

Motivation

The Semantic Web allows us to make queries against a huge amount of data. A typical problem a user has to deal in such scenario is to decide which results are the most “interesting” ones, even though they are all equally valid. Here we show several approaches in that consider topological properties of the RDF in order to obtain a ranking that helps user visualization of results.

Our approach

Our strategy consists of using graph metrics to calculate the relevance of different nodes in a specific RDF graph. In particular, we propose an iterative ranking algorithm based on closeness centrality and clustering where distances are related to the importance of different paths and the clustering criteria on the similarity of different nodes. Our methods equally apply to rich social network data as RDF.

The algorithm

Let's consider we have (at least) two types of entities of interest from a dataset.

0) Create two graphs based on the entities of interest. The weights are based on the relation between the nodes

Repeat until ranking converges:

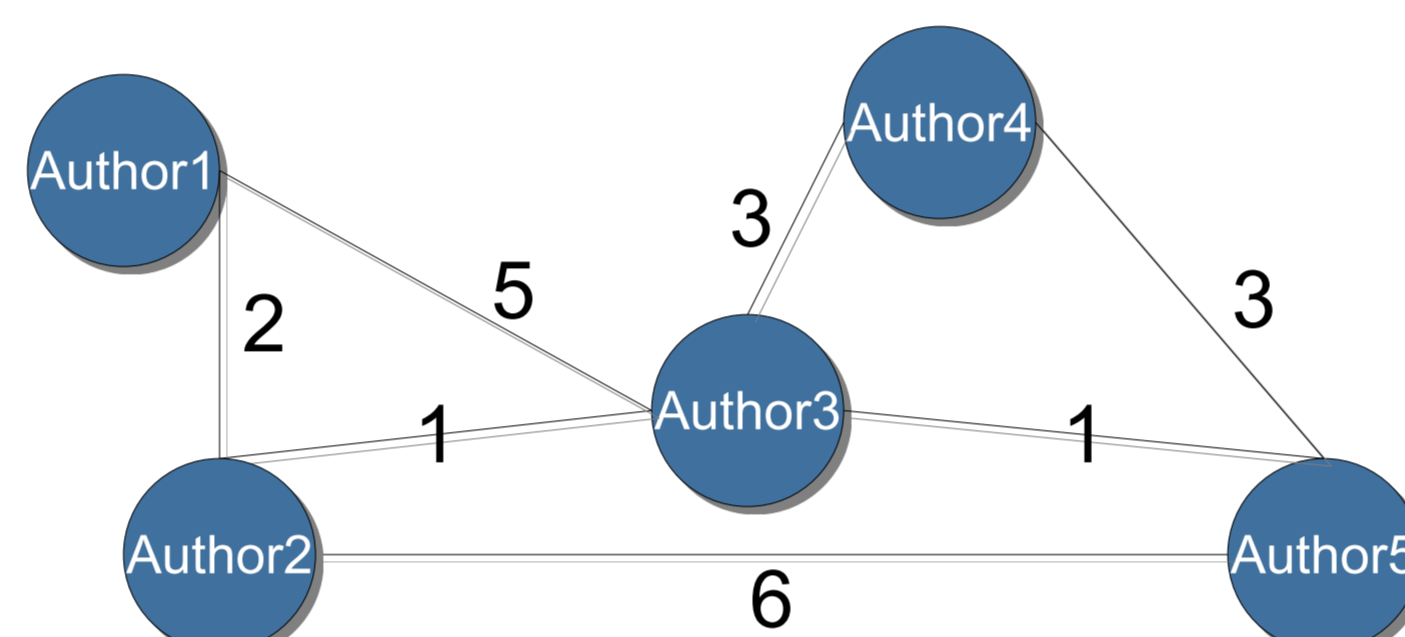
1) Compute centrality and obtain ranking for one of the graphs.

2) Create the weights of the second graph based on the ranking of entities in the first graph.

3) Compute centrality for the second graph and adapt weights in the first graph

Ranking DBLP

We applied our algorithm to DBLP, where our first graph contained authors as nodes and the edge's weights were based on how many publications they co-authored.



For our second graph we considered the different venues as nodes and the edges depended on how many authors have in common and their ranking.

Clustering graph

We improve ranking by considering locally important nodes. To find it, we rank locally dense sub-graphs (or clusters) that are limited to a specific domain. We find clusters using local optimization strategy which finds components based on the similarity of nodes and with respect to a particular predicate. Clusters can be used for:

- Finding nodes important to a specific domain
- Summarize the neighborhood of the node based on clusters.
- Constructing diverse result set that contains responsive nodes from different domains.

Ranking DBLP

We present the results for DBLP. We calculated the ranking for more than 500.000 different authors and over 4.000 conferences. We compared the author's results with their h-index. The first two tables are the global ranking and the ranking in the AI community cluster.

Pos	Author	h-idx	Pos	Author	h-idx
1	Philip Yu	53	6	Leonidas Guibas	51
2	Wan Wei	57	7	Zhang Li	69
3	Han Jiawei	74	8	HV Jagadish	53
4	Li Ming	50	9	Zhang HongJiang	38
5	Jeffrey Ullman	77	10	Hector Garcia-Molina	84

Table 1: Top10 ranked authors in global ranking

Pos	Author	h-idx	Pos	Author	h-idx
1	Manuela Veloso	53	6	Milind Tambe	51
2	Peter Stone	57	7	Michael Kearns	69
3	Hiroaki Kitano	74	8	Gal Kaminka	53
4	Minoru Asada	50	9	Sebastian Thrun	38
5	Satinder Singh	77	10	Patrick Riley	84

Table 2: Top10 ranked authors in AI cluster

The next two tables shows the results for the venues: the first one for the global ranking and the second for the AI cluster

Pos	Venue	Pos	Venue	Pos	Venue	Pos	Venue
1	PODS	6	Comp Geom	1	ICRA	6	ECAI
2	STOC	7	TODS	2	IJCAI	7	AAMAS
3	Symp on Comp Geom	8	SIAM	3	NIPS	8	ICALT
4	FOCS	9	ECCC	4	WebNet	9	ICML
5	SODA	10	PODC	5	GECCO	10	UAI

Table 3: Top10 ranked venues

Table 4: Top10 ranked venues

Ranking prog. languages from dbpedia

In the case of dbpedia, we used a subgraph of the programming languages linked by “influenced” predicates. After running our algorithm we present the results.

Pos	Language	Pos	Language
1	Python	359	JustBasic
2	Java	360	QBasic
3	C	361	MSXBasic
4	Perl	362	QuickBasic
5	C#	363	Basica
6	List	364	GWBasic
7	Haskell	365	XBasic
8	C++	366	Profile Scripting Lang.
9	Smalltalk	367	Cache Scripting Lang
10	Javascript	368	Vilnius Basic

Table 5: Top and bottom 10 ranked programming languages

Conclusions

In this work, we have introduced a simple iterative method for ranking nodes in RDF and social networks, which is based on centrality and clustering in an RDF graph. We have found that closeness centrality can be used to define an order that can be helpful for sorting large sets of nodes from the results of a SPARQL query. Also, we have found that clustering can help to improve the results for specific domains: thus, in our experiments with DBLP we have found that centrality and clustering can rank authors and venues showing very relevant people and conferences on top. Finally, we have implemented a parallel architecture that allow us to work with graph with millions of nodes.