Augmenting LOD-Based Recommender Systems Using Graph Centrality Measures

Bart van Rossum
vanrossum@ese.eur.nl

Flavius Frasincar
frasincar@ese.eur.nl

Erasmus University Rotterdam
the Netherlands

Contents

Motivation

- Paradox of Choice: users are overloaded with items to choose from (Schwartz, 2004)
- **Recommender systems**: present users only with relevant items
  - Content-based: recommend similar items to what users have previously liked
  - Collaborative filtering: recommended items previously liked by similar users
  - **Hybrid**: mix of content-based and collaborative filtering (state-of-the-art results) [our focus here]
- **LOD-based recommenders**: a specific class of hybrid recommenders that represent item information using Linked Open Data (LOD)
  - Build a knowledge graph linking all users, items, and relevant semantic information from LOD
  - Use path-based features (paths are from users to items)
  - The number of paths between a user and an item determine the likelihood that this item will be recommended to the user

Motivation

- Current LOD-based recommenders do not exploit all relevant graph information
- In previous work graph-based information has been used as additional item features

Main Ideas

- Unify graph-based features with the successful path-based ones
- Normalize path-based features based on the centrality of the nodes along each path
  - 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
  - We measure the popularity of nodes by graph centrality measures
- Potential benefits:
  - Accurate (specific) recommendations
  - Diverse recommendations

Related Work

- LOD-based recommender systems have been previously proposed and various learning-to-rank algorithms have been compared on different datasets (Noia et al., 2016)
  - As we focus on graph-based normalization, we employ only **random forests** (due to their previous good performance) and limit the number of possible paths by using a **network schema** (only certain paths are allowed)
- Our approach resembles the most the one from (Wever and Frasincar, 2017), which pioneered the use of network schemas in path-based recommender systems
  - While we employ the same evaluation strategy, our focus is on **feature normalization** instead of **feature selection**
- Graph centrality measures have been previously used as additional item features (Musto et al., 2017)
  - We aim to include **graph centrality** measures directly into the used path measures, thus exploiting the interaction effects between these

Methodology

- As in related work the evaluation is done on the `MovieLens` 1M dataset (movie domain)
- The methodology comprises four parts:
  1. **Information Network**: models the knowledge about the movie domain and user preferences
  2. **Meta Path**: represents the schema of the information network paths
  3. **Path Normalization**: normalizes the path features using graph centrality measures
  4. **Learning Algorithm**: trains a model for ranking items

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, $V = U \cup I \cup E$, where
  - U is the set of users
  - I is the set of items (movies)
  - E is the set of entities (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 (we are mainly interested in the association between two nodes rather than any causal links between them)
- We define the type function $t : A \rightarrow T$ representing that each edge a is of type $t(a) \in T$

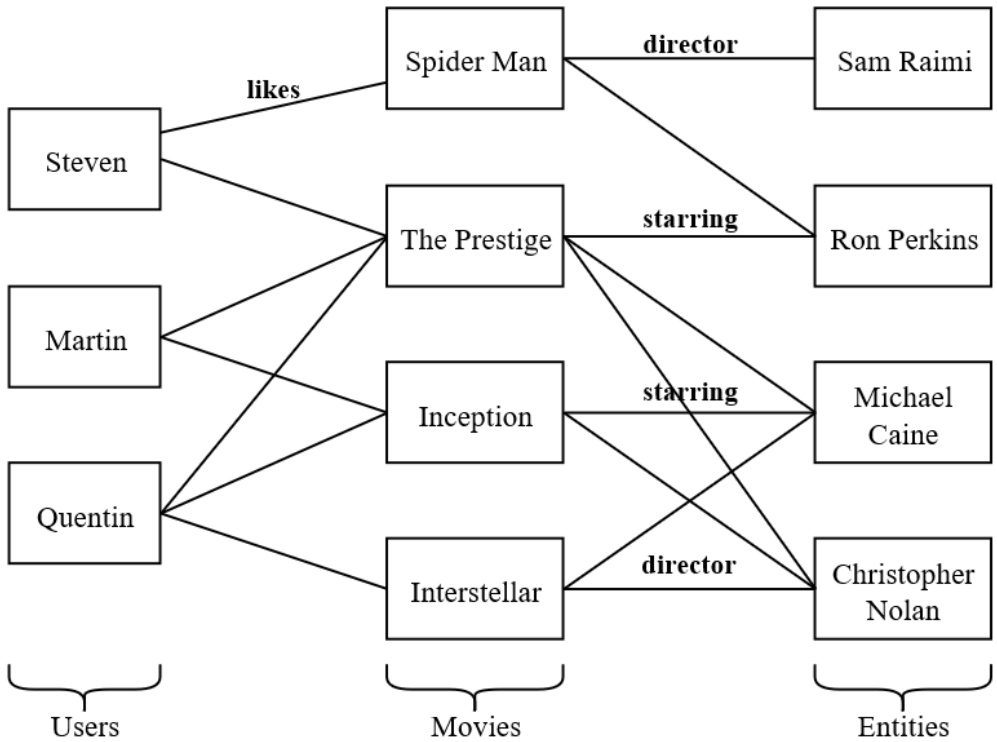
Information Network

The 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

Information Network

A small example of an information network:



- Uses only the first three types of edges (`likes`, `starring`, and `director`)
- Displays the way in which the sets U (users), I (movies), and E (directors and actors) are linked

Meta Path

- A **meta path** P of length l is be defined as:

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

  where $V_i \subseteq V,\ i = 1, \ldots, l$ and $t_i \in T,\ i = 1, \ldots, l-1$
- A **meta path instance** p of P can be defined as a sequence of nodes and edges $(v_1, a_1, v_2, a_2, \ldots, a_{l-1}, v_l)$ where $v_i \in V_i,\ i = 1, \ldots, l$ and $t(a_i) = t_i,\ i = 1, \ldots, l-1$
- 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 $V_1 = V_3 = I$
- We refer to $P(k),\ k = 1, \ldots, |T|$, as the meta path with t equal to the k-th element of T
- For example $P(3)$ represents the meta path $I \xrightarrow{director} Director \xrightarrow{director} I$

Meta Path

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

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

where

- I_u^+ is the set of items liked by user u
- $\#path_{i,j}(k)$ indicates the number of instances of $P(k)$ linking i and j

Intuition

Large values in the vector $\mathbf{x}_{u,i}$ mean many meta path instances between the user u and the item i, so we expect the user u to be interested in item i

Meta Path

A small example of the 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

- We evaluate $\mathbf{x}_{u,i}$ for $u =$ Martin and $i \in \{$Interstellar, Spider Man$\}$
- $I^{+}_{\text{Martin}} = \{$The Prestige, Inception$\}$
- $\mathbf{x}_{\text{Martin,Interstellar}} = \langle 2, 1, 1 \rangle$ and $\mathbf{x}_{\text{Martin,Spider Man}} = \langle 1, 1, 0 \rangle$ (Martin seems to prefer Interstellar over Spider Man)

Path Normalization

- The **centrality degree** of a node $v \in V$ with respect to meta path P is defined as:

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

where $deg(v, P)$ is the degree of node v in a network where only meta paths of type P are allowed

- For example $C_D(i, P(1))$ with $i \in I$ would measure the number of users that have liked item i and $C_D(i, P(2))$ would reflect the number of actors starring in i

Idea

The path between a user and an item should be inversely weighted by the relative centrality degree of the nodes along that path

- We only consider edges of one specific type, depending on the meta path of interest

Path Normalization

- We propose two normalizations:
  - **Compound**: based on products
  - **Mean**: based on averages (less sensitive to outliers)
- For the compound normalization the user-item feature $\mathbf{x}^c_{u,i}$ is defined as:

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

where

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

in which $\overline{C}_D(i, P(k))$ is the average of C_D taken over all nodes in the network of the same type as i with respect to meta path $P(k)$

Path Normalization

- For the mean normalization we have:

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

- Each path will contribute to the count with a value possibly bigger or smaller than 1
- For example, let us normalize two paths using the `starring` edge ($< i, s, p > \in P(2)$):
  - Interstellar $\xrightarrow{\texttt{starring}}$ Michael Caine $\xrightarrow{\texttt{starringIn}}$ The Prestige
  - Spider Man $\xrightarrow{\texttt{starring}}$ Ron Perkins $\xrightarrow{\texttt{starringIn}}$ The Prestige
- Let us also assume, in this example, that the average node degrees are 2 for all node types (for the considered edges of type 2)

Path Normalization

The results of the two normalizations are:

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: $1/3 \times (2 + 1 + 1) = 4/3$		

- Both paths get a score above 1 due to the fact that (in this example) they star fewer actors than an average movie
- The path via Ron Perkins receives a higher value than the one via the more popular actor Michael Caine
- The compounding procedure is more sensitive (to popularity) than the mean one: in both cases, it yields scores which in magnitude deviate the most from 1

Learning Algorithm

- We need to learn a scoring function $f : \mathbb{R}^{|T|} \rightarrow [0, 1]$ which converts each user-item feature vector into a score indicating the expected relevance of the item to the user
- For each item i rated $r_{u,i}$ by user u, we want that $f(\mathbf{x}_{u,i}) \approx r_{u,i}$, where $r_{u,i}$ is 1 if u likes i, and 0 otherwise
- We learn the function f on the training set $TS = \bigcup_{u \in U} \{\langle \mathbf{x}_{u,i}, r_{u,i} \rangle : i \in I_u^+ \cup I_u^-\}$, where
  - I_u^+ is the set of items liked by user u
  - I_u^- is the set of items disliked by u
- After we learn the function f we rank items from the test set based on their score and recommend the top-N to the user
- We use Random Forest as the algorithm due to its good performance on the learning-to-rank task (Chapelle and Chang, 2011)
  - 100 Classification and Regression Trees (CART) an in (Wever and Frasincar, 2017)
  - $w = \sqrt{|T|}$ randomly selected variables per tree node an in (Wever and Frasincar, 2017)

Evaluation Data

- We use `MovieLens` 1M dataset, containing 1,000,209 1-5-star ratings of 3,883 movies by 6,040 users
- We map each movie in the 1M dataset to a unique `DBpedia` URI and obtain 3,196 movies (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
- We obtain a total of 3,854 users (set U) and 193,255 ratings
- A total of 13,649 unique entities (set E) and corresponding edges (set A) are obtained from `DBpedia` that are linked to our movies using the procedure from (Wever and Frasincar, 2017)

Evaluation Protocol

Test Set

- For each user, we select 10 liked items that are used to 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
- We have 10 batches of 101 items for each individual user

Training Set

- The training set contains the remaining positively rated movies of each user, up to a maximum of m item
- The training set per user is augmented with $2m$ randomly selected irrelevant items
- We evaluate two sizes of training profiles $m = 5$ and $m = 50$

Evaluation Protocol

- We learn the model using the Random Forest on the training set
- We evaluate the model using 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 as

$$recall@N = \frac{\#relevant\ items\ recommended}{\#relevant\ items\ in\ batch}. \tag{5}$$

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

$$recall@N = \mathbf{1}_{relevant\ \ item\ \ in\ \ top-N} \tag{6}$$

  - Note that this is a **lower bound** on the actual accuracy of the system, as not all unliked items are irrelevant to the user
  - We average recall@N over the 10 batches per user and then for all users

Evaluation Results

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

- The accuracy is increasing in N, as the set of top items is larger, the chances of finding a relevant item increase
- The marginal effect is decreasing in N, indicating that the relevant item is unlikely to be positioned in the tail of the ranking
- A bigger training set ($m = 50$) increases the accuracy of the recommender system, as the random forest is better able to extract relevant patterns when trained on a bigger sample
- The compounded normalization never outperforms the standard one, while the mean normalization scores similarly to the standard procedure

Evaluation Results

Recall@N for long tail movies [We remove the most popular items (jointly accounting for 33% of the ratings) from the test set as in (Cremonesi et al., 2010)]:

	$m = 5$			$m = 50$		
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

- Generally lower accuracy, which can be attributed to the omission of popular items
- As before the mean normalization is better than the compound one
- The compound normalization benefits the most from this setup
- The mean normalization clearly outperforms the standard one, as we propose more specific desirable items (contributing to **diversity**)

Concluding Remarks

Conclusion

- We have augmented LOD-based recommenders with graph centrality measures (to reduce the influence of popular items)
- We have proposed two path normalizations: compound and average (compound is more sensitive to popular items)
- Using the `MovieLens` 1M dataset the average normalization provides better accuracy for long tail items

Future Work

- Apply the proposed method to other domains than movies (e.g., book and music)
- Experiment with other centrality measures (e.g., betweenness centrality) and normalization procedures (e.g., harmonic mean)
- Incorporate diversity measures into the recommendations (e.g., as in (Noia et al., 2017))

References

Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*, pages 1–24, 2011.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-N recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

Cataldo Musto, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Semantics-aware recommender systems exploiting linked open data and graph-based features. *Knowledge-Based Systems*, 136:1–14, 2017.

Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. 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.

Tommaso Di Noia, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. Adaptive multi-attribute diversity for recommender systems. *Information Sciences*, 382:234–253, 2017.

Barry Schwartz. *The Paradox of Choice*. Harper Perennial, 2004.

Thomas Wever and Flavius Frasincar. A linked open data schema-driven approach for top-N recommendations. In *Proceedings of the 32nd ACM SIGAPP Symposium on Applied Computing (SAC 2017)*, pages 656–663. ACM, 2017.