathbb{R}^{|T|} \rightarrow \mathbb{R}$ which converts each user-item feature vector into a score indicating the expected relevance of the item to the user. More formally, for each item i rated $r_{u,i}$ by user u, we want that $f(\mathbf{x}_{u,i}) \approx r_{u,i}$. Note that we define $r_{u,i}$ to be 1 if u has liked i, and 0 otherwise. The function f will be learnt by training it on a training set $TS = \bigcup_{u \in U} \{\langle \mathbf{x}_{u,i}, r_{u,i} \rangle : i \in I_u^+ \cup I_u^-\}$, where once again I_u^+ is the set of items liked by user u and I_u^- is the set of items disliked by u.

Given such a function f, we can rank items based on their score and recommend the top-N to the user. As random forests have showed to perform well at such ranking tasks [6], this is what we will use to train f. Random forests are an ensemble technique that can be used for classification and regression, and aggregate the output of multiple decision trees to a single output variable [5]. Following [17], the random forest we will employ consists of 100 Classification and Regression Trees (CART), and each tree is trained on a subset of w randomly selected variables to prevent overfitting. In this setting we fix w to be $\sqrt{|T|}$.

4 Evaluation

The following describes the results evaluation protocol, starting with introducing the data used and the training of the random forest. Subsequently, the performance of the recommender system is analysed using an accuracy measure. Finally, the results are linked to the existing literature on accuracy versus diversity.

4.1 Data

The recommender system will be trained and evaluated using the `MovieLens` 1M dataset, containing 1,000,209 1-5-star ratings of 3,883 movies by 6,040 users [10]. The authors of [14] have attempted a mapping of each movie in the 1M dataset to a unique `DBpedia` URI, which we use here as well. This leaves us with 3,196 movies that together will form the set I. The set of users U is constructed by retaining only those users that have positively rated at least 15 items in

I, to ensure there is sufficient information on each remaining user. We define a positive rating to be a 5-star one in order to remove any possible ambiguity w.r.t. liked movies. This way, a total of 3,854 users and 193,255 ratings remains. Using the aforementioned mapping, we query `DBpedia` for all RDF triples linking the movies to other entities. These can be triples from the `Film` domain, in which all information regarding a specific movie is stored, as well `dcterms:subject` triples, linking the movies to the Wikipedia categories they are part of. This way, the set of entities E and all the edges between I and E are retrieved. We refer the reader to [17] for more information on this querying procedure. A total of 13,649 unique entities is obtained.

Given these sets and the relations between them, we are able to construct the information network G. The implementation thereof is done in `Neo4j`[1], a graph database platform that allows for fast retrieval of meta paths between two items.

4.2 Evaluation Protocol

The procedure described above leaves us with a set of users that all have at least 15 positive ratings. For each user, we select 10 liked items that construct the test set for this user. For each positively rated item in this test set, we add 100 randomly selected items which the user did not like. Note that these can also be items that the user has not rated at all, for which we assume he does not like them. This leaves us with 10 batches of 101 items for each individual user.

The training set will contain the remaining positively rated movies of each user, up to a maximum of m items. We will evaluate two sizes of training profiles, $m = 5$ and $m = 50$. The training set per user is augmented with $2m$ randomly selected irrelevant items. The aggregate training set over all users is then used to train the random forest, as described in Sect. 3.4.

After the random forest has been trained, its accuracy is evaluated on the test set. The items within each batch are ranked based on their predictions from the ranking function. The top-N items are then recommended to the user, after which the recall@N measure is computed. This is defined as follows:

$$recall@N = \frac{\#relevant\ items\ recommended}{\#relevant\ items\ in\ batch}. \tag{7}$$

Since we assume there is only one relevant item per batch, this simplifies to:

$$recall@N = \mathbf{1}_{relevant\ item\ in\ top-N} \tag{8}$$

which can be seen as the probability of recommending a relevant item to a user when displaying the top-N results only. Note that this merely provides us with a lower bound on the actual accuracy of the system, as not all unliked items are indeed irrelevant to the user. This measure will be evaluated for multiple values of N, and subsequently averaged over the batches of all users.

[1] `www.neo4j.com`

4.3 Results

Table 4 shows the recall@N measures for various levels of N and two profile sizes, i.e., two sizes of m. As expected, the accuracy is increasing in N, which follows directly from (8): as the set of top items is larger, the chances of finding a relevant item increase. The marginal effect is decreasing in N, however, indicating that the relevant item is unlikely to be positioned in the tail of the ranking. Comparing results for $m = 5$ with those of $m = 50$, we find that a bigger training set increases the accuracy of the recommender system. This is expected, since the random forest is better able to extract relevant patterns when trained on a bigger sample. Finally, we are able to compare the standard path count procedure with the normalised versions. The results indicate that the compounded measure never outperforms the standard one, while the mean-normalised approach scores similarly or higher than the standard procedure. This higher score is especially present in the smaller profile size.

Table 4. Recall@N for all movies

	$m = 5$			$m = 50$		
N	Standard	Compound	Mean	Standard	Compound	Mean
5	0.483	0.223	0.488	0.534	0.258	0.532
10	0.692	0.336	0.701	0.718	0.391	0.707
15	0.807	0.422	0.813	0.799	0.511	0.809
20	0.872	0.480	0.882	0.876	0.612	0.872
25	0.917	0.546	0.922	0.918	0.697	0.915

Whereas Table 4 displays the results based on the full set of items, it is also interesting to look at a setting in which the most popular items are out of consideration. These popular items are responsible for a relatively large share of the positive ratings, and therefore liked by most users. Omitting those from the sample tells us how well our recommender system performs at extracting the preferences that are actually unique to users. In order to filter the sample, we keep removing the most popular item from the set, until the removed items jointly account for 33% of the ratings, hereby following the approach proposed by [7]. After removing these items, the evaluation procedure is repeated.

Table 5 displays the results for the long tail only. A striking observation is the generally lower accuracy, which can be attributed to the omission of popular items. These items are liked by many users and therefore boost the accuracy of a recommender system. Only the results for the compounded path counts seem to benefit from this omission, most likely because the negative effect of the compounding procedure is diminished in this setting. The general patterns of Table 4 are observable here as well, except for the fact that the mean-weighting procedure now clearly outperforms the standard one. The potential explanation for the compounded results might hold here as well: the possible negative effect on accuracy due to recommending less popular items is diminished in a setting with fewer popular items. In contrast, the positive effect due to the normalisation procedure seems to dominate here, indicating that these normalised path count measures are indeed more meaningful.

Table 5. Recall@N for long tail only

	$m = 5$			$m = 50$		
N	Standard	Compound	Mean	Standard	Compound	Mean
5	0.387	0.257	0.436	0.449	0.329	0.510
10	0.619	0.390	0.647	0.667	0.505	0.728
15	0.766	0.499	0.787	0.795	0.638	0.848
20	0.857	0.595	0.876	0.860	0.741	0.915
25	0.911	0.675	0.921	0.915	0.819	0.953

4.4 Diversity and Popular Items

It is well-known that recommendation systems maximising accuracy usually recommend the most popular items, even though this is not always in accordance with user utility [18]. Rather than simply receiving a set of accurate recommendations, the user also wishes to be surprised by diverse and novel set of recommendations. Popular items alone rarely achieve this goal, yet shifting to less-popular items also implies a reduction in accuracy. All in all, this yields a trade-off between diversity and accuracy [19], the resolution to which has until now remained unclear.

Given the previously mentioned background, the main result from Sect. 4.3 is especially surprising. Despite the fact that our proposed procedure actively discounts popular items, we find that accuracy does not suffer, but even increases in certain settings. Moreover, the authors of [1] have shown that recommending fewer popular items increases the diversity of the recommendation set. This implies that graph-based normalisation alleviates the dilemma by maintaining or even increasing recommendation accuracy, whilst yielding an increase in diversity of the recommended items.

5 Conclusion

Path-based recommender systems usually rely on information networks, the structure of which contains a lot of knowledge relevant to the recommendations. Despite its potential, this graph structure has so far received little attention in literature. This paper proposes a novel method of unifying the path-based and graph-based features extracted from an information network. More specifically, we implement and test several normalisation procedures that discount path counts between users and items by the degree centrality measures of the nodes along this path. When evaluated on the MovieLens 1M dataset, the linear weighting procedure has been shown to significantly outperform the default approach. This effect is especially strong in the long tail of the dataset, where the normalisation procedure is not punished for recommending less popular movies. Next to a higher recommendation accuracy, this approach also yields the benefit of presenting more novel items to the user and increased diversity in the recommendation set.

Despite careful considerations, several limitations to this paper remain. First of all, since only the movie domain has been investigated, it remains an open question whether this method performs equally well in, e.g., the book and music domains. Second, there exist multiple other centrality measures and normalisation procedures, the performance of which might surpass that of the current set-up. Both of these provide relevant suggestions for future research.

Nonetheless, the implementation of graph-based measures in the context of recommender systems has proven to be fruitful, and constitutes a solid basis for future work. Potential benefit is to be gained especially in the area of diversity, which is recently gaining extra interest. Employing the structure of the information graph might be a promising way of accentuating less-known items and of recommending items that are as different from each other as possible, thereby increasing the diversity among recommendations. Incorporating diversity measures, as proposed by for example [12] and [15], into the current setting would give further insight into the feasibility of the proposed approach.

Next to the suggestions mentioned above, it is of much interest to investigate other methods of incorporating structural information on the information network into a RS. A user clustering approach as proposed by [2] and [9], for example, might benefit hugely from graph clustering algorithms (see [8] for an overview). This is especially likely to prove fruitful when information graphs containing both user- as well as content-based information like ours are considered, as compared to collaborative filtering approaches tried so far.

References

1. Adomavicius, G., Kwon, Y.: Improving aggregate recommendation diversity using ranking-based techniques. IEEE Transactions on Knowledge and Data Engineering **24**(5), 896–911 (2012)
2. Altingovde, I.S., Subakan, Ö.N., Ulusoy, Ö.: Cluster searching strategies for collaborative recommendation systems. Information Processing & Management **49**(3), 688–697 (2013)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: The Semantic Web, pp. 722–735. Springer, Berlin, Heidelberg (2007)
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. International Journal on Semantic Web and Information Systems **5**(2), 1–22 (2009)
5. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
6. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. In: Proceedings of the learning to rank challenge. pp. 1–24 (2011)

7. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the fourth ACM conference on Recommender systems. pp. 39–46. ACM (2010)
8. Emmons, S., Kobourov, S., Gallant, M., Borner, K.: Analysis of network clustering algorithms and cluster quality metrics at scale. PloS One **11**(7) (2016)
9. Gong, S.: A collaborative filtering recommendation algorithm based on user clustering and item clustering. JSW **5**(7), 745–752 (2010)
10. Harper, F.M., Konstan, J.A.: The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems **5**(4) (2015), article 19
11. Jannach, D., Resnick, P., Tuzhilin, A., Zanker, M.: Recommender systems-beyond matrix completion. Communications of the ACM **59**(11), 94–102 (2016)
12. Lops, P., Gemmis, M.D., Semeraro, G.: Content-based recommender systems: State of the art and trends. In: Recommender Systems Handbook, pp. 73–105. Springer, Boston, MA (2011)
13. Musto, C., Lops, P., de Gemmis, M., Semeraro, G.: Semantics-aware recommender systems exploiting linked open data and graph-based features. Knowledge-Based Systems **136**, 1–14 (2017)
14. Noia, T.D., Ostuni, V.C., Tomeo, P., Sciascio, E.D.: SPrank: Semantic path-based ranking for top-n recommendations using linked open data. ACM Transactions on Intelligent Systems and Technology **8**(1), 9:1–9:34 (2016)
15. Noia, T.D., Rosati, J., Tomeo, P., Sciascio, E.D.: Adaptive multi-attribute diversity for recommender systems. Information Sciences **382**, 234–253 (2017)
16. Palumbo, E., Rizzo, G., Troncy, R.: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In: Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys 2017). pp. 32–36. ACM (2017)
17. Wever, T., Frasincar, F.: A linked open data schema-driven approach for Top-N recommendations. In: Proceedings of the Symposium on Applied Computing (SAC 2017). pp. 656–663. ACM (2017)
18. Zhang, M., Hurley, N.: Avoiding monotony: improving the diversity of recommendation lists. In: Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys 2008). pp. 123–130. ACM (2008)
19. Zhou, T., Kuscsik, Z., Liu, J.G., Medo, M., Wakeling, J.R., Zhang, Y.C.: Solving the apparent diversity-accuracy dilemma of recommender systems. Proceedings of the National Academy of Sciences **107**(10), 4511–4515 (2010)