RELISON: A Framework for Link Recommendation in Social Networks

Javier Sanz-Cruzado
University of Glasgow
javier.sanz-cruzadopuig@glasgow.ac.uk

Pablo Castells
Universidad Autónoma de Madrid
pablo.castells@uam.es

ABSTRACT

Link recommendation is an important and compelling problem at the intersection of recommender systems and online social networks. Given a user, link recommenders identify people in the platform the user might be interested in interacting with. We present RELISON, an extensible framework for running link recommendation experiments. The library provides a wide range of algorithms, along with tools for evaluating the produced recommendations. RELISON includes algorithms and metrics that consider the potential effect of recommendations on the properties of online social networks. For this reason, the library also implements network structure analysis metrics, community detection algorithms, and network diffusion simulation functionalities. The library code and documentation is available at https://github.com/ir-uam/RELISON.

CCS CONCEPTS

• Information systems → Information retrieval → Retrieval tasks and goals → Recommender systems • World Wide Web → Web applications → Social networks

KEYWORDS

Link recommendation, social network analysis, reproducibility, link prediction.

ACM Reference format:

Javier Sanz-Cruzado and Pablo Castells. 2021. RELISON: A Framework for Link Recommendation in Social Networks. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3477495.3531730

1 INTRODUCTION

The importance of online social networks like Facebook, Twitter, Instagram, TikTok or LinkedIn has grown beyond expectations since their emergence in the late 1990s. [9]. Hundreds of millions of people access these platforms every day to share content, discover new interests, and establish new relations with people around the world. The massive adoption of social network platforms has motivated the study of many different aspects around them, such as network structures, the way communities of users are formed, the mechanisms behind network evolution, or how information travels through the network. The online social network phenomenon has also raised new challenges and opportunities in fields such as information retrieval and recommender systems [100].

One of the compelling challenges at the confluence of online social networks and recommender systems consists of recommending people in the social network with whom a user might be interested to connect [45,92]. Link recommendation (also referred to as contact recommendation) has many interesting peculiarities with respect to the traditional recommendation task. Usually, users and items are separate sets. However, the candidate users in link recommendation are extracted from the same set of people to whom the recommendations are delivered: the same entities (people) are direct and indirect objects of recommendation here. Also, link recommendation algorithms can exploit different aspects of the social information around users to improve recommendations.

Link recommendation can moreover become an important driving force in how network takes shape, and a new agent in emerging network phenomena. For instance, added edges in the network give rise to the formation of new communication channels, not only for the two people directly involved in the social link, but for their local environments and farther. When a contact recommendation is accepted, the new link has potential impact on the network behavior and evolution [39].

Recommender system research and development involves techniques and methodologies from different areas, such as machine learning, information retrieval, statistics, human-computer interaction, and psychology. The variety of algorithmic approaches and evaluation methodologies [13] is a challenge for experimental design and reproducibility [21]. A good number of frameworks have been released to provide reproducible algorithm implementations, along with evaluation procedures and metrics [3, 28, 37, 51, 61, 71, 88, 99, 101, 105].

In addition to this methodological variety, further areas converge into link recommendation, namely social network analysis and network science. Not surprisingly, many contact recommendation approaches are derived from so-called *link prediction* in network science [65,69], a task that aims to identify links in the network that may be created in the future, commonly formulated as a classification problem.

SIGIR '22, July 11–15, 2022, Madrid, Spain

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
DOI: https://doi.org/10.1145/3477495.3531730

	Suggests	Social network	Primary task type	RELISON
Social rec.	People or items	✓	Ranking	
People rec.	People		Ranking	
Link pred.	Network links	✓	Classification	✓
Link rec.	People	✓	Ranking	✓

Table 1. Specific properties of the related tasks.

We aim to integrate both perspectives (link recommendation and link prediction) in our framework for their comparison under a common configuration. Our framework thus extends currently available recommender system frameworks with specific methods for recommending people in networks. Conversely, currently available software addressing the link prediction problem [54,58,67] do not support the specific formulation and methodology to apply prediction as a recommendation task.

The proposed RELISON framework is an extensible Java library for running and evaluating link recommendations in social networks.[1] This framework does not only consider the traditional accuracy-targeting problem, but also the potential effects of recommendations over social networks properties and behavior [2,19,22,50,79,90,93,98,97]. As part of this purpose, RELISON integrates functionalities to analyze the structure of the network and the flow of information that travels through social networks.

2 RELATED WORK

Before describing the RELISON framework, we provide a brief overview of the link prediction and recommendation tasks, as well as a summary of related available software libraries.

2.1 Link recommendation

Improving recommendations by exploiting online social network structures and traces is the motivation of social recommender systems [100]. Link recommendation is a particular case where the goal is to identify a subset of the people in the network with whom the target user might be interested in befriending or interacting [45, 92].

Recommending social links differs from other traditional recommendation tasks: in most domains, users and items are different objects; whereas in people recommendation, items range in the set of system users. This is also true for reciprocal recommendation in such domains as online dating or job recruiting [78]. In those domains, recommenders however a) do not need to have an underlying social network and b) an accepted connection between two users is somehow exclusive to them (for instance, once you find a job, it is not likely that you look for another job at least for some time). The framework we introduce here is only focused on people recommendation in social networks and does not cover other people recommendation problems like dating.

In a way, link recommendation can be seen as a link prediction problem [65,69], where existing but unobserved links, or links that will be created in the future, are sought to be identified. While being very similar tasks, link prediction and recommendation are mostly differentiated by how links are ranked and evaluated. When we want to recommend people, we generate independent rankings for each user in the network, and we evaluate them using information retrieval ranking metrics like precision, recall or nDCG [5,53]. In contrast, link prediction is commonly addressed as a classification problem, where a global ranking of the unobserved edges is produced, according to how likely they are to appear in the network according to the estimation. This global ranking is typically evaluated using classification metrics like accuracy, or AUC [31]. Because of this close relationship, RELISON includes tools for the execution and evaluation of link prediction. We summarize the properties of the different tasks we have introduced here in Table 1.

Many methods have been proposed for link prediction and recommendation, such as topological information [65,69,91], random walks [39,104], user-generated contents [47], machine learning classifiers [66] and similarity of node embeddings [42,70]. Also, since the beginning of the 2010s, the most important social network platforms have started providing link recommendation services, such as 'Who To Follow' on Twitter [39,44,94], or LinkedIn's 'People You May Know' [50].

Most work on link recommendation and prediction has targeted the accuracy of the recommendations, i.e. maximizing the amount of links added to the network thanks to the recommendation. However, in the last few years, several works have addressed the effect on the network as a whole: recommendation can modify the network topology [2,22,50,79,90,98], the information flow through it [19,93], or mitigate negative phenomena like the glass ceiling effect [97]. In the RELISON framework, we also consider these side effects.

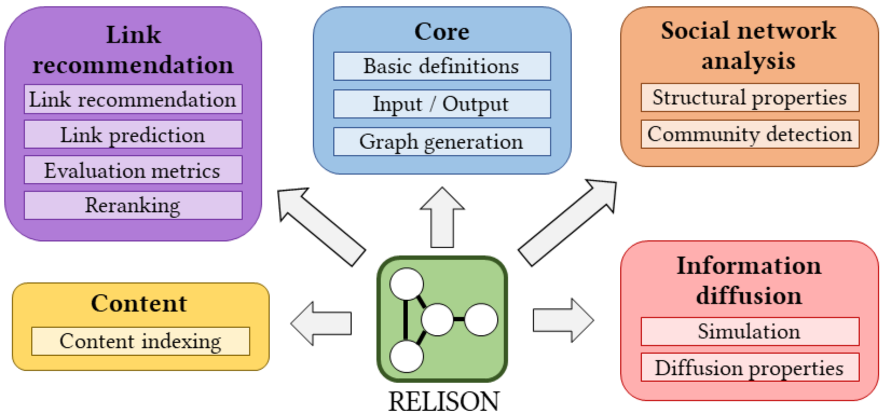
2.2 Related frameworks

As the RELISON framework appears in the intersection between recommender systems and social network analysis, we summarize here some of the most related software packages.

Reproducibility is known to be a non-trivial problem in recommender systems research [21]: the extent of experimental design options [13], the configuration alternatives of individual algorithms [77], and the application domain diversity make the comparison of recommendation methods a challenging task. Several software frameworks have been released to help address these issues, providing algorithm implementations, as well as evaluation procedures and metrics. This is the case of libraries such as BetaRecsys [71], Cornac [88], DaisyRec [99], DeepRec [108], Elliot [3], LensKit [28], LibRec [43], LightFM [61], MyMediaLite [37], OpenRec [105], RankSys [101] or Surprise [51]. Differently from our framework, these libraries have been developed for the general item recommendation case. Although they could also be applied for recommending people in social networks, they lack implementations of specific methods for the task – which we cover in RELISON.

[1] The code and documentation for the RELISON library is available at https://github.com/ir-uam/RELISON.

Figure 1. RELISON framework component overview.



On the other hand, the success and proliferation of online social network platforms has fueled research and the need for automated tools, transcending to complex networks in general, in areas like biology, psychology, physics, or linguistics. This demand is reflected in the creation of general-purpose network analysis frameworks like Jung[2], SNAP [64], iGraph [20], NetworkX [46], Graph-tool[3] or JGraphT [72]. These libraries offer tools for graph creation and manipulation, search in graphs, network visualization, structural analysis, community detection, information diffusion, node classification and link prediction.

Network science being a wide field, available software libraries tend to focus on specific problems. This is the case of libraries such as CDlib [85] or NDlib [86], focusing on community detection and information diffusion simulation, respectively.

In addition to link recommendation, our framework provides some of the functionalities covered by the previous libraries, so it does not have to rely on other software packages to understand the effects of different approaches on the network: it has tools for measuring structural properties of the graph, detecting communities, and simulating the diffusion of information in the network. However, as RELISON focuses on link recommendation, it provides a smaller collection of network algorithms (for instance, it lacks search algorithms in graphs, like A* or node coloring algorithms). Also, it does not provide tools for visualizing networks, for which we rely on other toolkits such as Gephi [7].

More closely related to our framework, several representative link prediction libraries have been released: LPMade [67] provides some classical unsupervised and supervised link prediction algorithms, whereas LinkPred [58] includes some classical approaches along with more recent methods based on graph embedding algorithms as node2vec [42]. These link prediction libraries support the classification view of the problem – they do not supply code for running and evaluating predictions in recommendation mode. RELISON supports both modes: recommendation and prediction.

3 RELISON

RELISON is available under the Mozilla Public License v.2.0.[4] The library provides tools for generating and evaluating link recommendations. It furthermore includes network structure analysis functionality, and tools for simulating the diffusion of information through networks. These functionalities can be used regardless of whether we have applied a recommendation over the network or not: although the main goal of this library is to support link recommendation, it can also be used as a tool for social network analysis. As shown in Figure 1, RELISON comprises five different modules that we describe in this section. Tables 2, 3, 4 and 5 compare RELISON to other social network libraries.

3.1 Core

The *core* RELISON module contains the basic functionalities for building social graphs, supporting the following network configurations:

- **Simple or multigraph:** depending on whether we allow a single or multiple links between two users.
- **Directed or undirected:** in undirected networks, relations are always reciprocated whereas in directed networks this is not necessarily the case.
- **Weighted or unweighted:** in weighted networks, we assign different weights (positive or negative) to network edges. Otherwise, edges are just binary (1 if the link exists, 0 otherwise).
- **Edge types**: it is also possible to assign an integer label to the edges of the network (indicating types of relations, creation timestamps, etc.).

The core module also includes input/output classes for reading and writing graphs from / to files, and random network generation, including random attachment [30], preferential attachment [5], and Watts-Strogatz small-world networks [103].

3.2 Link recommendation

The *link recommendation* module includes the main tools for recommending people, along with methods for evaluating these recommendations. This module is built upon the RankSys recommendation framework [101]. As a general recommendation library, RankSys already provides some common collaborative algorithms and a wide variety of accuracy, novelty and diversity

[2] Jung: http://jung.sourceforge.net
[3] GraphTool: https://graph-tool.skewed.de
[4] MPL v.2.0. https://www.mozilla.org/en-US/MPL/2.0 (Accessed 10th June 2021)

	Language	Structural metrics	Community detection	Random graph generators	Link prediction	Link recommendation	Information diffusion
RELISON	Java	✓	✓	✓	✓	✓	✓
Jung	Java	✓	✓	✓			
NetworkX [46]	Python	✓	✓	✓	✓		
Graph-tool	Python	✓	✓	✓			✓
JGraphT [72]	Java	✓		✓	✓		
iGraph [20]	R, Python, C++	✓	✓	✓	✓		
SNAP [64]	C++, Python	✓	✓	✓	✓		✓
Neo4j	Java	✓	✓		✓		
CDLib [85]	Python		✓				
NDLib [86]	Python						✓
LPMade [67]	C++				✓		
LinkPred [58]	C++				✓		

Table 2. Social network framework comparison.

			Neighbor-based						Path-based					Random walks							Collaborative filtering			Super-vised		Other		
	Popularity	Random	Adamic-Adar [1, 65]	Cosine [69]	IR models [91]	Jaccard [54]	MCN [65]	Total neighbors	Shortest distance [65]	Global LHN index [62]	Katz [56,65]	Local path index [68]	Matrix forest [16,69]	Commute time [29,34,65]	Hitting time [29,34,65]	Pers. HITS 39,44, 59]	Pers. PageRank [104]	Pers. SALSA [39,44, 63]	PropFlow [66]	SimRank [55]	User-based kNN [77]	Item-based kNN [77]	Implicit MF [49]	LambdaMART [11,36,91]	ML Classifiers [66]	HRB [17]	Graph embeddings [42]	**Num. total**
RELISON	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓			**55**
NetworkX	✓	✓				✓																						**8**
JGraphT	✓		✓	✓		✓	✓																					**10**
iGraph																										✓		**3**
Neo4j	✓		✓				✓	✓																			✓	**11**
SNAP																											✓	**1**
LPMade	✓	✓	✓			✓	✓				✓						✓		✓	✓					✓			**13**
LinkPred	✓		✓	✓		✓	✓	✓	✓			✓													✓		✓	**21**

Table 3. Link prediction and recommendation methods.

metrics that can be readily applied to the contact recommendation task. This includes collaborative filtering methods like nearest-neighbors [77] and several implementations of matrix factorization [49,80]. Evaluation metrics include common relevance-oriented ranking metrics (precision, recall [5] and nDCG [53]), as well as novelty [14,102] and diversity metrics [15,32,102,107].

On top of that, this module adds more than fifty link recommendation algorithms, as summarized in Table 3. This includes unsupervised methods such as:

- Methods based on common neighbors: Adamic-Adar [1], Jacccard similarity [54], common neighbor count [65] and recent adaptations of information retrieval models for recommending people in social networks [91].
- Methods based on the paths between the users in the network: Katz [56], local path index [68], shortest distance between the target and candidate users [65].
- Methods based on random walks: personalized PageRank [104], hitting and commute time [29,34,65], PropFlow [66] and Money (proposed by Twitter for their 'Who-to-follow' service) [39,44].
- A method based on user-generated content: Twittomender [47].

RELISON also provides supervised methods based on machine learning classifiers [66] using the Weka library [35] as a basis, and a learning to rank algorithm, LambdaMART [11], using the implementation provided by Ganjisaffar et al. [36].[5]

All the previous approaches can also be used for the link prediction task [65,69], where, as stated before, instead of taking an individual ranking for each user, we produce one global ranking, sorting the missing links in the network by their probability of occurrence. RELISON allows producing such global rankings and includes classification metrics to evaluate link prediction: accuracy, precision, recall, the F1-score and the area under the ROC curve [31].

Finally, this module also includes greedy rerankers targeting specific structural properties of the network. For this, the framework implements the Global Greedy Reranking algorithm in [93].

[5] https://github.com/yasserg/jforests

	Graph metrics								Vertex metrics								
	ASL [75]	Clustering. coef. [103]	Diameter [52]	Degree assortativity [73]	Degree Gini [90]	Density [75]	Radius [52]	**Num. total**	Betweenness [74]	Closeness [75]	Clustering. coef. [103]	Coreness [95]	Eccentricity [52]	HITS [59]	Katz centrality [56]	PageRank [10]	**Num. total**
RELISON	✓	✓	✓	✓	✓	✓	✓	**18**	✓	✓	✓	✓	✓	✓	✓	✓	**14**
Jung			✓					**2**	✓	✓	✓			✓		✓	**8**
NetworkX	✓	✓	✓	✓		✓	✓	**11**	✓	✓			✓	✓	✓	✓	**12**
Graph-tool	✓	✓	✓	✓				**8**	✓	✓	✓			✓	✓	✓	**9**
JGraphT		✓	✓				✓	**4**	✓	✓	✓	✓	✓		✓	✓	**10**
iGraph	✓	✓	✓	✓		✓	✓	**10**	✓	✓	✓	✓	✓	✓		✓	**12**
SNAP			✓					**3**	✓	✓	✓	✓	✓	✓		✓	**13**
Neo4j		✓						**1**	✓	✓	✓			✓		✓	**9**

	Edge / pair metrics						Community metrics					
							Indiv.			Global		
	Betweenness [74]	Distance [75]	Embeddedness [109]	Geodesics	Neighbor overlap [65]	**Num. total**	Comm. Size	Comm. Degree	Comm. Closeness	Modularity [4,18]	Edge Gini [90]	**Num. total**
RELISON	✓	✓	✓	✓	✓	**11**	✓	✓		✓	✓	**8**
Jung	✓	✓				**2**	✓					**2**
NetworkX	✓	✓			✓	**7**	✓	✓	✓	✓		**8**
Graph-tool	✓	✓		✓		**4**	✓			✓		**3**
JGraphT	✓	✓	✓	✓	✓	**7**						**0**
iGraph	✓	✓		✓		**5**	✓			✓		**3**
SNAP	✓	✓			✓	**5**	✓	✓	✓	✓		**4**
Neo4j		✓				**1**						**0**

Table 4. Structural metrics for social network analysis.

Although the rerankers included in the framework can be used for any structural property, most of them involve recomputing the metric values for the network after adding or removing a link. Nonetheless, for some of them, we include specific, optimized rerankers that exploit the formulation of the metrics for a faster computation. This is the case of the clustering coefficient and the Gini-based metrics that consider communities [90,93]. Some details about how these optimizations have been achieved can be seen in [89].

3.3 Social network analysis

The *social network analysis* module provides tools for understanding the structural properties of the network and automatically detecting clusters of users.

3.2.1 Structural metrics. RELISON provides a selection of common network topology analysis metrics. Following the work by Newman [75], we include the most important and common measures, such as the network diameter [52], the clustering coefficient [103] and modularity under a community partition [4,18]. In addition, we include structural diversity metrics based on the presence of weak ties, introduced in our prior work [90,93].

The framework classifies these metrics according to the network element they study: node metrics target properties of individual users (e.g. closeness, degree centrality [75]), edge/pair metrics analyze pairs of users (e.g. distance or embeddedness [27,109]), graph metrics consider the complete network structure (e.g. global clustering coefficient [103], diameter), and community metrics operate on node partitions. The latter metrics are divided into two groups: individual community metrics, considering properties of each community (like their size or their degree), and global graph metrics, such as the modularity [4,18] and the community edge Gini complement [90,93]. We list some of them in Table 4.

3.2.2 Community detection. People in online social networks tend to group in communities: groups of users tightly connected to each other, but with only a few connections to the rest of the network [33]. In our framework, we include several algorithms for detecting them: FastGreedy [74], Girvan-Newman [38], Infomap [87], label propagation [82], Louvain [8] and spectral clustering [106], along with algorithms for finding connected components [75]. We compare this set of algorithms with the ones provided by other frameworks in Table 5. Frameworks not appearing in the table do not support community detection.

RELISON relies on the modularity metric [4,18] for assessing the community partition quality. Other libraries like NetworkX [46] or CDlib [85] include a wider variety of metrics and techniques to evaluate community partitions, such as comparing the found communities to a ground truth partition.

3.4 Information diffusion

The diffusion of information in online social networks [106] is one of its foremost functionalities: people are constantly creating and sharing content. Online social networks further allow resharing and retweeting content published by other users. Understanding how information propagates in social networks is therefore a compelling (although complex) task.

The *information diffusion* module in RELISON provides tools for simulating this exchange of information. Other frameworks providing information diffusion functionalities, like NDlib [86] or Graph-tool, focus on scenarios where a single disease spreads through a network [48], or scenarios where opinions about a single topic are formed and evolve over time [96]. The simulator we provide in our framework takes a different direction: it simulates the exchange of multiple user-generated contents at the same time, about different topics (as it occurs in real network scenarios).

The RELISON simulator is highly configurable. Though we include some preconfigured diffusion protocols, such as the linear threshold model [57], the independent cascade model [40], or the

	Connected comp. [75]	Fast Greedy. [103]	Infomap [87]	Label propagation [82]	Louvain [90]	Spinglass [83]	Walktrap [81]	**Num. total**
RELISON	✓	✓	✓	✓	✓			**8**
Jung	✓							**4**
NetworkX			✓					**3**
iGraph	✓	✓	✓	✓	✓	✓	✓	**12**
SNAP					✓			**5**
Neo4j	✓			✓	✓			**7**
CDLib			✓	✓	✓	✓	✓	**40**

Table 5. Community detection algorithms.

push-pull model [23,24,25], library users can build custom diffusion protocols. Protocols define the way people in the network choose information pieces to propagate, which users receive and/or read the contents, under which criteria a user propagates one of the received contents, etc.

To analyze the outcome of these simulations, the module includes several metrics for such aspects as diffusion speed, how equally users receive new information, and how novel and diverse is the information received by the people in the social network [93].

3.5 Content

Finally, the *content* module works with the different contents generated by the users. At the moment of writing this paper, it can be just used to generate inverted indexes [5,12] to store the information about these contents (so they can be used for executing content-based recommendation approaches). In the future, this might be extended to study search problems in social network environments.

3.6 Executable programs

In addition to the previous modules, the RELISON framework provides a series of command-line programs that can be executed to perform different tasks:

- **Link recommendation:** the main program allows executing and evaluating recommendations using a) accuracy [5] and b) novelty and diversity [14] metrics. Another analyzes the effects of recommendation on the network structure [90,93]. The third program applies reranking algorithms over previously computed recommendations. A fourth program computes feature vectors for supervised algorithms [66].
- **Link prediction.** Differently from link recommendation, there is a single program to run and evaluate link prediction algorithms. However, we can also measure the effect of link prediction on network structure with the same program we provide for contact recommendation.
- **Social network analysis.** Two programs are provided: one that analyzes the structural properties of networks, and one that runs different community detection algorithms on a social network.
- **Information diffusion.** One program runs the simulation cycle, and another one carries out a set of measurements on the simulation.
- **Other:** we provide additional programs for a) generating random network graphs and b) creating inverted indexes from user-generated contents.

All these programs can be configured via their input parameters and YAML configuration files. In the next section, we illustrate a use case, in which we apply some of the previous programs to understand the effect of link recommendation on network structure and information diffusion.

Multigraph	✗
Directed	✓
Weighted	✓
# Users	9,528
# Training edges	170,425
# Test edges	54,335
# Tweets (total)	1,558,518
# Tweets (test)	622,795

Table 6. Details of the Twitter dataset.

4 EXAMPLE USE CASE

We use an example to illustrate how RELISON works, following [93]: given a social network, we first generate recommendations for a set of users; then we evaluate their effects on the network, and finally, we analyze how information propagates through the network.

4.1 Data

We run our example experiments over a social network graph downloaded from Twitter, which has been used in previous work [90,91,93]. In order to obtain it, we downloaded from the Twitter API all the tweets posted by a set of 10,000 users from June 16^{th} to July 16^{th} 2015. Then, we built a directed interaction network, where a directed link between two users indicates the source user has mentioned the target user, or retweeted one of their tweets, as reflected in the set of collected tweets.

For the experiments in this example, we split the network into training and test subsets: all interactions before July 9^{th} 2015 make up the training set, and the remaining ones the test set. Any edge appearing in both sets is removed from the test set, to avoid test data leakage. The frequency of interactions between each pair of users before the split time is used as the weight of the corresponding edge. We summarize in Table 6 the properties of this dataset, which is available in the GitHub repository, along with the code. The file names we shall use throughout the use case correspond to the names of the files in the repository.

4.2 Running link recommendations

Recommendation algorithms are run in the `recommendation` program provided in the framework. The program receives: a) the training and test networks (along with their configuration parameters), b) a YAML configuration file containing the information about link recommendations and metrics, c) the output directory, and d) a few additional recommendation parameters. The latter include the cutoff for the recommendation, whether recommendations should be produced for all users or just the ones involved in the test subset, and whether reciprocating link recommendation is allowed.

We run four link recommendation algorithms in this example: BM25 [84,91], implicit matrix factorization [49], popularity-based and random recommendation, with the hyperparameter configuration reported in [91]. We take a cutoff of 10 recommended links per user and, to avoid trivializing the problem, we do not consider recommending reciprocal edges. We use nDCG@10 and MAP@10 as evaluation metrics.

The command line for running the program is as follows:

```
java -jar RELISON.jar recommendation train.txt test.txt multigraph directed weighted selfloops readtypes algorithms-example.yaml output/ cutoff
```

where `multigraph`, `readtypes` and `selfloops` take the `false` value, `directed` and `weighted` take `true`, and `cutoff` is `10`.

We show part of the configuration file in Figure 2. We show in red the identifiers of the algorithms, and in blue the name of the algorithm parameters. We can see in the figure the configuration of the implicit matrix factorization algorithm [49]: the number of latent factors is $k = 300$, the regularization parameter is $\lambda = 150$, the rating confidence parameter is $\alpha = 40$, and the algorithm does not consider edge weights. The configuration for the rest of algorithms is available in the full file, which can be accessed through the link provided in the figure. Table 7 shows the outcome of this program. Confirming results reported in previous publications [91], iMF is the best algorithm under this setting, followed by BM25. As expected, popularity and random recommendation achieve much lower accuracy.

Figure 2. YAML configuration file for link recommendation.

```
algorithms:
  iMF:
    k:
      type: int
      values: 300
    lambda:
      type: double
      values: 150.0
    alpha:
      type: double
      values: 40.0
    weighted:
      type: boolean
      values: false
  BM25:
    <...>
metrics:
  nDCG:
    cutoff:
      type: int
      values: 10
  MAP:
    <...>
```

Link: https://github.com/ir-uam/RELISON/blob/master/Example configuration files/algorithms-example.yml

	nDCG@10	MAP@10
BM25	0.10416	0.04399
iMF	0.13865	0.06618
Popularity	0.05723	0.02908
Random	0.00107	0.00030

Table 7. Recommendation example results.

4.3 Effects on network structure

Once recommendations are computed and the effectiveness of the algorithms has been measured, we analyze the effect of recommendations on network structure. We use for this purpose two of the programs included in the framework.

We first check the original structural properties of the network using the *sna* program, which takes as input the network, a YAML configuration file containing the metrics to use, and a directory in which to store the outcome. The command line for this program is:

```
java -jar RELISON.jar sna train.txt multigraph directed weighted selfloops metrics-example.yaml output/ --distances
```

where `--distances` is an optional flag for precomputing the distances between users.

We then run the program named *effects*, that adds a set of recommended links back to the network from which the recommendations are produced (as in [90,93]). The structural metrics of interest are then computed over this extended network. The program takes as input the training and test networks (along with their configuration), a directory containing the recommendations, the YAML configuration file, a file for storing the output, the recommendation cutoff, a parameter indicating whether we wish to compute edge / pair metrics over all the network or just the recommended pairs, and a parameter indicating whether all recommended edges should be added or only the relevant ones (those appearing in the test set). The command line for this program is the following:

```
java -jar RELISON.jar effects train.txt test.txt multigraph directed weighted selfloops rec-folder/ metrics-example.yaml output.txt cutoff use-all-edges only-rel --distances
```

We now measure three properties of the network: the global clustering coefficient, the eccentricity of nodes, and the embeddedness of edges. As both *sna* and *effects* programs measure the same properties (over different networks), they share the same YAML configuration file. We show the YAML file for this experiment in Figure 3, with red for the metric names, and purple for their input parameters.

In our test, we measure the embeddedness of all the edges in the network (i.e. we set `use-all-edges` to `true`) and, following previous work [90,93], we add all the recommended links to the original network (i.e. we set `only-rel` to `false`). Results for executing these two programs are shown in Table 8. We see that different algorithms have diverse effects on the network. For instance, random recommendation reduces the value of all metrics

Figure 3. YAML configuration file for measuring structural properties of the network.

```
metrics:
  Clustering coefficient:
    type: graph
    params:
      uSel:
        type: orientation
        values: IN
      vSel:
        type: orientation
        values: OUT
  Eccentricity:
    type: vertex
  Embeddedness:
    type: edge
    <...>
```

Link: https://github.com/ir-uam/RELISON/blob/master/Example configuration files/metrics-example.yml

	Clustering coefficient	Average node eccentricity	Average edge embeddedness
BM25	0.12224	6.19689	0.02799
iMF	0.09851	6.69626	0.02571
Popularity	0.07575	6.00084	0.01542
Random	0.04839	4.25210	0.01479
Original network	0.05621	6.67338	0.02431

Table 8. Structural metrics example results.

with respect to the training network, while iMF has the opposite effect.

4.4 Effects on information diffusion

To conclude our example, we study the effect of recommendation on information diffusion.

For this purpose, the *diffusion* program simulates the information flow through the network. The program receives as input a YAML configuration file, the result output folder, the number of times we run each simulation, the network data, files containing the identifiers of users and content items (here, tweets) to be shared, and a file providing the authorship relationship between users and items. Optionally, we can read a file with recommended links to be added to the network, and additional information like user and item features. In this example, we use tweets as content items and hashtags as tweet features. The command line for running the program is the following:

```
java -jar RELISON.jar diffusion diffusion-example.yaml output/ numReps train.txt multigraph directed weighted selfloops readtypes user-index.txt info-index.txt tweets.txt -infofeats tweet-hashtag.txt (-rec rec-file.txt)
```

We now simulate the diffusion of information over the training graph introduced in Section 4.1 and the extended versions of this graph after adding the links recommended by the algorithms in section 4.2. The information to be propagated are the tweets created by users in the network after the time of the split.

Figure 4 shows the YAML configuration file. For each simulation, we must provide the program with three elements. First, a diffusion protocol: in the example, we use one of the pre-configured protocols, the independent cascade model [40], in which users propagate a piece of information received from another user with a fixed probability (here, $p = 0.001$). In addition, each user propagates one of her own created content items. Second, a stop condition, that indicates when the simulation must finish. Here, we stop it after 1,000 iterations. And third, a set of filters, which clean all the received information before the simulation starts. Here, we apply two different filters: the first one removes all the information items without a creator in the training graph; the second removes the tweets without any hashtag.

Figure 4. YAML configuration file for the diffusion simulation.

```
simulations:
- filters:
    Creator:
    Information feature:
      feature:
        type: string
        value: hashtag
  protocol:
    name: Independent cascade model
    type: PRECONFIGURED
    params:
      numOwn:
        type: int
        value: 1
      prob:
        type: double
        value: 0.001
  stop:
    name: Num. iter
    params:
      numIter:
        type: int
        value: 1000
```

Link: https://github.com/ir-uam/RELISON/blob/master/Example configuration files/diffusion-example.yml

Once the diffusion simulations have been run and their trace is stored, we run the second program, *diffusion-eval*, for analyzing the properties of the information flow across the network. This second program receives all data that was provided to the *diffusion* program (the network, the user-generated contents, etc.) and, in addition, a folder containing the trace produced by the simulations to be analyzed, and an output directory in which to store the metric results. The command line for this program is then:

```
java -jar RELISON.jar diffusion-eval diffusion-metrics-example.yaml train.txt multigraph directed weighted selfloops readtypes user-index.txt info-index.txt tweets.txt diffusion/ output/ -infofeats tweet-hashtag.txt
```

The YAML configuration file for this program is illustrated in Figure 5. In this case, it has two parts: a list of data filters (the same as in the configuration file for the *diffusion* program), and the diffusion properties we want to measure. In this example, we are measuring two properties: the diffusion speed (how many information items have been received by all the users in the network over time), and a diversity metric, measuring how balanced the distribution of the received hashtags in the network is (using the complement of the Gini index [26]).

We plot the outcome of this program for our example in Figure 6, where the x axis shows the number of iterations in the simulation, and the y axis shows the value of each diffusion metric at the given point of the simulation. As can be observed, adding the links of all recommendations increases the speed of diffusion but, in general, decreases the diversity of the information that users receive – the only exception is the random recommender, which also increases the diversity of the information received by the users.

5 CONCLUSION

We have introduced RELISON, an extensible Java framework for experimentation in link recommendation. The framework provides a large collection of state-of-the-art contact recommendation algorithms, along with ranking-based metrics for evaluating them, including accuracy, novelty and diversity metrics [13,14]. The library allows measuring structural properties of networks – by the implementation of more than fifty network analysis metrics –, finding communities and analyzing how the information travels through social networks.

To the best of our knowledge, RELISON represents the first framework addressing link prediction a proper recommendation task, and also the first to consider the effects that the recommendations have on the network.

The framework can be extended in the future to include more link recommendation and prediction algorithms, like those based on graph embeddings [42,70]. We plan to add further functionality for more general social recommendation [100], where we might consider the traces and structures of online social networks to support the recommendation of items like the contents generated by the users in the network (tweets, posts).

ACKNOWLEDGMENTS

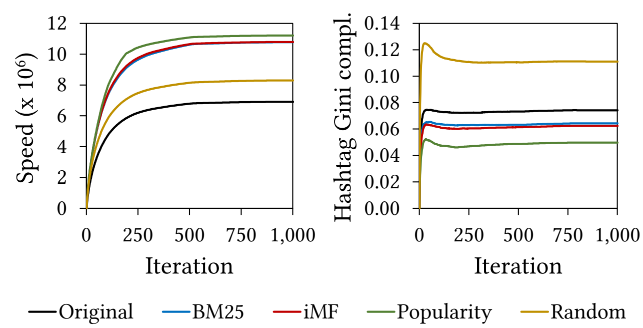
This work has been partially funded by the Spanish Government (grant ref. PID2019-108965GB-I00). This work was carried out as part of the Infinitech project which is supported by the European Union's Horizon 2020 Research and Innovation programme under grant agreement no. 856632.

Figure 5. YAML configuration file for the diffusion simulation metrics.

```yaml
filters:
  Creator:
  Information feature:
    feature:
      type: string
      value: hashtag
metrics:
  Speed:
  Global feature Gini complement:
    feature:
      type: string
      value: hashtag
    userFeature:
      type: boolean
      value: false
    unique:
      type: boolean
      value: true
```

Link: https://github.com/ir-uam/RELISON/blob/master/Example configuration files/diffusion-metrics-example.yml

Figure 6. Results of the information diffusion simulation.



REFERENCES

[1] Lada A. Adamic and Eytan Adar. 2003. Friends and neighbors on the Web. *Social Networks* 25, 3 (July 2003), 211-230. DOI: 10.1016/S0378-8733(03)00009-1.

[2] Luca M. Aiello and Nicola Barbieri. 2017. Evolution of Ego-networks in Social Media with Link Recommendations. In *Proceedings of the 10th ACM international conference on Web Search and Data Mining (WSDM 2017)*. ACM, Cambridge United Kingdom, 111-120. DOI: 10.1145/3018661.3018733.

[3] V. Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice A. Merra, Claudio Pomo, Francesco M. Donini and Tomasso Di Noia. 2021. ELLIOT: a Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation. In *Proceedings of the 44th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. ACM, Online, in press.

[4] Alexandre Arenas, Jordi Duch, Alberto Fernández and Sergio Gómez. 2007. Size reduction of complex networks preserving modularity. *New Journal of Physics* 9, 6 (June 2007), Article 176. DOI: 10.1088/1367-2630/9/6/176.

[5] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 2011. *Modern Information Retrieval: The Concepts and Technology Behind Search* (2nd ed.). Pearson Education Ltd.

[6] Albert-László Barabási and Réka Albert. 1999. Emergence of Scaling in Random Networks. *Science* 286, 5439 (September 1999), 509-512. DOI: 10.1126/science.286.5439.509.

[7] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy. 2009. Gephi: An Open Source Software for Exploring and Manipulating Networks. In *Proceedings of the 3rd AAAI Conference on Web and Social Media*. AAAI.

[8] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte and Ettiene Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, (October 2008), Article P10008. DOI: 10.1088/1742-5468/2008/10/p10008.

[9] danah m. boyd and Nicole B. Ellison. 2007. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication* 13, 1 (October 2007), 210-230. DOI: 10.1111/j.1083-6101.2007.00393.x.

[10] Sergey Brin and Larry Page. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proceedings of the 7th international conference on World Wide Web (WWW 1998)*. Elsevier, Brisbane, Australia, 107-117. DOI: 10.1016/S0169-7552(98)00110-X.

[11] Chris Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. *Microsoft Technical Report* MSR-TR-2010-82.

[12] Stefan Büttcher, Charles L.A. Clarke and Gordon V. Cormack. 2010. *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press.

[13] Rocío Cañamares, Pablo Castells and Alistair Moffat. 2020. Offline evaluation options for recommender systems. Information Retrieval Journal 23 (August 2020), 387-410. DOI: 10.1007/s10791-020-09371-3.

[14] Pablo Castells, Neil J. Hurley and Saúl Vargas. 2015. Novelty and Diversity in Recommender Systems. In: F. Ricci, L. Rokach, B. Shapira (eds.) *Recommender Systems Handbook (2nd. ed)*. Springer, Boston, MA, USA, 881-918. DOI: 10.1007/978-1-4899-7637-6_26.

[15] Olivier Chapelle, Shihao Ji, Emre Velipasaoglu, Larry Lai and Su-Lin Wu. 2011. Intent-based diversification of web search results: metrics and algorithms. *Information Retrieval* 14, 6 (May 2011), 572-592. DOI: 10.1007/s10791-011-9167-7

[16] Pavel Y. Chebotarev and Elena V. Shamis. 1997. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control* 58, 9 (September 2007), 125-137.

[17] Aaron Clauset, Christopher Moore and Mark E.J. Newman. 2008. Hierarchical structure and the prediction of missing links in networks. *Nature* 453 (May 2008), 98-101. DOI: 10.1038/nature06830.

[18] Aaron Clauset, Mark. E.J. Newman and Christopher Moore. 2004. Finding community structure in very large networks. *Physical Review E* 70, 6 (December 2004), Article 066111. DOI: 10.1103/PhysRevE.70.066111.

[19] Federico Corò, Gianlorenzo D'Angelo and Yllka Velaj. 2019. Recommending Links to Maximize the Influence in Social Networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*. IJCAI, Macao, China, 2195-2201. DOI: 10.24963/ijcai.2019/304

[20] Gabor Csardi, Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal Complex Systems*, 1695 (2006).

[21] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi and Dietmar Jannach. 2021. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Transactions on Information Systems* 39, 2 (March 2021), Article 20. DOI: 10.1145/3434185.

[22] Elizabeth M. Daly, Werner Geyer, David R. Millen. 2010. The Network Effects of Recommending Social Connections. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys 2010)*. ACM, Barcelona, Spain, 301-304. DOI: 10.1145/1864708.1864772.

[23] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Dan Swinehart and Doug Terry. 1987. Epidemic Algorithms for Replicated Database Maintenance. In *Proceedings of the 6th annual ACM Symposium on Principles of Distributed Computing (PODC 1987)*. ACM, Vancouver, BC, Canada, 1-12. DOI: 10.1145/41840.41841.

[24] Benjamin Doerr, Mahmoud Fouz and Tobias Friedrich. 2011. Social Networks Spread Rumors in Sublogarithmic Time. In *Proceedings of the 43rd Annual Symposium on Theory of Computing (STOC 2011)*. ACM, San Jose, California, USA, 21-30. DOI: 10.1145/1993636.1993640.

[25] Benjamin Doerr, Mahmoud Fouz and Tobias Friedrich. 2012. Why Rumors Spread So Quickly in Social Networks?. *Communications of the ACM* 55, 6 (June 2012), 70-75. DOI: 10.1145/2184319.2184338.

[26] Robert Dorfman. 1979. A Formula for the Gini Coefficient. *The Review of Economics and Statistics* 61, 1 (February 1979), 146-149. DOI: 10.2307/1924845.

[27] David Easley and Jon M. Kleinberg. 2010. *Networks, Crowds and Markets*. Cambridge University Press.

[28] Michael D. Ekstrand. 2020. LensKit for Python: Next-Generation Software for Recommender System Experiments. In *Proceedings of the 29th ACM international Conference on Information & Knowledge Management (CIKM 2020)*. ACM, Online, 2999-3006. DOI: 10.1145/3340531.3412778.

[29] Ivan Erdelyi. 1967. On the matrix equation Ax = λBx. *Journal of Mathematical Analysis and Applications* 17, 1 (January 1967), 119-132. DOI: 10.1016/0022-247X(67)90169-2.

[30] Paul Erdös and Alfréd Rényi. 1959. On Random Graphs I. *Publicationes Mathematicae Debrecen* (1959), 290-297.

[31] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (June 2006), 861–874. DOI: 10.1016/j.patrec.2005.10.010.

[32] Daniel Fleder and Kartik Hosanagar. 2009. Blockbuster Culture's Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity. *Management Science* 55, 5 (May 2009), 697-712. DOI: 10.1287/mnsc.1080.0974.

[33] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3 (February 2010), 75-174. DOI: 10.1016/j.physrep.2009.11.002.

[34] François Fouss, Alain Pirotte, Jean-Michel Renders and Marco Saraens. Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 19, 3 (March 2007), 355–369. DOI: 10.1109/TKDE.2007.46.

[35] Eibe Frank, Mark A. Hall and Ian H. Witten. 2016. The WEKA Workbench. In: *Data Mining: Practical Machine Learning Tools and Techniques* (4th ed.). Morgan Kaufmann, 533-552. DOI: 10.1016/B978-0-12-804291-5.00024-6.

[36] Yasser Ganjisaffar, Rich Caruana and Cristina Videira Lopes. 2011. Bagging Gradient-boosted Trees for High Precision, Low Variance Ranking Models. In *Proceedings of the 34th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011)*. ACM, Beijing, China, 85-94. DOI: 10.1145/2009916.2009932.

[37] Zeno Gantner, Steffen Rendle, Cristoph Freudenthaler and Lars Schmidth-Thieme. 2011. MyMediaLite: a free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)*. ACM, Chicago, Illinois, USA, 305-308. DOI: 10.1145/2043932.2043989.

[38] Michelle Girvan, Mark E.J. Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Science of the USA* 99, 12 (June 2002), 7821-7826. DOI: 10.1073/pnas.122653799.

[39] Ashish Goel, Pankaj Gupta, John Sirois, Dong Wang, Aneesh Sharma and Siva Gurumurthy. 2015. The Who-To-Follow system at Twitter: Strategy, algorithms and revenue impact. *Interfaces* 45, 1 (February 2015), 98-107. DOI: 10.1287/inte.2014.0784.

[40] Jacob Goldenberg, Barak Libai and Eitan Muller. 2001. Talk of the Network: A complex System Look at the Underlying Process of Word-of-Mouth. *Marketing letters* 12 (August 2001), 211-223. DOI: 10.1023/A:1011122126881

[41] Mark S. Granovetter. 1973. The Strength of Weak Ties. *American Journal of Sociology* 78, 6 (May 1973), 1360-1380. DOI: 10.1086/225469.

[42] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD 2016)*. ACM, San Francisco, California, USA, 855-864. DOI: 10.1145/2939672.2939754.

[43] Guibing Guo, Jie Zhang, Zhu Sun and Neil Yorke-Smith. 2015. LibRec: A Java Library for Recommender Systems. In *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation and Personalization (UMAP 2015)*. Dublin, Ireland.

[44] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang and Reza Zadeh. 2013. WTF: The Who to Follow Service at Twitter. In *Proceedings of the 22nd international conference on World Wide Web (WWW 2013)*. ACM, Rio de Janeiro, Brazil, 505-514. DOI: 10.1145/2488388.2488433

[45] Ido Guy. 2018. People Recommendation on Social Media. In: P. Brusilovsky, D. He (eds.) *Social Information Access: Systems and Technologies. Lecture Notes in Computer Science*, vol 1045. Springer, Cham, 511-543. DOI: 10.1007/978-1-4899-7637-6_15.

[46] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart. 2008. Exploring network structure, dynamics and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy 2008)*. SciPy, Pasadena, California, USA, 11-15.

[47] John Hannon, Mike Bennett and Barry Smyth. 2010. Recommending Twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, Barcelona, Spain, 199-206. DOI: 10.1145/1864708.1864746.

[48] Herbert W. Hethcote. 2000. The Mathematics of Infectious Diseases. *SIAM Review* 42, 4 (December 2000), 599-653. DOI: 10.1137/S0036144500371907

[49] Yifan Hu, Yehuda Koren and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*. IEEE, Pisa, Italy, 263-272. DOI: 10.1109/ICDM.2008.22.

[50] Xinyi L. Huang, Mitul Tiwari, and Sam Shah. 2013. Structural Diversity in Social Recommender Systems. In *Proceedings of the 5th ACM RecSys Workshop on Recommender Systems and the Social Web (RSWeb 2013) at the 7th ACM Conference on Recommender Systems (RecSys 2013)*. Hong Kong, China.

[51] Nicolas Hug. 2020. Surprise: A Python library for recommender systems. *Journal of Open Software* 5, 52 (August 2020), Article 2174. DOI: 10.21105/joss.02174.

[52] Aleksandar Ilić. 2012. On the extremal properties of average eccentricity. *Computer & Mathematics with Applications* 64, 9 (November 2002), 2877-2885. DOI: 10.1016/j.camwa.2012.04.023.

[53] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulative Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems* 20 (October 2002), 422-446. DOI: 10.1145/582415.582418.

[54] Paul Jaccard. 1901. Étude de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles* 37, 142 (January 1901), 547-579. DOI: 10.5169/seals-266450.

[55] Glen Jeh and Jennifer Widom. 2002. SimRank: A Measure of Structural-Context Similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*. ACM, Edmonton, Alberta, Canada, 538-543. DOI: 10.1145/775047.775126.

[56] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18 (March 1953), 39-43. DOI: 10.1007/BF02289026.

[57] David Kempe, Jon M. Kleinberg and Éva Tardos. 2003. Maximizing the Spread of Influence through a Social Network. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD 2003)*. ACM, Washington, DC, USA, 137-146. DOI: 10.1145/956750.956769.

[58] Said Kerrache. 2021. LinkPred: a high performance library for link prediction in complex networks. *PeerJ Computer Science* 7 (May 2021). DOI: 10.7717/peerj-cs.521.

[59] Jon M. Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM* 46, 5 (September 1999), 604-632. DOI: 10.1145/324133.324140.

[60] Yehuda Koren, Robert Bell and Chris Volinksy. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (August 2009), 30-37. DOI: 10.1109/MC.2009.263.

[61] Maciej Kula. 2015. Metadata Embeddings for User and Item Cold-start Recommendations. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with the 9th ACM Conference on Recommender Systems (RecSys 2015)*. Vienna, Austria, 14-21.

[62] Elizabeth A. Leicht, Petter Holme and Mark E.J. Newman. 2006. Vertex similarity in networks. *Physical Review E* 73, 2 (February 2006), Article 026120. DOI: 10.1103/PhysRevE.73.026120.

[63] Ronny Lempel and Schlomo Moran. 2001. SALSA: The Stochastic Approach for Link-Structure Analysis. *ACM Transactions on Information Systems* 19, 2 (April 2001), 131-160. DOI: 10.1145/382979.383041.

[64] Jure Leskovec and Rok Sosič. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology* 8, 1 (October 2016), Article 1. DOI: 10.1145/2898361.

[65] David Liben-Nowell and Jon M. Kleinberg. 2007. The Link-Prediction Problem for Social Networks. *Journal of the American Society for Information Science and Technology* 58, 7 (March 2007), 1019-1031. DOI: 10.1002/asi.20591.

[66] Ryan N. Lichtenwalter, Jake T. Lussier and Nitesh V. Chawla. 2010. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD 2010)*. ACM, Washington, DC, USA, 243-252. DOI: 10.1145/1835804.1835837

[67] Ryan N. Lichtenwalter, Nitesh V. Chawla. 2011. LPmade: Link Prediction Made Easy. The Journal of Machine Learning Research 12 (January 2011), 2489-2492.

[68] Linyuan Lü, Ci-Hang Jin and Tao Zhou. Similarity index based on local paths for link prediction of complex networks. *Physical Review E* 80, 4 (October 2009), Article 046122. DOI: 10.1103/PhysRevE.80.046122.

[69] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A* 390 (March 2011), 1150-1170. DOI: 10.1016/j.physa.2010.11.027.
[70] Zaiqiao Meng, Shangsong Liang, Hongyan Bao and Xiangliang Zhang. 2019. Co-Embedding Attributed Networks. In *Proceedings of the 12th ACM international conference on Web Search and Data Mining (WSDM 2019)*. ACM, Melbourne, Australia, 393-401. DOI: 10.1145/3289600.3291015.
[71] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, Iadh Ounis, Siwei Liu, Yaxiong Wu, Xi Wang, Shangsong Liang, Yucheng Liang, Guangtao Zeng, Junhua Liang and Qiang Zhang. 2020. BETA-Rec: Build, Evaluate and Tune Automated Recommender Systems. In *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys 2020)*. ACM, Online, 588-590. DOI: 10.1145/3383313.3411524
[72] Dimitrios Michail, Joris Kinable, Barak Naveh, John V. Sichi. JGraphT – A Java Library for Graph Data Structures and Algorithms. *ACM Transactions on Mathematical Software* 46, 2 (June 2020), Article 16. DOI: 10.1145/3381449.
[73] Mark E.J. Newman. 2002. Assortative Mixing in Networks. *Physical Review Letters* 89 (October 2002), Article 208701. DOI: 10.1103/PhysRevLett.89.208701.
[74] Mark E.J. Newman. 2004. Fast algorithm for detecting community structure in networks. *Physical Review E* 69, 6 (June 2004), Article 066133. DOI: 10.1103/PhysRevE.69.066133.
[75] Mark E.J. Newman. 2018. Networks (2nd. ed). Oxford University Press.
[76] Mark E.J. Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69 (February 2004), Article 026113. DOI: 10.1103/PhysRevE.69.026113.
[77] Xia Ning, Christian Desrosiers and George Karypis. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In: F. Ricci, L. Rokach, B. Shapira (eds.) *Recommender Systems Handbook (2nd ed.)*. Springer, Boston, MA, USA, 37-77. DOI: 10.1007/978-1-4899-7637-6_2.
[78] Iván Palomares, James Neve, Carlos Porcel, Luiz Pizzato, Ido Guy and Enrique Herrera-Viedma. Reciprocal Recommender Systems: Analysis of state-of-art literature, challenges and opportunities towards social recommendation. *Information Fusion* 69 (May 2021), 103-127. DOI: 10.1016/j.inffus.2020.12.001.
[79] Nikos Parotsidis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2016. Centrality-Aware Link Recommendations. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM 2016)*. ACM, San Francisco, California, 503-512. DOI: 10.1145/2835776.2835818.
[80] István Pilászy, Dávid Zibriczky and Domonkos Tikk. 2010. Fast ALS-based Matrix Factorization for Explicit and Implicit Feedback Datasets. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys 2010)*. ACM, Barcelona, Spain, 71-78. DOI: 10.1145/1864708.1864726.
[81] Pascal Pons and Matthieu Latapy. 2005. Computing Communities in Large Networks Using Random Walks. In *Proceedings of the 20th International Symposium on Computer and Information Sciences (ISCIS 2005). Lecture Notes in Computer Science,* vol. 3733. Springer, Istanbul, Turkey, 284-293. DOI: 10.1007/11569596_31.
[82] Usha N. Raghavan, Réka Albert and Soundar Kumara. 2007. Near line time algorithm to detect community structures in large-scale networks. *Physical Review E* 76, 3 (September 2007), Article 036106. DOI: 10.1103/PhysRevE.76.036106.
[83] Jörg Reichardt and Stefan Bornholdt. 2006. Statistical mechanics of community detection. *Physical Review E* 74, 1 (July 2006), Article 016110. DOI: 10.1103/PhysRevE.74.016110.
[84] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (April 2009), 333-389. DOI: 10.1561/1500000019.
[85] Giulio Rossetti, Letizia Milli and Rémy Cazabet. 2019. CDLIB: a python library to extract, compare and evaluate communities from complex networks. *Applied Network Science* 4 (July 2019), Article 52. DOI: 10.1007/s41109-019-0165-9.
[86] Giulio Rossetti, Letizia Milli, Salvatore Rinzivillo, Alina Sîrbu, Dino Pedreschi and Fosca Giannotti. 2018. NDLIB: a python library to model and analyze diffusion process over complex networks. *International Journal of Data Science and Analytics* 5 (February 2018), 61-79. DOI: 10.1007/s41060-017-0086-6.
[87] Martin Rosvall and Carl T. Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the USA* 105, 4 (January 2008), 1118-1123. DOI: 10.1073/pnas.0706851105.
[88] Aghiles Salah, Quoc-Tuan Truong and Hady W. Lauw. 2020. Cornac: A Comparative Framework for Multimodal Recommender Systems. *Journal of Machine Learning Research* 21, 95 (May 2020), 1-5.
[89] Javier Sanz-Cruzado. 2021. Contact recommendation in social networks: algorithmic challenges, diversity and network evolution. *PhD thesis*. Universidad Autónoma de Madrid.
[90] Javier Sanz-Cruzado and Pablo Castells. 2020. Beyond Accuracy in Link Prediction. In Boratto, L., Faralli, S., Marras, M., Stilo, G. (eds): *BIAS 2020: Bias and Social Aspects in Search and Recommendation.. Communications in Computer and Information Science*, vol 1245. Springer, Cham, 79-94. DOI: 10.1007/978-3-030-52485-2_9
[91] Javier Sanz-Cruzado, Pablo Castells, Craig Macdonald and Iadh Ounis. 2020. Effective Contact Recommendation in Social Networks by Adaptation of Information Retrieval Models. *Information Processing and Management* 57, 5 (September 2020), Article 102285. DOI: 10.1016/j.ipm.2020.102285.
[92] Javier Sanz-Cruzado and Pablo Castells. 2018. Contact Recommendations in Social Networks. In: I. Cantador, S. Berkovsky, D. Tikk (Eds.), *Collaborative Recommendations: Algorithms, Practical Challenges and Applications.* World Scientific Publishing, Singapore, 519-569. DOI: 10.1142/9789813275355_0016.
[93] Javier Sanz-Cruzado and Pablo Castells. 2018. Enhancing Structural Diversity in Social Networks by Recommending Weak Ties. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys 2018)*. ACM, Vancouver, BC, Canada, 233-241. DOI: 10.1145/3240323.3240371.
[94] Venu Satuluri, Yao Wu, Xun Zheng, Yilei Qian, Brian Wichers, Qieyun Dai, Gui Ming Tang, Jerry Jiang and Jimmy Lin. 2020. SimClusters: Community-Based Representations for Heterogeneous Recommendations at Twitter. In *Proceedings of the 26th ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD 2020)*. ACM, Online, 3183-3193. DOI: 10.1145/3394486.3403370.
[95] Stephen B. Seidman. 1983. Network structure and minimum degree. *Social Networks* 5, 3 (September 1983), 269-287. DOI: 10.1016/0378-8733(83)90028-X.
[96] Alina Sîrbu, Vittorio Loretto, Vito D.P. Servedio, Francesca Tria. 2017. Opinion Dynamics: Models, Extensions and External Effects. In V. Loretto, M. Haklay, V Servedio, G. Stumme, J. Theunis and F. Tria (eds.) *Participatory Sensing, Opinions and Collective Awareness*. Springer, Cham, 363-401. DOI: 10.1007/978-3-319-25658-0_17.
[97] Ana-Andreea Stoica, Christopher Riederer and Augustin Chaintreau. 2018. Algorithmic Glass Ceiling in Social Networks: The effects of social recommendations on network diversity. In *Proceedings of The Web Conference 2018 (WWW 2018)*. IW3C2, Lyon, France, 923-932. DOI: 10.1145/3178876.3186140.
[98] Jessica Su, Aneesh Sharma and Sharad Goel. 2016. The Effect of Recommendations on Network Structure. In *Proceedings of the 25th International Conference on World Wide Web (WWW 2016)*. IW3C2, Montréal, Québec, Canada, 1157-1167. DOI: 10.1145/2872427.2883040.
[99] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang and Cong Geng. 2020. Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. In *Proceedings of the 14th ACM conference on Recommender Systems (RecSys 2020)*. ACM, Online, 23-32. DOI: 10.1145/3383313.3412489.
[100] Jillian Tang, Xia Hu, Huan Liu. Social recommendation: a review. *Social Network Analysis and Mining* 3 (November 2013), 1113-1133. DOI: 10.1007/s13278-013-0141-9
[101] Saúl Vargas. 2015. Novelty and diversity evaluation and enhancement in recommender systems. *PhD Thesis*. Universidad Autónoma de Madrid.
[102] Saúl Vargas and Pablo Castells. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)*. ACM, Chicago, Illinois, 109-116. DOI: 10.1145/2043932.2043955.
[103] Duncan J. Watts and Steven Strogatz. 1998. Collective dynamics of 'small world' networks. *Nature* 393 (June 1998), 440-442. DOI: 10.1038/30918.
[104] Scott White and Padhraic Smyth. 2003. Algorithms for Estimating Relative Importance in Networks. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD 2003)*. ACM, Washington, DC, USA, 266-275. DOI: 10.1145/956750.956782.
[105] Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh and Deborah Estrin. 2018. OpenRec: a Modular Framework for Extensible and Adaptable Recommendation Algorithms. In *Proceedings of the 11th ACM international conference on Web Search and Data Mining (WSDM 2018)*. ACM, Marina del Rey, California, USA, 664-672. DOI: 10.1145/3159652.3159681.
[106] Reza Zafarani, Mohammad A. 2014. Abassi and Huan Liu. Social Media Mining: An Introduction. Cambridge University Press. DOI: 10.1017/CBO9781139088510.
[107] ChengXiang Zhai, William W. Cohen and John Lafferty. 2003. Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2003)*. ACM, Toronto, Canada, 10-17. DOI: 10.1145/860435.860440.
[108] Shuai Zhang, Yi Tai, Lina Yao, Bin Wu and Aixin Sun. 2019. DeepRec: An Open-Source Toolkit for Deep Learning based Recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*. IJCAI, Macao, China, 6581-6583. DOI: 10.24963/ijcai.2019/963.
[109] Jichang Zhao, Junjie Wu and Ke Xu. 2010. Weak ties: subtle role of information diffusion in online social networks. *Physical Review E* 82, 1 (July 2010). DOI: 10.1103/PhysRevE.82.016105