Effective Contact Recommendation in Social Networks by Adaptation of Information Retrieval Models

Javier Sanz-Cruzado[a,*], Pablo Castells[a], Craig Macdonald[b], Iadh Ounis[b]

[a] *Universidad Autónoma de Madrid, Escuela Politécnica Superior, C/Francisco Tomás y Valiente,11,28049, Madrid, Spain*
[b] *School of Computing Science, University of Glasgow, Lilybank Gardens, G12 8QQ, Glasgow, Scotland, United Kingdom*

Abstract

We investigate a novel perspective to the development of effective algorithms for contact recommendation in social networks, where the problem consists of automatically predicting people that a given user may wish or benefit from connecting to in the network. Specifically, we explore the connection between contact recommendation and the text information retrieval (IR) task, by investigating the adaptation of IR models (classical and supervised) for recommending people in social networks, using only the structure of these networks.

We first explore the use of adapted unsupervised IR models as direct standalone recommender systems. Seeking additional effectiveness enhancements, we further explore the use of IR models as neighbor selection methods, in place of common similarity measures, in user-based and item-based nearest-neighbors (kNN) collaborative filtering approaches. On top of this, we investigate the application of learning to rank approaches borrowed from text IR to achieve additional improvements.

We report thorough experiments over data obtained from Twitter and Facebook where we observe that IR models, particularly BM25, are competitive compared to state-of-the art contact recommendation methods. We provide further empirical analysis of the additional effectiveness that can be achieved by the integration of IR models into kNN and learning to rank schemes. Our research shows that the IR models are effective in three roles: as direct contact recommenders, as neighbor selectors in collaborative filtering and as samplers and features in learning to rank.

Keywords: social networks, contact recommendation, information retrieval models, k nearest neighbors, learning to rank, collaborative filtering

*Corresponding author

Email addresses: `javier.sanz-cruzado@uam.es` (Javier Sanz-Cruzado), `pablo.castells@uam.es` (Pablo Castells), `craig.macdonald@glasgow.ac.uk` (Craig Macdonald), `iadh.ounis@glasgow.ac.uk` (Iadh Ounis)

1. Introduction

The creation of online social network applications such as Twitter, Facebook and LinkedIn, and their subsequent expansion along the 2,000s has given rise to new perspectives and challenges in the information retrieval (IR) field and, as a particular case, in recommender systems. One of the most compelling problems in this area is recommending people with whom users might want to engage in an online network. The social nature of these networks, and the large amount of users accessing them every day has raised the need for contact recommendation in both industry [23, 26] and within the research communities [9, 27, 28]. The most prominent social platforms routinely offer user recommendation services since the end of the past decade, with systems such as 'Who-to-follow' on Twitter [23, 26] or 'People you may know' on Facebook and LinkedIn.

Contact recommendation represents a very particular perspective of the recommendation task. First, the recommendation domain lays connections to social network analysis and network science, with rich potential implications [27, 55]. Second, while in most domains, users and items are different objects, the task of contact recommendation has the peculiar and interesting characteristic that users and items are the same set. These particularities have motivated the creation of a wide variety of people recommendation algorithms from diverse fields, such as network science [35, 38], machine learning [29], recommender systems [28] and, to a lesser extent, information retrieval [28]. The confluence of recommendation and online social networks has also motivated the research of social item recommendation methods, which augment their input with network structures (in addition to the observed user-item interactions), while keeping a traditional item space (movies, songs, news, posts, purchases, etc.) as their output range for recommendation [20, 21, 60]. Different from this line of development, our present work specifically focuses on recommendation where the output consists of people in the network rather than any other different set of items.

In this context, our research investigates the relation between contact recommendation in social networks and text retrieval. As a canonical formulation of the problem, we consider recommendation based on the network structure only (i.e. not using side-information of any kind). Our research aims to make contributions at two levels: a) at a theoretical level, an enhanced understanding of the contact recommendation problem, through new connections to the information retrieval and recommender systems tasks, concepts and techniques, and b) at a practical level, opening new pathways for the development of new and effective contact recommendation methods by importing information retrieval models and algorithms. The adaptation of IR models to the recommendation task has been researched at a generic level [13, 47, 64, 65], but the specific adaptation to recommend people in social networks is a qualitatively different problem, as we see in Sections 4 and 5, and a largely unexplored direction.

For such a purpose, we establish associations between the fundamental elements involved in both tasks, in order to adapt classic IR models to the task of suggesting people in a social network. In particular, we explore the adaptation of a wide range of IR models in the task of contact recommendation, namely the vector space model [53] (VSM), BM25 [50], query likelihood [48] and the divergence from randomness models [4, 8]. We empirically compare the effectiveness of the resulting algorithms to state-of-the-art contact recommendation methods over data samples extracted from Twitter and Facebook. We find that the adapted IR models, particularly BM25, are competitive with the best alternative contact recommendation approaches.

We further close the effectiveness gap between the IR models and the best alternative contact recommendation approaches in terms of relevance metrics, by using the IR model adaptations in the role of similarity measures between users as part of a kNN collaborative filtering scheme. The dual role of users as candidate recommended "objects" or as "third-person" neighbors enables this new use of IR models as a component in a kNN construct rather than as a final recommendation method.

Finally, we study the use of supervised learning to rank techniques, originally found to be effective in text retrieval, and propose a framework to use them for the recommendation of social links. We then study the use of ensembles of IR-based models, achieving yet an additional improvement in terms of relevance over the best alternatives in our original comparison.

This paper continues previous work where we initially explored the adaptation of IR models as contact recommendation algorithms [56]. Our initial results showed that basic text IR models can be both effective and efficient as contact recommendation algorithms when applied to dynamic interaction networks. These results encouraged us to further explore this idea, and develop new people-to-people recommendation approaches based on IR. Beyond the original work, this article extends the previous research in several new directions:

- We extend the set of IR models adapted in previous work, adding divergence from randomness models [4, 8] as contact recommendation approaches, with competitive results.
- We propose the use of IR models in the role of similarity functions in a user-based and item-based kNN framework, obtaining further effectiveness improvements, which are quite substantial with respect to the IR models used as standalone recommendation algorithms.
- We design a framework for adapting supervised learning to rank IR techniques to contact recommendation, and we evaluate their effectiveness by building ensembles of IR models, deriving yet additional enhancements.
- We test the adaptation of IR models – including prior and present proposed approaches – on further types of social networks, including not only interaction networks, but also explicit follows networks extracted from Twitter and an undirected friendship network from Facebook, where we can study the behavior of different algorithms under varied conditions in terms of network properties such as density or clustering strength.

The core of our contribution is structured into four main blocks. A first block describes and motivates the parallels between text IR and contact recommendation, as well as shows the mappings between the two problems (Section 4). On top of this block, our specific contributions involve three main parts. First, we adapt a large set of IR models into standalone contact recommendation methods (Section 5). Second, we integrate the adapted IR models to play the role of similarity functions in kNN schemes (Section 7).Finally, we integrate the models as features in a learning to rank approach (Section 8). Each of these three parts includes its own block of experiments (Sections 6.4, 7.3, 8.3) – using the same social network datasets and evaluation approach (introduced in Section 6) – where the resulting methods are tested and their effectiveness is compared to that of state-of-the-art contact recommendation algorithms, as well as to each other. By and large, the empirical results in the three blocks show the achievement of an increasing longitudinal improvement in effectiveness. Each of these blocks is to a fair degree self-contained, and the reader can skip and/or jump to any of the parts while still being able to follow the paper's narrative.

2. Related Work

Before presenting our work, we provide a brief background of the different related areas: contact recommendation and link prediction, the relation between recommender systems and information retrieval, and learning to rank.

2.1. Contact Recommendation and Link Prediction

In the context of online social networks, social recommender systems [60] exploit the structures and traces in online social platforms to provide better recommendations to their users. As a particular case in this domain, contact recommendation (a.k.a. people-to-people recommendation) aims at identifying people in a social network that a given user would benefit from relating to [54]. This problem differs from most classical recommendation domains, such as movie, video and music streaming, or e-commerce recommendation, in the fact that users and items do not belong to different spaces: they live in the same one, and the candidate items are simply other users in the platform.

Contact recommendation functionalities have been present in the most popular social platforms since the beginning of the previous decade, with systems like Facebook or LinkedIn's 'People You May Know' [31] or Twitter's 'Who To Follow' [23, 26] functionalities. These services have been proven valuable for the growth and evolution of online social networks: for instance, in 2015, it was estimated that nearly one eighth of the new links in Twitter were created through their recommendation service [23].

The problem of recommending people to people in these environments has its foundation in a network science problem known as link prediction [35, 38]. Link prediction aims to accurately identify unobserved links that exist or will exist in the future in a network. The differences between both tasks are subtle. First, the domain of application of link prediction techniques is wider: these methods can apply to different network domains beyond social network platforms, such as citation networks [44], biological networks [17, 36], and many more, like, for example networks that represent flying routes between airports or high voltage lines between electrical generators and transformers [36]. The second difference consists in how the predicted links are ranked: in link prediction, all the possible links (or, at least, a fraction of them [44]) are ranked depending on the estimated probability of appearance in the network; however, in contact recommendation, a ranking for each different user in the network is created.

Link prediction and recommendation is an established topic at the confluence of social network analysis and recommender systems for which many methods have been proposed in the literature, based on the network topology [35], random walks across the network graph [9, 23, 26], user-generated content [28] or side-information such as location [63].

2.2. Relation between IR and Recommendation

The connections between recommendation and textual IR date back to the earliest recommender systems and their relation to the information filtering task [12]. Even though most of this connection has focused on content-based methods [2], it has also developed into collaborative filtering algorithms [13, 47, 61, 62, 64, 65].

A particularly representative and relevant approach for our work was developed by Bellogín et al. [13], allowing the adaptation of any IR term weighting scheme to create a collaborative filtering algorithm. To this end, the approach represents users and items in a common space, where users are the equivalent of queries, and items play the role of the documents to be retrieved. They proposed two different approaches. In the first one, both users and items are represented in the user space. Each item is represented by its ratings (the users who rated the item) and users are represented by their similarities to other users. In the second one, users and items are represented in the item space: users are represented by their ratings, and items are represented by their similarities. Earlier on, Wang et al. [64, 65], and later Parapar et al. [47], explored similar developments upon different – more specific – IR-inspired probabilistic models. Our work pursues a similar goal to this research direction, but taking a step further: if Bellogín et al., Wang et al. and Parapar et al. folded three spaces (terms, documents, queries) into two (users, items), we fold them into just one, as we later explain in Section 4.

Another relevant study for our work is the proposal by Valcarce et al. [62], who explored the use of query likelihood models [48] with different smoothing techniques as similarities for neighborhood-based collaborative filtering recommendations. They tested their approach in datasets from different domains and found that, indeed, using those similarities instead of others like cosine similarity improves both the accuracy and the diversity of the recommendations. In this paper, we adapt this work to the task of recommending people-to-people, and expand it by testing the effectiveness of multiple IR weighting schemes as similarities for both user-based and item-based neighborhood-based approaches.

Finally, there are already some established connections between the link prediction and contact recommendation tasks and and that of textual information retrieval. For example, some link prediction approaches like the Jaccard index [32, 35, 52] or cosine similarity [38, 52] have their roots in IR. More recently, Hannon et al. [28] adapted the vector space model [53] to recommend users on Twitter, based on both content-based and collaborative filtering algorithms. Our work seeks to extend, generalize and systematize this point of view to adapt any state-of-the-art IR model to contact recommendation.

2.3. Learning to rank

Learning to rank has become very popular in IR as a means to combine different sources of evidence as features within a learned model, appropriately weighted in a discriminative manner. In doing so, learning to rank aims to minimise a loss function that measures the error in producing an effective ranking, compared to the ground truth relevance judgments (labels) of a set of training documents for various queries.

Differently from regression or classification machine learning, learning to rank is characterized by the desire not to most accurately estimate the labels of the training data (as might be performed by a classifier or a regression), but instead to get the relative ordering of documents with the highest predicted labels at the top of the ranking. This matches better with the user expectations in getting the most relevant documents at the top of the ranking, as rewarded by classical IR evaluation metrics such as MAP, MRR (for binary labels), nDCG and ERR (for multi-graded relevance labels).

To this end, going beyond the traditional pointwise evaluation loss functions that represent classification and regression models, Liu [37] identified that pairwise loss functions (which consider the partial ordering of pairs of documents based on their labels), and listwise loss functions (which consider the overall ranking, usually based on a classical IR evaluation measure such as MAP and nDCG) were more effective.

In this work, we apply the LambdaMART [15, 67] learning to rank technique to contact recommendation. LambdaMART is characterized by the training of gradient boosted regression trees using a combination of pointwise loss with a listwise component that measures the improvement (or degradation) in a standard IR evaluation metric (typically nDCG) in changing the scores of pairs of documents of different labels. In this way, LambdaMART is a hybrid learning-to-rank technique, combining the infinitely flexible regression trees with listwise evaluation measures.

We now discuss the families of features typically deployed in a learning to rank scenario. While defined for each document, these families of features differ in terms of their dependence on the query and the document:

- *Query dependent:* These features differ for each document and for each query – typically, these might involve the score for a ranking model such as BM25 computed for the current query on the document's contents, or the application of BM25 on the title of a document.

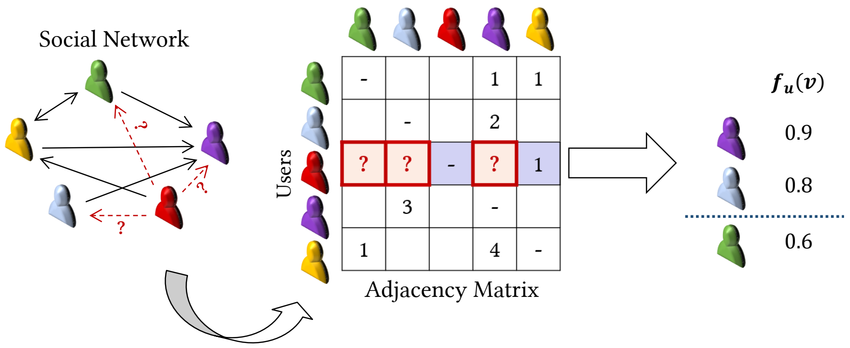


Figure 1: The contact recommendation task.

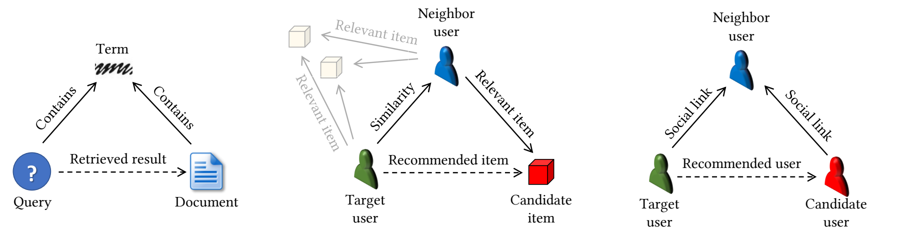
- Query independent: Such features vary between documents, but have the same value for each document, regardless of the query. Examples might be the PageRank of a web document, the length of a document, or its spamminess.
- *Query features:* These are independent of the document, in as much as each document for a given query carries the same value for a given query feature. Examples include query length, query performance predictors, or the query type detection (e.g. presence of entities, navigational vs. informational query).

In this work, we focus upon query dependent features within the context of contact recommendation. In Section 3, we define the task, while in Section 4 and 5 we discuss the adaptation of IR models to the task of contact recommendation. Later Section 8 particularly focuses on the application of learning to rank to the task of contact recommendation.

3. Preliminaries

We start by formally stating the contact recommendation task, and introducing the notations we use in the remainder of the article. We represent the structure of a social network as a graph $\mathcal{G} = \langle \mathcal{U}, E \rangle$, where $\mathcal{U}$ is the set of network users, and $E \in \mathcal{U}_*^2$ is the set of relations between users (friendship, interactions, or whatever the network is representing), where $\mathcal{U}_*^2 = \{(u,v) \in \mathcal{U}^2 | u \neq v\}$ is the set of pairs formed by different users.

For each user $u \in \mathcal{U}$, we denote their neighborhood as $\Gamma(u)$ (the set of users that u has established relations with). In directed networks, three different neighborhoods can be considered, depending on which orientation we choose for selecting the neighbors: the incoming neighborhood $\Gamma_{in}(u)$ (users who create links towards u), the outgoing neighborhood $\Gamma_{out}(u)$ (users towards whom u creates links), and the union of both neighborhoods $\Gamma_{und}(u)$. In weighted graphs, we have additionally a weight function $w : \mathcal{U}_*^2 \to \mathbb{R}$, which returns the



(a) Three spaces in text retrieval (tripartite graph)

(b) Two spaces in item recommendation (bipartite graph)

(c) Single space in contact recommendation (unipartite graph)

Figure 2: Textual IR elements (a) vs. item recommendation elements (b) vs. contact recommendation elements (c).

weight of an edge if $(u, v) \in E$, and 0 otherwise. In unweighted graphs, we can consider that $w(u, v) = 1$ if the link exists, and 0 otherwise.

Next, given a target user u, the contact recommendation task consists in finding a subset of users $\hat{\Gamma}_{out}(u) \subset \mathcal{U} \setminus \Gamma_{out}(u)$ towards whom u has no links but who might be of interest to the target user. We address the recommendation task as a ranking problem, in which we find a fixed number of users $n = |\hat{\Gamma}_{out}(u)|$ sorted by decreasing value of a ranking function $f_u : \mathcal{U} \setminus \Gamma_{out}(u) \to \mathbb{R}$. Figure 1 illustrates the contact recommendation task thus viewed under the perspective of a conventional recommendation task.

4. IR Model Adaptation Framework for Contact Recommendation

Even though recommendation and text retrieval have been traditionally addressed as separate problems, it is possible to establish analogies and equivalences between both tasks. Recommender systems are indeed often described as retrieval systems where the query is absent, with the records of user activity being available instead [13]. The approaches we develop follow this perspective.

In order to adapt textual IR models to the recommendation task, we need to establish equivalences between the elements in the contact recommendation task (users and interactions between them) and the spaces involved in text search (queries, documents and terms). In previous adaptations of IR models for recommendation, the three IR spaces commonly folded into two: the set of users and the set of items [13]. However, when we seek to recommend people in social networks, the latter two spaces are the same. Therefore, to adapt the IR models to our task, we fold the three IR spaces into a single dimension: the set of users in the social network, playing three different roles, as we illustrate in Figure 2. Next, we explain in more detail how we carry this mapping through.

First, the natural equivalent of documents in the search space are candidate users (to be recommended as contacts), as they play the same role: they are the elements to be retrieved in order to fulfil a user need. The information need is explicit in the search task, expressed by a query. However, it is implicit in

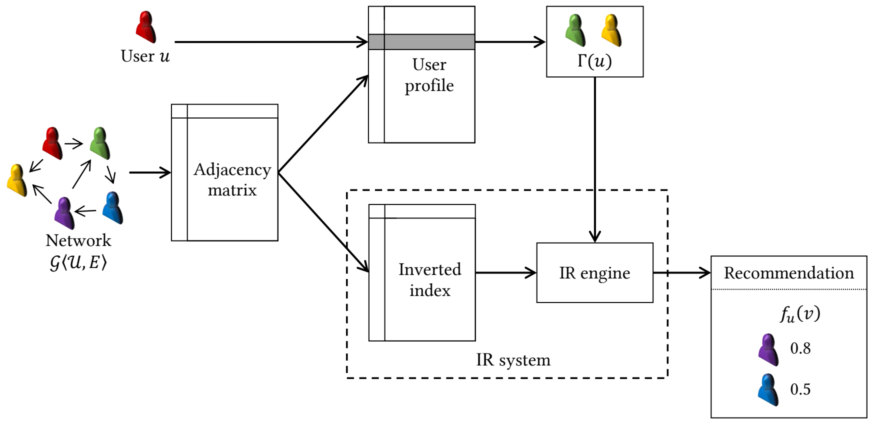


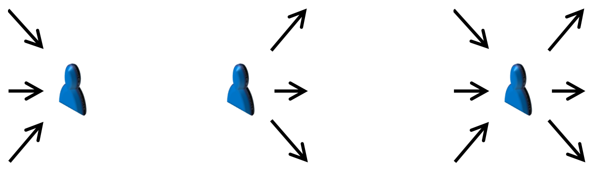
Figure 3: Adaptation of IR models to recommend users in social networks.

contact recommendation: the need for creating new bonds. This social need is to be predicted based on records of past user activity, which therefore play an equivalent role to the query keywords in textual IR. In a social network, past user activity is encoded into existing links to and/or from the target user.

Finally, we need an equivalent to the term representation of documents. In prior adaptations of IR models for recommendation, this was the main difficulty: users and items were different objects, so a representation that suits one might not work for the other [13]. In contact recommendation this becomes in fact easier: users and items are the same thing, so any term representation for target users is automatically valid for the "items" (the candidate users). The possibilities for defining an equivalent to terms are manifold, and result in very different algorithms. For instance we can define content-based recommendation methods by using texts associated to users, such as messages or documents posted (or liked) by the users [28]. On the other hand, if we take users as the term space, and we equate the term-document relationship to interactions between users, we obtain collaborative filtering algorithms. In this article, we focus on the latter approach.

Figure 3 illustrates our proposed collaborative filtering adaptation framework. A social network is encoded as a weighted adjacency matrix A, where $A_{uv} = w(u, v)$. Using link data, we build two elements: on one hand, an inverted index that allows for the fast retrieval of candidate users and, on the other, a structure that provides direct access to the neighborhood of the target users, i.e. the query term representation. The inverted index uses network users as keys (playing the role of terms), and the postings lists store the set of candidate users to whose neighborhood representation (as "documents") the "key" users belong to.

Using this index and the "query" structure, any textual IR system can be used as a contact recommendation algorithm. Those new algorithms are part of a family of methods known as "friends of friends", which recommend people



(a) Incoming neighborhood Γ_{in} (b) Outgoing neighborhood Γ_{out} (c) Undirected neighborhood Γ_{und}

Figure 4: Possible neighborhood orientations.

with whom their neighbors have a relation. In addition to our IR approaches, this family includes some link prediction approaches like Adamic-Adar [1, 35], Jaccard similarity [32, 35] or simply recommending users by the greatest number of common neighbors between the candidate and the target user [35]. Additional details and options remain open however when developing a specific instance of this framework in full detail, as we describe in the following sections. An important aspect concerns the direction of social links in the re-interpretation of IR models, to which we pay specific attention.

In directed social networks such as Twitter or Instagram, three definitions of user neighborhood can be considered, as illustrated in Figure 4: the incoming neighborhood $\Gamma_{in}(u)$, the outgoing neighborhood $\Gamma_{out}(u)$ and the union of both, $\Gamma_{und}(u) = \Gamma_{in}(u) \cup \Gamma_{out}(u)$. Any of the three options is valid in our adaptation of IR models. Since the inverted index and user profiles are created independently, it is even possible to take a different choice for target and candidate users: since we still use the same elements to represent (the equivalent of) both queries and documents, it is possible to work just smoothly with different neighborhood orientation choices for user targets and candidates [56].

5. Adaptation of Specific IR Models

As examples of the general unification framework we proposed in the previous section, we show in some detail the adaptation of the IR models we use in our experiments . We divide the IR models in several categories, according to the typical classification in IR: probability ranking principle models [49, 50], language models [48], divergence from randomness models [4, 8] and the vector space model [53]. In the formulations presented in this section, we denote the neighborhood representation of the target user as $\Gamma^q(u)$ and the neighborhood representation of candidate users as $\Gamma^d(v)$. Table 1 provides a summary of the notations used in the remainder of this article, as well as the relation between the IR and the contact recommendation elements. Further details about the notations included in the table are provided in this section. In addition, Table 2 summarizes the resulting formulations of the IR models for content recommendation.

Table 1: Relation between the IR and contact recommendation elements.

Information retrieval	Contact recommendation
Document collection, D	Set of users, $\mathcal{U}$
Query, q	Target user's neighborhood, $\Gamma^q(u)$
Document, d	Candidate user's neighborhood, $\Gamma^d(u)$
Term $t \in q/d$	Neighbor user, $t \in \Gamma^q(u)/\Gamma^d(v)$
Documents containing a term, D_t	User's inverse neighborhood, $\Gamma^d_{inv}(t)$
Frequency of a term, $\text{freq}(t, d)$	Weight of a link, $w^d(v, t)$
Document length, $\lvert d' \rvert$	Length of the user, $\text{len}^l(v)$

5.1. Probability Ranking Principle (PRP) Models

The probability ranking principle introduced by Robertson [49] establishes that, given a query, ranking documents according to their probability of relevance maximizes the expected effectiveness of the returned results. We study two models, namely BIR and BM25 [50] and propose a new one, Extreme BM25, based on the second model.

5.1.1. Binary Independence Retrieval (BIR)

The model known as BIR (binary independence retrieval) [50] is the simplest representative of IR models building on the probability ranking principle [49]. Under the assumption that term occurrence follows a (multiple) Bernoulli distribution, this model estimates the probability of relevance of a document d for a query q as:

$$P(r|d,q) \propto \sum_{t \in d \cap q} \text{RSJ}(t) \tag{1}$$

where r denotes the event that the document is relevant, and RSJ is defined as follows [50]:

$$\text{RSJ}(t) = \log \frac{|R_t|(|D| - |D_t| - |R| - |R_t|)}{(|R| - |R_t|)(|D| - |D_t|)} \tag{2}$$

In the above equation R is the set of relevant documents for the query, R_t is the set of relevant documents containing the term t, D is the document collection, and D_t is the set of documents containing t. Since the set R of relevant documents is not known, the following approximation can be taken, considering that typically only a tiny fraction of documents are relevant:

$$\text{RSJ}(t) = \log \frac{|D| - |D_t| + 0.5}{|D_t| + 0.5} \tag{3}$$

As described in Section 4, to adapt this model for contact recommendation, we equate queries and documents to target and candidate users, respectively. We also equate the term-document relationship to social network edges. Under this equivalence, $|D|$ is the number of users in the network, and $|D_t|$ is the number of users that t is a neighbor of (i.e. their neighbor size in the transposed network). Denoting inverse neighborhoods as $\Gamma^d_{inv}(t)$, the adapted BIR equation becomes as follows:

$$f_u(v) = \sum_{t\in\Gamma^q(u)\cap\Gamma^d(v)} \mathrm{RSJ}(w) = \sum_{t\in\Gamma^q(u)\cap\Gamma^d(v)} \log \frac{|\mathcal{U}| - |\Gamma^d_{inv}(t)| + 0.5}{|\Gamma^d_{inv}| + 0.5} \tag{4}$$

5.1.2. BM25

BM25 is one of the best-known and most effective probabilistic IR models [50]. It starts from similar principles to BIR, but modeling term occurrence in documents as a Poisson instead of a Bernoulli distribution. The resulting ranking function is defined as:

$$P(r|d,q) \propto \sum_{t\in d\cap q} \frac{(k+1)\mathrm{freq}(t,d)}{k\left(1-b+b\frac{|d|}{\mathrm{avg}_{d'\in D}|d'|}\right)+\mathrm{freq}(t,d)} \mathrm{RSJ}(t) \tag{5}$$

where $\mathrm{freq}(t,d)$ denotes the frequency of t in d,$|d|$ is the document length, $\mathrm{RSJ}(w)$ is defined in Equation (3), and $k = [0,\infty)$ and $b \in [0,1]$ are free parameters controlling the effect of term frequencies and the influence of the document length, respectively.

The text retrieval space can be mapped into a social network just as before, now taking, additionally, edge weights as the equivalent of term frequency. In directed networks, we also need to make a choice between the weight of incoming or outgoing links as the equivalent of frequency. We link this decision to the edge orientation selected for candidate users (as pointed out earlier in Sections 4 and the beginning of Section 5), as follows:

$$\mathrm{freq}(t,v) = w^d(v,t) = \begin{cases} w(t,v) & \text{if } \Gamma^d := \Gamma_{in} \\ w(v,t) & \text{if } \Gamma^d := \Gamma_{out} \\ w(t,v)+w(v,t) & \text{otherwise} \end{cases} \tag{6}$$

Finally, document length can be now defined as the sum of edge weights of the candidate user. In unweighted graphs, this is simply equivalent to the degree of the node. On the other hand, in directed networks, we have again different choices. The BM25 formulation for text retrieval considers different options in defining document length (number of unique terms, sum of frequencies, etc.) [50]. We find similarly worthwhile to decouple the orientation choice for document length from the one for the term representation of candidate users. We reflect this by defining length as follows:

$$\mathrm{len}^l(v) = \sum_{t\in\Gamma^l(v)} w^l(v,t) \tag{7}$$

where $\Gamma^l(v)$ represents the candidate's neighborhood in a specific orientation choice for document length. Based on this, the adaptation of BM25 becomes:

$$f_u(v) = \sum_{t\in\Gamma^q(u)\cap\Gamma^d(v)} \frac{(k+1)w^d(v,t)}{k\left(1-b+b\frac{\mathrm{len}^l(v)}{\mathrm{avg}_{x\in\mathcal{U}}\mathrm{len}^l(x)}\right)+w^d(v,t)} \mathrm{RSJ}(t) \tag{8}$$

5.1.3. Extreme BM25

In addition to the previous probabilistic models, we introduce a new one, which we denote by Extreme BM25. This algorithm is defined as the BM25 algorithm when the parameter k tends to ∞. Its equation is the following:

$$f_u(v) = \sum_{t \in \Gamma^q(u) \cap \Gamma^d(v)} \frac{w^d(v,t)}{1 - b + b \frac{\mathrm{len}^l(v)}{\mathrm{avg}_{x \in \mathcal{U}} \mathrm{len}^l(x)}} \mathrm{RSJ}(t) \tag{9}$$

The main difference between BM25 and Extreme BM25 is the removal of the smoothing of the numerator.

5.2. Language Models

Another family of probabilistic IR approaches is the family of language models. A language model is a probability distribution over linguistic units, such as words, sentences or whole documents. In these models, a document is relevant to a query if the document model is likely to generate the query. One of the most widely used and successful language modelling approaches to IR is the so-called query likelihood (QL) model [48]:

$$f_q(d) = p(q|d) \propto \sum_{t \in q} \mathrm{freq}(t,q) \log_2(p(t|d)) \tag{10}$$

In the contact recommendation task, this model translates to:

$$f_u(v) = p(u|v) \propto \sum_{t \in \Gamma^q(u)} w^q(u,t) \log_2 p(t|v) \tag{11}$$

where probability $p(t|v)$ can be estimated by maximum likelihood with some smoothing to avoid the whole score of a "document" user to vanish only because a single "query" user neighbor does not appear in the document. We explore three different smoothing techniques, which have been previously applied in recommender systems [13, 61, 62]:

- The Jelinek-Mercer smoothing (QLJM): [33]

$$f_u(v) = \sum_{t \in \Gamma^q(u)} w^q(u,t) \log \left((1-\lambda) \frac{w^d(v,t)}{\mathrm{len}^d(v)} + \lambda \frac{\mathrm{len}^d_{inv}(t)}{\sum_{x \in \mathcal{U}} \mathrm{len}^d(x)} \right) \tag{12}$$

- The Dirichlet smoothing (QLD) [42]:

$$f_u(v) = \sum_{t \in \Gamma^q(u)} w^q(u,t) \log \left(\frac{w^d(v,t) + \mu \cdot \mathrm{len}^d_{inv}(t) / \sum_{x \in \mathcal{U}} \mathrm{len}^d(x)}{\mathrm{len}^d(v) + \mu} \right) \tag{13}$$

- The Laplace smoothing (QLL) [62]:

$$f_u(v) = \sum_{t \in \Gamma^q(u)} w^q(u,t) \log \left(\frac{w^d(v,t) + \gamma}{\mathrm{len}^d(v) + \gamma |\mathcal{U}|} \right) \tag{14}$$

5.3. Divergence from Randomness (DFR) Models

The divergence from randomness (DFR) models are a family of probabilistic IR models, based on the idea that, given a term t and a document d, the more the distribution of the term in the document diverges from the distribution of the term in the whole collection, the more informative the term is for that document [4] . The DFR family of algorithms is very wide, and we focus here on five popular DFR algorithms: PL2, DFRee, DFReeKLIM, DLH and DPH. All these DFR models are available in the Terrier IR platform [39, 46].

5.3.1. PL2

PL2 [4, 8] is one of the best-known DFR models. This model is characterized by three different elements. First, the divergence between the distribution of the terms in each individual document and the collection is measured using a Poisson distribution. Second, a Laplace aftereffect estimation is applied as a first normalization of the model and, last, the term frequency is normalized using the so-called Normalisation 2 [4, 8]. As a contact recommendation algorithm, this leads to the following formulation:

$$f_u(v) = \sum_{t \in \Gamma^q(u)} \frac{w^q(u,t)}{\hat{w}^d(v,t)+1} \left[\hat{w}^d(v,t) \log_2 \frac{|\mathcal{U}|\hat{w}^d(v,t)}{\sum_{x\in\mathcal{U}} w^d(x,t)} + \frac{\log_2(2\pi\hat{w}^d(v,t))}{2} \right. \\ \left. + \left(\frac{\sum_{x\in\mathcal{U}} w^d(x,t)}{|\mathcal{U}|} + \frac{1}{12\hat{w}^d(v,t)} - \hat{w}^d(v,t) \right) \log_2 e \right] \tag{15}$$

where $\hat{w}^d(v,t)$ is the weight of the link after applying Normalisation 2:

$$\hat{w}^d(v,t) = w^d(v,t) \log_2 \left(1 + c \cdot \frac{\operatorname{avg}_{x\in\mathcal{U}} \operatorname{len}^d(x)}{\operatorname{len}^d(v)} \right) \tag{16}$$

5.3.2. DFRee

DFRee stands for DFR free from parameters. Instead of using a probability distribution to measure the divergence, this model applies concepts from information theory [59]. Specifically, given a term in the query, this models uses as measure of divergence the average number of extra bits, which would be needed to code the document if we added an extra appearance of the term. As there are two possible ways of sampling the distribution (considering only the document or the complete collection), this method computes the average of both information measures (i.e. their inner product). When adapted to contact recommendation, the formulation for this model is the following:

$$f_u(v) = \sum_{t \in \Gamma^q(u)} \left[w^d(v,t) \log \frac{\hat{p}(t,v)}{p(t)} + (w^d(v,t)+1) \log_2 \frac{\hat{p}^+(t,v)}{p(t)} + \right. \\ \left. + \frac{1}{2} \log \frac{\hat{p}^+(t,v)}{\hat{p}(t,v)} \right] w^q(u,t) w^d(v,t) \log \frac{\hat{p}^+(v,t)}{\hat{p}(v,t)} \tag{17}$$

where

$$p(t) = \frac{\sum_{x\in\mathcal{U}} w^d(x,t)}{\sum_{x\in\mathcal{U}} \text{len}^d(x)} \tag{18}$$

represents the prior probability of the term in the collection,

$$\hat{p}(v,t) = \frac{w^d(v,t)}{\text{len}^d(v)} \tag{19}$$

represents the relative frequency of the term in the “document” user and

$$\hat{p}^+(v,t) = \frac{w^d(v,t)+1}{\text{len}^d(v)+1} \tag{20}$$

represents the relative frequency of the term (adding 1 to the value of the weight).

5.3.3. DFReeKLIM

This DFR model has been specifically proposed for dealing with short documents such as tweets. Similarly to the DFRee weighting model, this parameter-free DFR approach uses information theory concepts to compute the divergence [7]. This model is computed as the inner product of two information measures: the additional cost, in bits, of coding the document this term belongs to, when considering the probability $\hat{p}(v,t)$, and the cost of adding this term to the document with respect to the optimal encoding of the document. Using the definitions provided for DFRee, the mathematical formulation of the approach is given as follows:

$$f_u(v) = \text{len}^d(v) \sum_{t\in\Gamma^q(u)} w^q(u,t)\hat{p}(v,t)\left[\log\frac{\hat{p}(v,t)}{p(t)}\log\frac{\hat{p}^+(v,t)}{\hat{p}(v,t)}\right] \tag{21}$$

5.3.4. DPH and DLH

Both DLH [5] and DPH [6] are parameter-free DFR models that use a hypergeometric distribution as the divergence measure. They differ in their normalization scheme: whereas DLH uses Laplace normalization, DPH uses that of Popper. The equations for those methods as recommendation approaches can then be formulated as follows:

$$f_u(v) = \sum_{t\in\Gamma^q(u)} w^q(u,t)\frac{w^d(v,t)\log\frac{\hat{p}(v,t)}{p(t)} + 0.5\log\left(2\pi w^d(v,t)(1-\hat{p}(v,t))\right)}{w^d(v,t)+1} \tag{22}$$

where $p(t)$ and $\hat{p}(v,t)$ are defined in Equations (18) and (19) for DLH and

$$f_u(v) = \sum_{t\in\Gamma^q(u)} \left(1 - \frac{w^d(v,t)}{\text{len}^d(v)}\right)^2 \left[w^d(v,t)\log\left(\frac{w^d(v,t)\sum_{x\in\mathcal{U}}\text{len}^d(x)}{\text{len}^d(v)}\right) + \frac{1}{2}\log\left(2\pi w^d(v,t)\right)\left(1 - \frac{w^d(v,t)}{\text{len}^d(v)}\right)\right] \frac{w^q(u,t)}{w^d(v,t)+1} \tag{23}$$

for the DPH model.

5.4. Vector Space Model

In the vector space model (VSM) [53] documents and queries are represented as $|\mathcal{V}|$-dimensional vectors, where $\mathcal{V}$ represents the set of terms in the collection. Using this model, the ranking function is often the cosine similarity of the query and document vectors. In the case of contact recommendation, the cosine similarity of the target and candidate user vectors can be formulated as follows:

$$f_u(v) = \sum_{t\in\Gamma^q(u)\cap\Gamma^d(v)} \frac{u_t v_t}{\sqrt{\sum_{t\in\Gamma^d(v)} v_t^2}} \tag{24}$$

where u_t and v_t represent the coordinates of the user vectors for term t. This is defined as the product of two separate terms: the term frequency (tf), which is a function of the number of appearances of the term in the document, and the inverse document frequency (idf), which, similarly to RSJ in the BM25 model, gives more value to discriminative terms. In this article, we use the following simple and common tf-idf definition [10]:

$$u_t = \text{tf-idf}(u,t) = (1 + \log_2 w^q(u,t)) \cdot \log_2\left(1 + \frac{|\mathcal{U}|}{1 + |\Gamma^q_{inv}(t)|}\right) \tag{25}$$

$$v_t = \text{tf-idf}(v,t) = (1 + \log_2 w^d(u,t)) \cdot \log_2\left(1 + \frac{|\mathcal{U}|}{1 + |\Gamma^d_{inv}(t)|}\right) \tag{26}$$

where we add 1 to the denominators in the idf expressions to avoid division by zero when a user has no neighbors (which might happen in partial directed network samples, as opposed to text collections where all terms are supposed to occur in at least one document).

6. Experiments with Standalone IR Models

In order to analyze the performance of our adaptation of the IR methods to contact recommendation and compare them with baseline alternatives, we conduct several offline experiments using social network data extracted from Twitter and Facebook. In the following, we describe the data, as well as our experimental approach and setup.[1]

[1]All the source code used in the experiments reported here is available at https://github.com/ir-uam/IR-models-4-contact-recommendation.

Table 2: Adaptation of IR models for recommendation

Probability Ranking Principle (PRP)

Algorithm	Equation $f_u(v)$
BIR	$\sum_{t\in\Gamma^q(u)\cap\Gamma^d(v)} \log \frac{\lvert\mathcal{U}\rvert - \lvert\Gamma^d_{inv}(t)\rvert + 0.5}{\lvert\Gamma^d_{inv}\rvert + 0.5}$
BM25	$\sum_{t\in\Gamma^q(u)\cap\Gamma^d(v)} \frac{(k+1)w^d(v,t)\mathrm{RSJ}(t)}{k\left(1-b+b\left(\mathrm{len}^l(v)/\operatorname{avg}_x \mathrm{len}^l(x)\right)\right)+w^d(v,t)}$
Extreme BM25	$\sum_{t\in\Gamma^q(u)\cap\Gamma^d(v)} \frac{w^d(v,t)}{1-b+b\left(\mathrm{len}^l(v)/\operatorname{avg}_x \mathrm{len}^l(x)\right)}\mathrm{RSJ}(t)$

Language models – Query likelihood (QL)

Algorithm	Equation $f_u(v)$
QLD	$\sum_{t\in\Gamma^q(v)} w^q(u,t)\log\left((1-\lambda)\frac{w^d(v,t)}{\mathrm{len}^d(v)}+\lambda\frac{\mathrm{len}^d_{inv}(t)}{\sum_{x\in\mathcal{U}}\mathrm{len}^d(x)}\right)$
QLJM	$\sum_{t\in\Gamma^q(v)} w^q(u,t)\log\left((1-\lambda)\frac{w^d(v,t)}{\mathrm{len}^d(v)}+\lambda\frac{\mathrm{len}^d_{inv}(t)}{\sum_{x\in\mathcal{U}}\mathrm{len}^d(x)}\right)$
QLL	$\sum_{t\in\Gamma^q(u)} w^q(u,t)\log\left(\frac{w^d(v,t)+\gamma}{\mathrm{len}^d(v)+\gamma\lvert\mathcal{U}\rvert}\right)$

Divergence from Randomness (DFR)

Algorithm	Equation $f_u(v)$
PL2	$\sum_{t\in\Gamma^q(u)} \frac{w^q(u,t)}{\hat{w}^d(v,t)+1}\left[\hat{w}^d(v,t)\log_2\left(\frac{\lvert\mathcal{U}\rvert\hat{w}^d(v,t)}{\sum_{x\in\mathcal{U}}w^d(x,t)}\right)+\frac{\log_2(2\pi\hat{w}^d(v,t))}{2} + \left(\frac{\sum_{x\in\mathcal{U}}w^d(x,t)}{\lvert\mathcal{U}\rvert}+\frac{1}{12\hat{w}^d(v,t)}-\hat{w}^d(v,t)\right)\log_2 e\right]$
DFRee	$\sum_{t\in\Gamma^q(u)} w^q(u,t)w^d(v,t)\log\frac{\hat{p}^+(v,t)}{\hat{p}(v,t)}\left[w^d(v,t)\log\frac{\hat{p}(t,v)}{p(t)}+ +(w^d(v,t)+1)\log_2\frac{\hat{p}^+(t,v)}{p(t)}++\frac{1}{2}\log\frac{\hat{p}^+(t,v)}{\hat{p}(t,v)}\right]$
DFReeKLIM	$\sum_{t\in\Gamma^q(u)} w^q(u,t)\hat{p}(v,t)\left[\log\frac{\hat{p}(v,t)}{p(t)}\log\frac{\hat{p}^+(v,t)}{\hat{p}(v,t)}\right]\mathrm{len}^d(v)$
DLH	$\sum_{t\in\Gamma^q(u)} w^q(v,t)\frac{w^d(v,t)\log\frac{\hat{p}(v,t)}{p(t)}+0.5\log\left(2\pi w^d(v,t)(1-\hat{p}(v,t))\right)}{w^d(v,t)+1}$
DPH	$\sum_{t\in\Gamma^q(u)} \left(1-\frac{w^d(v,t)}{\mathrm{len}^d(v)}\right)^2\left[w^d(v,t)\log\left(\frac{w^d(v,t)\sum_{x\in\mathcal{U}}\mathrm{len}^d(x)}{\mathrm{len}^d(v)}\right)+ \frac{1}{2}\log\left(2\pi w^d(v,t)\right)\left(1-\frac{w^d(v,t)}{\mathrm{len}^d(v)}\right)\right]\frac{w^q(u,t)}{w^d(v,t)+1}$

Vector Space Model (VSM)

Algorithm	Equation $f_u(v)$
VSM	$\sum_{t\in\Gamma^q(u)\cap\Gamma^d(v)} \frac{u_t v_t}{\sqrt{\sum_{t\in\Gamma^d(u)} v_t^2}}$

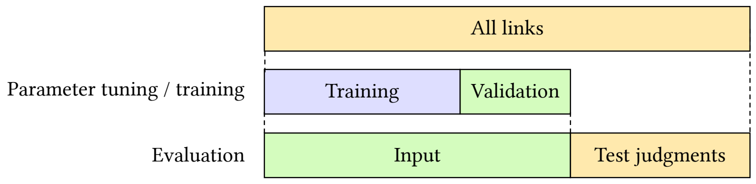


Figure 5: The random and temporal network partitioning approaches.

6.1. Data

We run our experiments over graphs collected from two of the largest social networking sites: Facebook and Twitter. The Facebook data was taken from the Stanford Large Network Dataset collection[2]. In particular, we use the ego-Facebook dataset, formed by gathering all the edges between the friends of ten users. This network has 4,039 users and 88,234 edges in total [43].

For our Twitter data, we use two types of networks: dynamic, implicit networks, induced by the interactions between users (i.e. $(u, v) \in E$ if u retweeted or mentioned v) and explicit networks, formed by static follow links (i.e. $(u, v) \in E$ if u follows v). To collect these graphs, we first sample the implicit networks using a snowball crawling approach [24]: we start with a single seed user, and we take the interaction tweets (retweets and mentions) by each user as outgoing network edges to be traversed. User sampling stops when 10,000 users are reached in the traversal; at that point, any outgoing edges from the remaining users in the crawl frontier pointing to the sampled users are added to the network. For the follows graph, we take the collected users and retrieve all the follow relationships between them.

Using the previous procedure, we built two different datasets: one containing all tweets posted by a set of around 10,000 users from June 19th to July 19th 2015, and one containing the last 200 tweets (the maximum number of tweets that can be collected in a single Twitter API call) posted by 10,000 users as of August 2nd 2015. We denote the first dataset as *"1 month"* whereas the second is denoted as *"200 tweets"*. Parts of those datasets were previously used in several works [54, 55, 56, 57].

6.2. Experimental Procedure

For training and evaluation purposes, given a network dataset (or data source), we handle three subnetworks, that is, three disjoint sets of edges: **training**, **validation** and **test**, which can be obtained in different manners. In all cases, the hyperparameters (IR model parameters, neighborhood orientation, etc.) are tuned using the training set as input, and the validation set as relevance judgments. After this, the union of the training and validation sets – to which we shall henceforth refer to as the input network – is supplied as input

[2]https://snap.stanford.edu/data (Accessed July 18 2019)

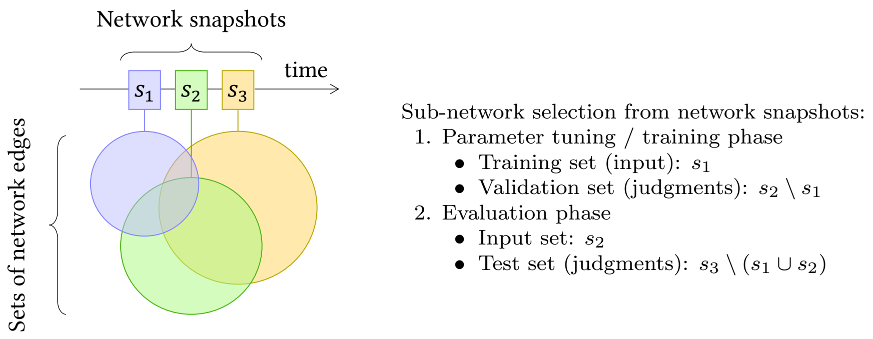

Network snapshots
time
s_1
s_2
s_3
Sets of network edges
Sub-network selection from network snapshots:
1. Parameter tuning / training phase
• Training set (input): s_1
• Validation set (judgments): $s_2 \setminus s_1$
2. Evaluation phase
• Input set: s_2
• Test set (judgments): $s_3 \setminus (s_1 \cup s_2)$

Figure 6: The snapshot-based network partitioning approach. The procedure is based on network growth over time, but also considers that some edges may disappear (i.e. we do not necessarily have $s_1 \subset s_2 \subset s_3$).

Table 3: Dataset statistics. The clustering coefficient is computed over the union of all edge sets for each network, and ignoring the edge direction.

	Twitter 1-month		Twitter 200-tweets		Facebook
	Interactions	Follows	Interactions	Follows	
Directed	Yes	Yes	Yes	Yes	No
Clustering coefficient	0.065	0.116	0.050	0.123	0.519
# Users	9,528	9,770	9,985	9,964	4,039
# Training edges	116,332	630,504	104,866	427,568	56,466
# Validation edges	33,867	46,628	29,131	46,760	14,100
# Input edges	170,425	645,022	137,850	475,730	70,566
# Test edges	54,335	81,110	21,598	98,519	17,643

to the recommendation algorithms, and the test graph is held out from the algorithms for their evaluation. IR metrics such as precision, recall or nDCG [10] can be computed on the output of a recommendation algorithm by considering the test edges as binary relevance judgments: a user v is relevant to a user u if – and only if – the edge (u, v) appears in the test graph.

We use three different approaches to obtain these three sets: random split, temporal split, and temporal snapshots.

- **Random split:** We start from a single given network (a dataset), and we simply split the set of edges into three disjoint sets uniformly at random based on the desired split ratios.

- **Temporal split:** We divide the network into the desired ratios, ensuring that the timestamps of the training set precede the validation set, and that the latter also precedes the test set. If any edges appear in more than one subset (e.g. when two users interact repeatedly at different times across a split point), they are removed from the most recent ones, to avoid "test contamination" with training data.

- **Temporal snapshots:** We take three snapshots at three consecutive time points. The first snapshot is taken as the training graph; the new links in the second snapshot (i.e. the second snapshot minus the first) are used as the validation set; and the new edges in the third download (i.e. the third download minus the second and the first) are used as the test data.

The random and temporal splits are illustrated in Figure 5, while the temporal snapshots are illustrated in Figure 6. We now explain what data partitioning approach we apply on each of the specific datasets we use in our experiments. Table 3 summarizes the statistics and properties of the different datasets and the resulting subgraphs.

For the Facebook network, since we do not have temporal information, we split the graphs randomly: we take 60% of the links as training, 20% for validation, and 20% in the test set.

In our Twitter interaction networks, we apply a temporal split, which better represents a real setting. In these graphs, the frequency of training interaction between every pair of users is available to the evaluated systems as part of the training information. The split points for the 1-month dataset are July 5^{th} 0:00:00 GMT and July 12^{th} 2015 0:00:00 GMT, thus taking two weeks for training, one week for validation, and one week for testing. In the 200-tweets dataset, the split dates are July 24^{th} 17:15:26 GMT and July 29^{th} 2015 1:08:07 GMT chosen to have 60%, 20% and 20% of interactions in the training, validation and test graph, respectively. After removing repeated edges crossing the network partition, the final size of the validation and test subsets may slightly decrease from the initial split ratio – in our case the difference, when any, is quite minor, as can be verified in Table 3.

For the follows networks, since Twitter does not provide information about the times when links were created, we apply the snapshot approach: we download the links between the users in the dataset three times. For the 1-month dataset, the first download of the follows graph contains the links created in the network before October 9^{th} 2015, and for 200-tweets, the connections between users as of October 20^{th} 2015. For both datasets, the second snapshot is taken four months after the first one, and the third snapshot two years after the second.

Note that given the way our experiments are designed, the task of the evaluated systems is, in a strict sense, to predict what the users will do next (follow or interact with other people). Inasmuch as the observed user's own actions can be assumed to be useful for them, these experiments provide a fair proxy for evaluating how useful the recommendations could be if they were delivered as suggestions to the users, which is a common perspective in evaluation practice in the field [25, 54].

6.3. Recommendation Algorithms

We assess the proposed IR model adaptations by comparing them to a selection of the most effective and representative algorithms in the link prediction and contact recommendation literature. Our selection includes Adamic-Adar

[1], most common neighbors (MCN) [35], Jaccard [32, 35], cosine similarity [38], personalized PageRank [66], and collaborative filtering – item-based and user-based kNN with cosine similarity) [45] and implicit matrix factorization (iMF) [30], as implemented in the RankSys library[3]. In addition, we implement the Money algorithm [23, 26] developed at Twitter, in which, for simplicity, we include all users in the circle of trust. We also include a random and most-popular recommendation as sanity-check baselines.

Using the different previously explained data partitions, we optimize the hyperparameters of all the algorithms (both edge orientation and parameter settings) by grid search targeting nDCG@10. The detailed parameter grid is shown in Table A.1 in the additional material. For those that can take advantage of edge weights (i.e. the adapted IR models and link prediction algorithms), we select the best option. The resulting optimal settings are detailed in Table A.2 in the supplementary material. The configuration of the algorithms includes setting the edge direction where applicable. We have already explored such setting in some detail in prior work [56].

To avoid trivializing the recommendation task on directed networks, reciprocating links are excluded from both the test network and the systems' output. Given the high reciprocation ratio on Twitter, recommending reciprocal links would be a trivial, hard-to-beat baseline. Moreover, users do already notice when another user retweets or mentions them since Twitter systematically sends notifications, thereby an additional recommendation would be redundant and would barely add any value for users.

6.4. General Results

In our experiments, we measure the behaviour of the IR models described in Section 5 in relation to the rest of the algorithms, in terms of nDCG and MAP at cutoff 10. The results of this comparison are shown in Table 4. We examine the statistical significance of the comparisons between algorithms using the two-tailed paired t-tests at $p < 0.05$. Furthermore, we complement this statistical significance analysis with additional Tukey Honest Significant Difference (HSD) tests at $p < 0.05$ to account for multiple comparisons, as suggested in the literature [18, 51]. The outcome of all these statistical significance tests is reported in full detail in Section B in the supplementary material. We observe that, in general, as expected, the Tukey HSD test is more conservative – increasingly so when we compare higher numbers of systems. However, both tests broadly agree on the most important conclusions. Through the rest of the paper, when making observations about statistical significance, we do them using both tests unless otherwise specified.

A first observation in our results shows that the IR models are, indeed, quite effective for our task since, in every dataset, at least one of them appears among the top 5 algorithms in the comparison, in terms of both metrics. Among the

[3]http://ranksys.org (Last accessed April 23rd 2020)

Table 4: Effectiveness of the IR model adaptations and baselines. A cell color goes from red (lower) to blue (higher values) for each metric/dataset, with the top value both underlined and highlighted in bold. The differences between BM25 (the best IR model) and iMF (the best baseline) are always statistically significant by a two-tailed paired t-test at $p < 0.05$, except MAP@10 in the 200-tweets interactions network. A more conservative Tukey HSD test also finds that all the differences between BM25 and iMF are statistically significant except those for both MAP@10 and nDCG@10 on the 200-tweets interactions network. Full statistical significance tests are reported in Figures B.1 to B.3 in the supplementary material.

	1-month				200-tweets				Facebook	
	Interactions		Follows		Interactions		Follows			
Algorithm	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP
BM25	0.1042	0.0440	0.1177	0.0479	**0.1097**	**0.0583**	0.1159	0.0405	0.5731	0.3686
BIR	0.0983	0.0399	0.1173	0.0475	0.1004	0.0522	0.1140	0.0389	0.5720	0.3670
PL2	0.0962	0.0390	0.1184	0.0467	0.0983	0.0507	0.1166	0.0405	0.5712	0.3670
DFRee	0.0849	0.0335	0.1112	0.0424	0.0976	0.0511	0.1164	0.0408	0.5601	0.3548
Extreme BM25	0.0848	0.0326	0.1167	0.0472	0.1034	0.0543	0.1132	0.0390	0.5519	0.3480
QLD	0.0767	0.0311	0.0985	0.0370	0.0937	0.0500	0.1152	0.0411	0.5799	0.3732
QLJM	0.0882	0.0350	0.1154	0.0456	0.0694	0.0373	0.1161	0.0412	0.5664	0.3616
QLL	0.0855	0.0338	0.1098	0.0410	0.0939	0.0497	0.1160	0.0399	0.5599	0.3564
DPH	0.0806	0.0317	0.1063	0.0399	0.0976	0.0514	0.1159	0.0409	0.5568	0.3514
DFReeKLIM	0.0709	0.0273	0.0927	0.0336	0.0953	0.0506	0.1138	0.0402	0.5565	0.3519
DLH	0.0643	0.0246	0.0901	0.0325	0.0901	0.0480	0.1122	0.0396	0.5566	0.3526
VSM	0.0292	0.0121	0.0503	0.0174	0.0425	0.0223	0.0787	0.0275	0.5237	0.3224
iMF	**0.1388**	0.0663	**0.1462**	0.0590	0.1035	0.0558	**0.1329**	**0.0465**	0.5210	0.3207
User-based kNN	0.1367	**0.0669**	0.1413	**0.0594**	0.0954	0.0510	0.1297	0.0462	0.5457	0.3491
Item-based kNN	0.1174	0.0533	0.1296	0.0529	0.0724	0.0389	0.1205	0.0413	0.4542	0.2650
Money	0.1315	0.0638	0.1129	0.0465	0.0932	0.0492	0.1131	0.0401	0.5867	0.3809
Adamic-Adar	0.0981	0.0398	0.1175	0.0467	0.0997	0.0513	0.1140	0.0388	0.5746	0.3695
MCN	0.0918	0.0368	0.1189	0.0467	0.0948	0.0493	0.1110	0.0373	0.5585	0.3546
Pers. PageRank	0.0800	0.0315	0.0965	0.0362	0.0630	0.0317	0.0843	0.0276	**0.5891**	**0.3826**
Jaccard	0.0317	0.0117	0.0543	0.0176	0.0571	0.0307	0.0848	0.0287	0.4913	0.2967
Cosine	0.0393	0.0186	0.0497	0.0168	0.0480	0.0243	0.0768	0.0265	0.4943	0.2981
Popularity	0.0572	0.0291	0.0449	0.0161	0.0422	0.0212	0.0397	0.0098	0.0523	0.0234
Random	0.0014	0.0005	0.0011	0.0002	0.0003	0.0001	0.0018	0.0003	0.0030	0.0009

different models, two of them stand out above the rest in terms of accuracy and robustness across the different datasets, namely BM25 and PL2.

Indeed, BM25 is always among the top 6 algorithms in our comparison in terms of both nDCG and MAP, even achieving the top accuracy for the interaction network in the 200-tweets dataset. The only exception is the follows network for that same dataset, although the difference with the 4^{th} algorithm in the comparison, PL2, is not statistically significant in this network (see Figure B.3 for statistical significance detail in the supplementary material). One of the reasons for such a performance is likely the ability of BM25 to take advantage of interaction frequency (edge weights) better than any other algorithm – in fact, all other algorithms except VSM and cosine similarity, produce worse results when using a non-binary edge representation.

The second model, PL2, is the other consistent algorithm in the comparison, always among the top 7 best approaches. To a lesser extent, BIR also provides good results in our comparison. On the contrary, the vector space model achieves the worst values for an IR model in our experiments, close to the popularity-based recommendation. The rest of the IR algorithms usually stand as mid-packers in the different used datasets.

Classical collaborative filtering algorithms are a hard baseline to beat in the Twitter networks, especially in the case of the implicit matrix factorization algorithm, which achieves top values of nDCG for three of the Twitter datasets, and second-to-best in the remaining one. This is also true for MAP@10, where the algorithm is second to best in the interaction networks and the follows network for the 1-month dataset, whereas it is the best in the 200-tweets follows network. However, on the Facebook network, the algorithm is far from optimal, even achieving a worse performance than the vector space model. This shows that the algorithm iMF is unable to capture the undirected nature of the network when generating the recommendations as effectively as it does in directed networks. Almost the same can be stated for the user-based kNN approach, which obtains top performance values in the 1-month dataset, and stands out in the 200-tweets follows network. However, its performance decays in the interaction network of 200-tweets, and the Facebook network.

For the rest of algorithms, MCN, Adamic-Adar and item-based collaborative filtering behave as mid-packers in the different datasets. Jaccard and cosine similarity perform rather poorly in comparison, similarly to the vector space model. These three models are the ones that penalize the popularity of the candidate users the most, thus showing that in general, avoiding popularity bias of the recommenders in recommendations does not allow us to get lead to good results. We can relate this to the so-called preferential attachment effect in social networks [11], in which users with a high degree are more likely to be contacted by other users than users with low degree.

Finally, the random walk algorithms show very different behaviors depending on the graph. They achieve the best results for the Facebook graph, but they are suboptimal in the directed networks. In those networks, Money works better than personalized PageRank, especially in the 1-month dataset.

Overall, we find that information retrieval models, and, particularly, BM25 and PL2 make for highly competitive contact recommendation approaches in different types of networks. BM25 is however not the top algorithm, since iMF overall has a slight advantage in effectiveness. Based on our own previous experience with iMF in different domains, we note that iMF works particularly well under higher data density, which may explain its advantage over the IR models in the follows graphs – as well as its better performance in the 1-month vs. 200-tweets interaction networks. On the other hand, iMF seems fragile and sensitive to high network clustering degrees, as is the case in the Facebook dataset, where iMF performs rather poorly: as we see later in Table 5, over 99% of the test edges are only at distance 2 of the target user in this network (i.e. they are triad closing edges, in accordance with the high clustering coefficient shown in Table 3). While the IR models and the friends-of-friends approaches can only recommend users precisely at distance 2, iMF can recommend very distant people in the network, who are highly unlikely – in this dataset – to hit a link in the test set. Random walks can also travel long distances but their personalized teleportation parameter directly controls how far they pick their recommendations. This can be tuned (in the automatic parameter setting) to gravitate as near the target user as is most optimal – as is the case when the clustering coefficient of the network is high – which explains why these algorithms perform quite well in this network.

Money and kNN obtain decent results in some datasets, but are quite sub-optimal in the others. We can therefore say that BM25 is a decent second best in recommendation accuracy after matrix factorization. Hence, the IR models prove to be quite an effective option for contact recommendation, and all the more so considering that they are orders of magnitude faster than iMF, as was reported in [56]. In the next sections, we propose new IR-based algorithms for closing the accuracy gap with the top performing contact recommendation methods (i.e. iMF on Twitter and random walks on Facebook).

7. IR Models for Neighbor Selection in Collaborative Filtering

In the previous sections, we have studied how competitive IR models like BM25 [50] or PL2[4] are when they are used as contact recommendation methods. We have observed that these algorithms regularly achieve high accuracy, and could be considered among the most effective approaches for the task. However, we have also observed that other baselines, such as matrix factorization [30] in directed networks or personalized PageRank [66] in undirected networks are still hard to beat. In order to close the gap between our models and the top-performing ones, we now explore the use of user-based and item-based nearest-neighbor (kNN) collaborative filtering approaches, following the idea that Valcarce et al. [61, 62] proposed for general recommendation: applying IR models as similarities.

7.1. Algorithm Definition

Recommendation by nearest-neighbors (kNN) is a wide and popular family of collaborative filtering algorithms [45]. They rely on the principle that similar users prefer similar items and similar items are preferred by similar users. The main idea behind these algorithms is to select the top k most similar users/items to the target user/candidate item (known as the neighborhood of the user/item), and compute their recommendation scores as a linear combination of the neighborhood ratings. Similarity values are computed using the ratings provided by the users in the system to the different items.

When we use the neighborhood of the target user, we refer to such methods as user-based kNN. There are many variations of such methods, but we focus, from now on, on the following straightforward variant, which is well-behaved in general domains [16]:

$$f_u(i) = \sum_{w \in N_k(u)} \mathrm{sim}(u,w) r_w(i) \tag{27}$$

where $N_k(u)$ represents the top k most similar users to u; $\text{sim}(u, w)$ is the value of a similarity function between users u and w, and $r_w(i)$ is the rating provided by w to the item i.

When we find the most similar items to the candidate item i ($N_k(i)$), we talk about item-based kNN methods. A possible approach is the following:

$$f_u(i) = \sum_{j \in N_k(i)} \text{sim}(i, j) r_u(j) \tag{28}$$

where $N_k(i)$ is the neighborhood of item i, of size k, $\text{sim}(i, j)$ represents the similarity between items, and $r_u(j)$ is the rating the target user provides for the neighbor item j.

We illustrate these methods in the first row of Figure 7. If we want to use these methods for the contact recommendation task, it is straightforward: since users also play the role of items, we just need to substitute the items by other users, as shown in the second row of the same figure. The equations for these approaches are then defined as:

$$f_u(v) = \sum_{t \in N_k(u)} \text{sim}(u, t) w(t, v) \tag{29}$$

for the user-based version, and

$$f_u(v) = \sum_{t \in N_k(v)} \text{sim}(v, t) w(u, t) \tag{30}$$

for item-based kNN.

In the previous experiments, as a typical and effective option in the kNN approaches [45], we used the cosine similarity $\text{sim}(v, t) = \cos(v, t)$. When observing Figure 7, we can realize that the structure for computing the similarity between users in both user-based and item-based approaches for contact recommendation is familiar: it is essentially the same structure as in the adapted IR models and friends-of-friends link prediction approaches (Adamic-Adar [1], MCN [35] and Jaccard [32]), which we have used in the previous sections as direct contact recommendation methods. But now, we consider the use of such contact ranking functions as novel methods to find nearest neighbors, that is, as similarity functions, in a kNN scheme, which finally recommends contacts.

Furthermore, in the item formulations, to find the similarities between users, we used the ratings they provided to the items (which here, are represented by the outgoing edges). In the item-based case, we used the ratings provided to the different items (here, the incoming edges). However, as we already did in Sections 4 and 5 for the standalone models, we also find it interesting to allow different orientations for the target/candidate user and their neighbors. Because of that, we denote as Γ^q the orientation of the target/candidate user, and, as Γ^d the orientation selection for the neighbor users.

Table 5: Percentage of edges in the test set whose endpoint users are at a given (undirected) distance from each other in the input network.

	Twitter 1-month		Twitter 200-tweets		Facebook
Distance	Interactions	Follows	Interactions	Follows	
2	79.61%	96.45%	57.60%	91.90%	99.43%
3	19.62%	3.53%	37.73%	8.05%	0.55%
4	0.75%	0.02%	4.53%	0.06%	0.01%
5	0.02%	–	0.13%	–	–
∞	–	–	–	–	0.01%

7.2. Motivation

Before we test these models, we provide an intuition about why these models might improve the IR and friends-of-friends link prediction approaches when applied as contact recommendation algorithms. We have, mainly, two reasons for that. The first one responds to a limitation of the standalone models: all friends-of-friends algorithms just recommend neighbor users of their own neighbors. Therefore, no link is recommended towards users at undirected distance (ignoring edge direction) greater than 2. In Table 5, we show the number of links in the test graphs for which the endpoints are at different distances in the input network: in general, most links are created toward users at distance 2 (even more than 95% of them in several networks), but there is a reasonably high amount of links at distance 3, which cannot be reached by IR models and friends-of-friends.

In a nearest-neighbors method, this distance increases to 3, allowing us to reach more than 95% of the new edges in the test set for all the different networks. To see why this happens, it is sufficient to observe Figure 7. For example, for user-based kNN, if we take the target user as origin, all the common users between the target user and the selected user are, for sure, at undirected distance 1. Neighbor users might be at distances 1 or 2 from the target user, and, because of that, the target user, who is at distance 1 from the neighbor user, might be at distance 2 or 3. An analogous rationale explains why item-based approaches achieve a similar effect.

The second reason follows our experiments in Section 6.4: for the algorithm comparison shown in Table 4, the performance of the cosine-based collaborative filtering approaches (user-based and item-based kNN) showed a marked improvement with respect to cosine similarity on its own. It is hence natural to wonder if the IR models might also result in better performance when similarly integrated in a kNN scheme. The idea may seem all the more promising considering that IR models perform considerably better than cosine as standalone contact recommendation methods.

7.3. Experimental Results

To evaluate this new idea, we apply the same experimental setup as in the experiments reported in Section 6.4: we first tune our parameters using the

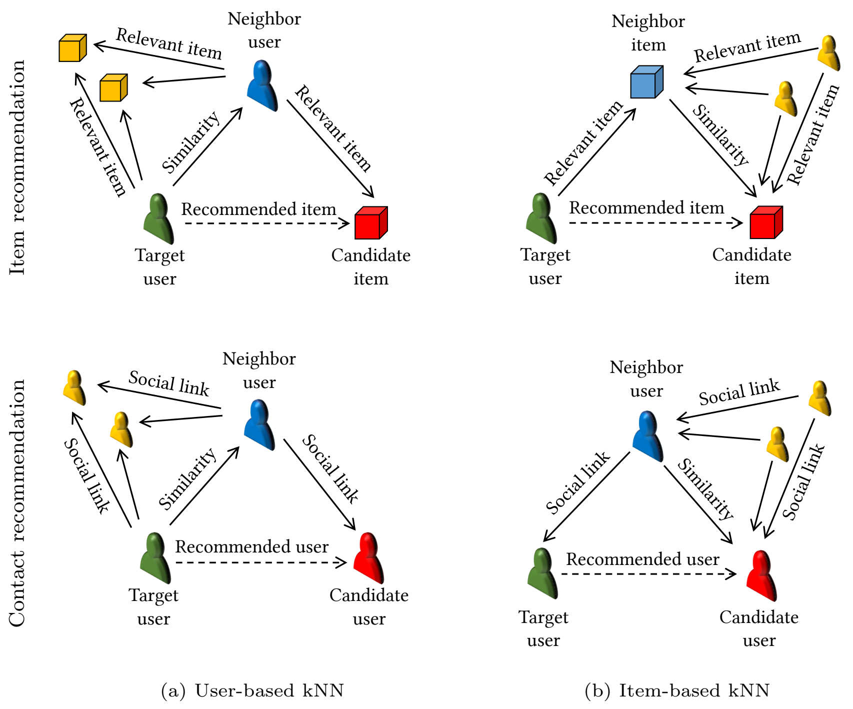


Figure 7: User-based and item-based kNN in classical item recommendation task vs. the same models in contact recommendation.

validation graph, and then, we generate recommendations using the training set and evaluate them on the test set. In the supplementary material (Tables A.3 to A.5), we provide the optimal parameters selection for the different user-based and item-based approaches for the different datasets.

Figure 8 gives a broad perspective of the comparison between the optimized standalone models and the optimized kNN approaches in terms of nDCG@10. We observe that, with the exception of the Facebook network, user-based kNN generally improves over the behavior of IR and link prediction approaches as standalone methods (green points are above the $y = x$ line in the graphs, by considerable distance in many cases). In the Facebook network, only a few algorithms achieve that improvement: BM25, Extreme BM25, Jaccard and cosine similarity. Results are mixed in the item-based variant: in general, it improves the worst approaches, but the advantages for the effective models such as BM25 are not as clear. For example, item-based kNN improves over these competitive models for the interactions networks of the 1-month dataset, but not in any of the 200-tweets networks. Surprisingly, one of the models does not work correctly for neither the item-based or user-based approaches on any of the datasets: the query likelihood model with Laplace smoothing does not seem to find good neighbors in any of the strategies, achieving low nDCG.

We can notice that, in general, user-based kNN has a better performance

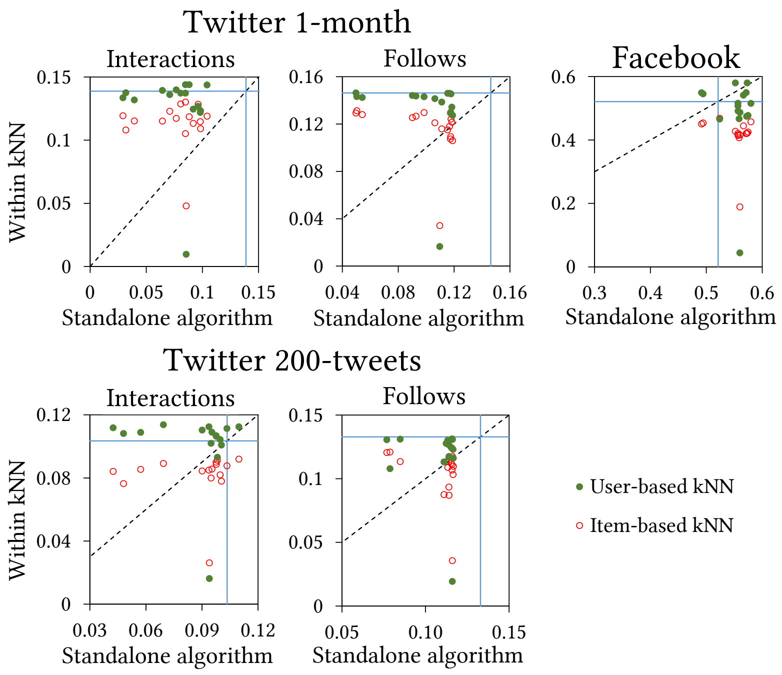


Figure 8: Comparison of nDCG@10 values between friends-of-friends as standalone models vs. when integrated in user-based (green) and item-based (red) kNN. The x axis represents the nDCG@10 value for the standalone methods, and the y axis is the nDCG@10 value for the corresponding kNN approach using the standalone method as a similarity function. The dotted line shows the $y = x$ cut, and the solid blue lines show the nDCG@10 value of the implicit matrix factorization algorithm as the top-performing algorithm in the standalone experiments.

than item-based kNN. This is consistent with previous observations in recommender systems, where item-based kNN usually works best when the number of items is much smaller than the number of users in the system, and user-based kNN works best in the opposite situation [45]. Here, the number of users and items is the same, which seems to result in an advantage for the user-based variant. We therefore focus on the user-based option in the comparisons that follow. We see already in Figure 8 that many points in the graphs lie above the blue horizontal line: they show that many kNN combinations perform better than the matrix factorization algorithm, which was hard to beat by standalone alternatives in our previous experiments (blue vertical line in the graphs).

Table 6 shows a detailed comparison of user-based kNN integrating different contact recommendation algorithms as a neighbor selection function, against the most effective algorithms in our experiments in Section 6.4: matrix factorization (best in three of the four Twitter datasets), personalized PageRank (best in the Facebook dataset) and BM25 (best IR algorithm in general, and best algorithm in the 200-tweets interaction network). As can be observed, the user-based approach with BM25 similarity stands out among the rest in all directed graphs:

Table 6: Effectiveness of the user-based kNN (abbreviated as "UB" in the table) + the IR model adaptations and other baselines. The cell color goes from red (lower) to blue (higher values) for each metric/dataset, with the top value both underlined and highlighted in bold. The best user-based approach (with BM25 similarity) differences with iMF are statistically significant for the Facebook and the interactions networks in terms of nDCG@10 and MAP@10 (except for the 200-tweets dataset) and are not significant in the case of the follows networks in any of the metrics. Full additional detail of statistical significance is given in Figures B.4 to B.6 in the supplementary material.

	1 month				200 tweets				Facebook	
	Interactions		Follows		Interactions		Follows			
Algorithm	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP
UB BM25	**0.1494**	0.0730	0.1455	**0.0603**	0.1125	0.0588	0.1313	**0.0467**	0.5802	0.3734
UB Extreme BM25	0.1493	**0.0730**	0.1460	0.0597	0.1114	0.0585	0.1302	0.0459	0.5799	0.3733
UB QLJM	0.1482	0.0718	0.1460	0.0594	**0.1137**	0.0597	0.1309	0.0455	0.5414	0.3406
UB QLD	0.1444	0.0701	0.1431	0.0577	0.1125	**0.0602**	0.1247	0.0448	0.5156	0.3194
UB DLH	0.1439	0.0698	0.1442	0.0584	0.1103	0.0576	0.1277	0.0438	0.5159	0.3225
UB DFReeKLIM	0.1407	0.0672	0.1437	0.0581	0.1090	0.0571	0.1267	0.0432	0.5075	0.3150
UB VSM	0.1387	0.0675	0.1432	0.0581	0.1118	0.0597	0.1080	0.0354	0.4664	0.2811
UB DFRee	0.1402	0.0668	0.1386	0.0552	0.1068	0.0552	0.1231	0.0420	0.4872	0.3018
UB DPH	0.1410	0.0671	0.1416	0.0569	0.1064	0.0549	0.1239	0.0427	0.4914	0.3035
UB BIR	0.1287	0.0615	0.1294	0.0507	0.1008	0.0514	0.1176	0.0402	0.4748	0.2881
UB PL2	0.1260	0.0636	0.1344	0.0553	0.0931	0.0487	0.1162	0.0405	0.5498	0.3449
UB QLL	0.0097	0.0037	0.0166	0.0054	0.0162	0.0074	0.0194	0.0057	0.0443	0.0247
UB Cosine	0.1367	0.0669	**0.1464**	0.0599	0.1082	0.0569	0.1306	0.0456	0.5457	0.3491
UB Jaccard	0.1392	0.0683	0.1425	0.0589	0.1089	0.0572	0.1311	0.0459	0.5494	0.3521
UB Adamic-Adar	0.1264	0.0597	0.1301	0.0529	0.1043	0.0539	0.1168	0.0400	0.4775	0.2915
UB MCN	0.1291	0.0611	0.1274	0.0511	0.1020	0.0530	0.1132	0.0382	0.4670	0.2832
iMF	0.1388	0.0663	0.1462	0.0590	0.1035	0.0558	**0.1329**	0.0465	0.5210	0.3207
BM25	0.1042	0.0440	0.1177	0.0479	0.1097	0.0583	0.1159	0.0405	0.5731	0.3686
Pers. PageRank	0.0800	0.0315	0.0965	0.0362	0.0630	0.0317	0.0843	0.0276	**0.5891**	**0.3826**
Popularity	0.0572	0.0291	0.0449	0.0161	0.0422	0.0212	0.0397	0.0098	0.0523	0.0234
Random	0.0014	0.0005	0.0011	0.0002	0.0003	0.0001	0.0018	0.0003	0.0030	0.0009

in both interaction networks, it beats matrix factorization, with a significant difference in nDCG (and also in terms of MAP for the 1-month network); in the follows networks, the model is slightly behind iMF in terms of nDCG and slightly above it for MAP, but the results are not statistically significant in either metric (thus indicating a technical tie). On the Facebook network, user-based BM25 is also the best kNN combination, but it does not improve over personalized PageRank. Other strong IR models when used as similarities are Extreme BM25 and query likelihood with Jelinek-Mercer smoothing. In contrast, BIR and PL2 do not provide good results in general, and Laplace query likelihood fails at finding useful neighbors for recommendation.

In conclusion, the use of IR models as similarities within nearest-neighbors recommendation schemes provides an improvement over the standalone models, especially in the case of directed networks like Twitter, where user-based approaches with BM25 or query likelihood similarities improve or, at least, tie with the results of the best approaches in the experiments described in Section 6.4. Therefore, the user-based kNN "pass" after the standalone methods manages to catch up with and overcome the implicit matrix factorization approach. This is interesting since, despite not being as fast as the IR models on their own, the training and execution times needed for a user-based nearest-neighbors approach is orders of magnitude smaller than that needed for the iMF algorithm [56], thus posing a good baseline for link recommendation.

8. Learning to Rank

Thus far, we have defined and analyzed different unsupervised approaches for suggesting people to follow or befriend in a given network: no information about the relevance of the approaches was used in order to train them. However, it has been found in the last few years that the top performance in text retrieval is commonly achieved by supervised machine learning approaches such as learning to rank techniques [37]. Therefore, we now study whether we can further enhance the effectiveness of IR models in contact recommendation through such learning to rank techniques.

8.1. Adapting Learning to Rank to Contact Recommendation

Since applying learning to rank to search markedly differs from the traditional unsupervised models, before we report our experiments, we need to determine how these approaches can be applied for suggesting users in networks. In Section 8.1.1, we describe the general process for deploying a learning to rank technique in textual IR. Then, in Section 8.1.2, we describe how we adapt such a process to the task of contact recommendation.

8.1.1. Learning to Rank in Textual IR

In devising a search system using a learning to rank approach, two steps can be distinguished: how the model is built (i.e. how the training is achieved), and how the trained model is applied in production. To train the model, two sets of queries are needed: a set of training queries and a set of validation queries (different from the training ones). Since learning to rank methods are supervised, a set of relevance judgments is needed for each of these queries. Once both sets have been collected, the following steps are typically followed:

1. For each training query, obtain a set of k candidate documents using a simple IR model.
2. For each retrieved document, generate a fixed set of features involving the document and the query.
3. Using those features and example relevance judgments for each query-document pair, train the model. Use the validation data to prevent overfitting.

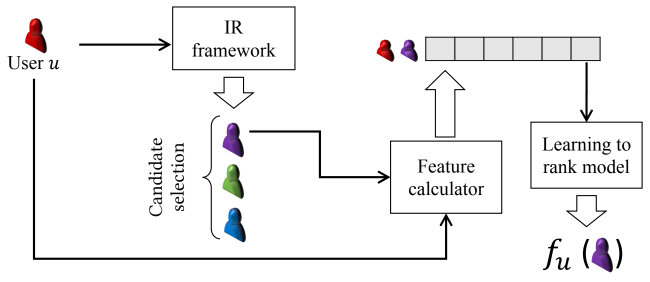


Figure 9: Framework for using learning to rank models.

The first step allows the system to do some initial pruning of documents: typically, only a tiny fraction of the documents in a collection provides useful information for the query. Since unsupervised IR models are usually effective for the IR task, applying one of these models allows to retrieve a subset of relevant documents, as well as a set of negative examples. A typical way to retrieve the k initial documents is to use a model that has good values for such metrics as recall at cutoff k. Indeed, Liu [37] described the common use of the classical BM25 weighting model to retrieve $k = 1000$ documents. Macdonald et al. [40] examined the necessary value of k, the number of candidate documents to retrieve – for large corpora such as ClueWeb09, this was found to be upwards of 1000 for informational queries, but less for navigational queries when anchor text was used.

The second step is possibly the most important: determining a good set of features that effectively capture the search task can make a major difference for learning to rank models to achieve good results. In IR, a large variety of features have been proposed and evaluated, such as deploying various weighting models [41] or (query independent) document properties such as the PageRank [14] score or the document length.

Once the model is trained, it can be used to provide search results. To do that, the procedure for a query is:

1. Use the same IR model as before to select k initial documents.
2. Generate the features for each query-document pair. Features should be the same as used in training.
3. Using the feature vectors for each document, obtain the score of the document.
4. Sort the documents according to their scores to produce the final ranking.

8.1.2. Learning to Rank in Contact Recommendation

Since we want to apply learning to rank techniques for recommending contacts, we have to adapt the steps listed in the previous section to our task. To do that, we make use of the mapping we previously applied in Sections 4 and 5. We can then adapt learning to rank to contact recommendation by substituting

Data subsets in unsupervised recommendation	Learning to rank elements
Training subset	Training/validation features
Validation subset	Training/validation relevance judgments
Input subset	Input features for evaluation
Test subset	Test relevance judgments

Table 7: Relation between the split described in Section 6 and the learning to rank elements.

target users for training queries, candidate users for documents, and user pair features for query-document pair features.

Similarly to the evaluation of unsupervised contact recommendation, we use three disjoint sub-networks (three disjoint sets of edges): training, validation and test sets, as described in Section 6.2. But now, the training and validation sets play quite a different role: rather than used for hyperparameter tuning, they are leveraged to generate feature vectors for target-candidate user pairs, labelled with relevance judgments. The relevance judgments are taken from the validation set: edges present in the validation set are used as positive relevance, and user pairs absent from it are taken as non-relevant. The features are built from the training set as follows: taking the training edges as input, we sample the top k candidate contacts for each target user using a standalone contact recommender. Then, we generate a feature vector (possibly including recommender scores) for each sampled target-candidate user pair.

Now the set of relevance-labelled features is randomly split into a new training and a new validation subset, in such a way that all feature vectors of the same target user are assigned to the same side of the split. Then, the learning to rank algorithm is trained on these two sets. Finally, we run the trained learning algorithms on the union of the training and validation sets: candidates are sampled, features are generated, and the final contact recommendations are produced using those features, which are then evaluated using the test edges as final relevance judgments. We illustrate these steps in Figure 9, while Table 7 summarizes the relationship between the elements in the original split, described in Section 6 and the feature sets we use in our learning to rank experiments.

8.2. Experimental Setup

Using the approach described in he previous section, we now want to study whether we can actually obtain better contact recommendation performances with the supervised learning to rank techniques than with the unsupervised IR-based models (both standalone and within a kNN scheme) as described in Sections 5 and 7. For this purpose, we first describe the setup of our experiments. In the following, we describe the sampling approach for each dataset, the cutoff for selecting the candidate users, the learning to rank approaches, the set of features, and how to build the training, validation and test sets.

First, we describe how we select the sampling approach. Document sampling in learning to rank in search tasks is commonly undertaken using IR models [37, 40]: taking a query as input, the IR models return documents that are related to the query through the terms they have in common. If we see the query/term/document space as a tripartite graph, this returns graph nodes (documents) at distance two from the query node. In order to follow the same principle in the adaptation to contact recommendation, we consider sampling methods that return "document" users at undirected distance two from the target users.

Of all the standalone methods considered in our previous sections, only the IR models and the friends-of-friends approaches ensure this constraint. Thus, taking all the standalone IR models described in Section 5 and the friends-of-friends baselines, we take their optimal configurations for each network, as described in Table A.2. in the supplementary material. Then, on the validation set we select, for each network, the recommendation method that maximizes recall at the same cutoff as that we chose for the sampling approach, which we set to $k = 1,000$. The selected samplers are the following: BIR for the Facebook network and the 200-tweets follows datasets, BM25 for the 1-month interactions dataset, and Extreme BM25 for the remaining networks.

Once the labelled features for training are generated, we randomly select the features of 80% of the users as the training set, and use the rest as the validation set for training the learning to rank approaches. In our experiments, we train and evaluate a LambdaMART model [15, 67]: a learning to rank model based on gradient-boosted regression trees. In particular, we use the Jforests LambdaMART implementation developed by Ganjisaffar [22]. Note that we do not vary the hyperparameters of the learning to rank algorithm – indeed, we apply the default parameters provided by the authors of Jforests in their implementation[4]. Tuning the hyperparameters of LambdaMART would only likely improve the effectiveness, yet, as we show later in Section 8.3, the LambdaMART model using the default parameters already outperforms those models evaluated in this article thus far.

Since both the friends-of-friends and kNN approaches provide competitive baselines for contact recommendation, following previous works in IR [41], we have chosen to combine different contact recommendation algorithms as learning to rank ensembles to improve their effectiveness: we take as features the ranking functions of a selection of approaches, which we list in Table 9. Instead of taking the raw values of these ranking functions, we take a common approach when multiple algorithms are combined: we normalize the ranking scores by minmax [34]. For each target user u, we normalize the feature values by:

$$\bar{f}_u(v) = \frac{f_u(v) - \min_{w \in \mathcal{U}_u} f_u(w)}{\max_{w \in \mathcal{U}_u} f_u(w) - \min_{w \in \mathcal{U}_u} f_u(w)} \tag{31}$$

where $\mathcal{U}_u$ denotes the selection of candidate users for user u. If a target-candidate pair is not retrieved by the feature algorithm, we set $\bar{f}_u(v) = 0$.

[4]https://github.com/yasserg/jforests (Last accessed 31th July 2019)

Table 8: Comparison between LambdaMART and the best algorithms in the previous sections using nDCG@10 and MAP@10. A cell color goes from red (lower) to blue (higher values) for each metric/dataset, with the top value both underlined and highlighted in bold. Full statistical significance details are given in Figures B.7 in the supplementary material.

	1 month				200 tweets				Facebook	
	Interactions		Follows		Interactions		Follows			
Algorithm	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP
LambdaMART	0.1424	0.0673	**<u>0.1538</u>**	**<u>0.0641</u>**	**<u>0.1272</u>**	**<u>0.0695</u>**	**<u>0.1373</u>**	**<u>0.0486</u>**	**<u>0.6112</u>**	**<u>0.4006</u>**
UB BM25	**<u>0.1494</u>**	**<u>0.0730</u>**	0.1455	0.0603	0.1125	0.0588	0.1313	0.0467	0.5802	0.3734
iMF	0.1388	0.0663	0.1462	0.0590	0.1035	0.0558	0.1329	0.0465	0.5210	0.3207
Pers. PageRank	0.0800	0.0315	0.0965	0.0362	0.0630	0.0317	0.0843	0.0276	0.5891	0.3826

8.3. Results

In the following, we first report and analyze the performance of learning to rank in contact recommendation. Next, we assess the relative importance of the various used features in learning to rank.

8.3.1. General Effectiveness

First, we check how effective learning to rank is when including all the proposed algorithms in Sections 5 and 7 as features. Table 8 shows the comparison of the LambdaMART ensemble with the top standalone recommendation algorithms: the user-based BM25, as the best algorithm in the interaction networks, implicit matrix factorization (iMF) as the top approach in the follows networks, and Personalized PageRank, as the best algorithm in the Facebook network. We note that, with the exception of the interactions network of the 1-month dataset, the learning to rank ensembles outperform all algorithms, thus resulting in the most effective approach in our reported experiments. This advantage is always statistically significant for both nDCG@10 and MAP@10, excepting MAP for the 200-tweets follows network. In the 1-month interaction network, despite not beating the user-based kNN combined with BM25, the LambdaMART ensemble obtains slightly (though not significantly) better results than matrix factorization, making it the second best approach. The reason why LambdaMART does not beat the user-based kNN on that dataset, despite including it as a feature, lies in the properties of the nearest-neighbors approaches: as we explained earlier, the user-based approaches are able to recommend users up to distance three, while we intentionally restricted learning to rank to recommend users only at distance 2, as a result of the initial selection of candidates by standalone IR models (in this case, BM25) which, by structure, operate at two hops in the network.

8.3.2. Feature Selection

Another question that arises is the identification of the specific features that make the learning to rank approach particularly effective. In order to answer this question, we first train different LambdaMART models, selecting different

Table 9: Groups of features.

Group	Algorithms
Probability ranking principle (PRP)	BIR, BM25, Extreme BM25
Query likelihood (QL)	QLD, QLJM, QLL
Divergence from Randomness (DFR)	DFRee, DFReeKLIM, DLH, DPH, PL2
Vector space model (VSM)	VSM
Link prediction (LP)	Adamic-Adar, Cosine, Jaccard, MCN
Item-based (IB kNN)	All item-based kNN algorithms
User-based (UB kNN)	All user-based kNN algorithms
All IR models (IR)	VSM, PRP, QL, DFR
Friends-of-friends (FOAF)	IR, LP
Nearest neighbors (All kNN)	UB kNN, IB kNN
All models	FOAF, UB kNN, IB NN

groups of features. The different groups we choose are shown in Table 9. Then, we evaluate these algorithms over the test set to compare their differences.

As we can see in Figure 10, in all the networks, the user-based kNN approaches combined with the friends-of-friends methods turn out to be the best features on their own. Beyond that, we observe some commonalities and differences between the different networks. When it comes to the IR algorithms, the probability ranking principle usually achieves the best results, followed by DFR, while the query likelihood approaches do not make good features for our task. This is a bit different in Facebook, where the QL model is tied with the PRP ones. Link prediction methods such as Adamic-Adar and Jaccard also work better as features than the combination of all IR models. The only exception to this is the interaction networks for the 200-tweets dataset. In this network, the IR models achieve a large advantage thanks to the performance of the BM25 algorithm on this dataset, thus making the LambdaMART model trained only with the link prediction approaches a bit worse than it is in other networks.

Finally, for three of the networks, ensembles of item-based nearest-neighbor models provide a better performance than standalone friends-of-friends models, and are the second best set of features after the user-based kNN. The two only exceptions are the Facebook network, where item-based kNN is among the worst approaches, and the 200-tweets interactions, where BM25 has a marked advantage, making it difficult for the item-based approaches to improve the performance of the IR models.

9. Conclusions and Future Work

Although separately developed to much extent by different communities, text-based search and recommendation are very related tasks. This relation has been explored in prior research on the general perspective of adapting IR techniques to item recommendation [13, 62]. In our present work, we further instantiate this step to the recommendation of contacts in social networks. Our thorough investigation has found that adapting IR models leads to empirically effective solutions, that are to some extent simpler than the previously developed

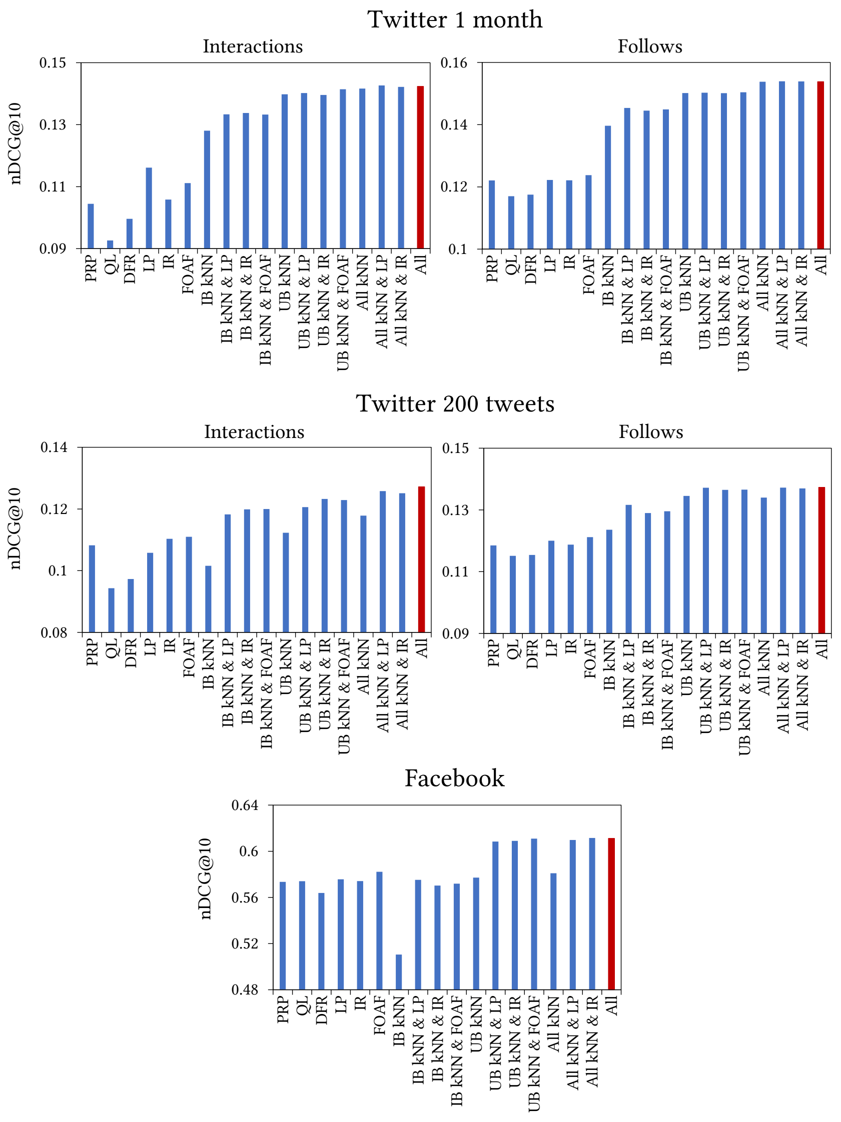


Figure 10: Comparison of LambdaMART models with different sets of features, defined in Table 9.

adaptations of IR models for the general item recommendation task [13]. We found that BM25 in particular is competitive with the best state-of-the-art contact recommendation approaches in terms of effectiveness over Twitter and Facebook data representing social interactions of different types – and obtained with different sampling approaches.

Interestingly, in our experiments, IR models have been shown to be even

– consistently and substantially – more effective when used as neighbor selection methods within nearest-neighbor collaborative filtering schemes. Even IR models with average or poor performance as standalone methods become competitive within this strategy – i.e. even if they are not particularly effective at directly recommending social bonds by themselves, they are effective at finding good neighbors whose friends are interesting people for the target user to bond or interact with. As a natural step for exploring how far we can improve effectiveness by borrowing and tailoring successful techniques from IR, we have investigated the adaptation of learning to rank techniques – an approach that generally enables the best results in search tasks – confirming that indeed further improvements are achieved in this direction in the contact recommendation task. Overall, we found that the IR models are effective in three roles: as direct contact recommenders, as neighbor selectors, and as samplers and features in learning to rank.

Compared to alternative heuristic solutions, the translation of new and principled IR models and their thorough adaptation to contact recommendation can add new and deeper insights to our understanding of the contact recommendation task and how we solve it, by importing the theory and foundations upon which the IR models were developed. Reciprocally, this can contribute to a broader perspective on IR models, their meaning, interpretation, and usefulness in different tasks, bringing higher levels of abstraction. We have found for instance that some IR models (e.g. BM25 in our experiments) are able to take a better advantage of the user-user interaction frequency than heuristic algorithms. We have likewise observed that the followers seem to describe the social value of candidate recommendations better than the followees, whereas the union of both consistently appears to best represent the social needs of target users.

It should be stressed that the analogy between contact recommendation and text IR is not a trivial equivalence. It involves, to begin with, mappings from a tridimensional space (a tripartite graph of queries, documents and terms) to a unidimensional space (a unipartite graph of users). Moreover, in text retrieval the content of documents is often assumed to be static; if it does change, a document update can be processed as a new document. In our approach, the equivalent of the words in documents are the social bonds of candidate users which, in contrast to document retrieval, evolve over time – a condition inherently needed for the recommendation task to make sense: we are predicting how the network will grow in the future based on its present structure, and no useful recommendation is possible at zero growth. In the text IR analogy, the contact recommendation task can be seen from a certain angle as predicting the new words that will be added into a document in the future. On the other hand, if a link is present in the input network, it would not be useful to recommend it, in the general scenario where users may find little use in being recommended people they are already connected to. This sets an important difference compared to text retrieval, where the equivalent constraint would imply that a document that is known to be relevant to the query should not be returned.

Beyond our present results, our research opens a new direction through which innovations in the area of IR models can be potentially translated to the contact recommendation problem. As future work, we plan to extend our current study by considering further evaluation dimensions beyond accuracy, such as recommendation novelty and diversity [19, 58], or the effects that recommendation can have on the evolution of the network structure [3, 55].

Acknowledgements

Javier Sanz-Cruzado and Pablo Castells were partially supported by the Spanish Government (grant ref. TIN2016-80630-P). Craig Macdonald and Iadh Ounis were partially supported by the European Community's Horizon 2020 research and innovation programme, under grant agreement nº 779747 entitled BigDataStack.

References

[1] Adamic, L. A., & Adar, E. (2003). Friends and neighbors on the Web. *Social Networks*, *25*, 211–230. doi:10.1016/S0378-8733(03)00009-1.

[2] Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*, 734–749. doi:10.1109/TKDE.2005.99.

[3] Aiello, L. M., & Barbieri, N. (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)* (pp. 111–120). Cambridge, United Kingdom: ACM. doi:10.1145/3018661.3018733.

[4] Amati, G. (2003). *Probability Information Models for Retrieval based on Divergence from Randomness*. Ph.D. thesis University of Glasgow.

[5] Amati, G. (2006). Frequentist and bayesian approach to information retrieval. In *ECIR 2006: Advances in Information Retrieval* number 3936 in LNCS (pp. 13–24). London, United Kingdom: Springer. doi:10.1007/11735106_3.

[6] Amati, G., Ambrosi, E., Bianchi, M., Gaibisso, C., & Gambosi, G. (2007). FUB, IASI-CNR and University of Tor Vergata at TREC 2007 Blog track. In *Proceedings of The 16th Text REtrieval Conference (TREC 2007)*. Gaithersburg, Maryland, USA: NIST.

[7] Amati, G., Amodeo, G., Bianchi, M., Marcone, G., Bordoni, F. U., Gaibisso, C., Gambosi, G., Celi, A., Nicola, C. D., & Flammini, M. (2011). FUB, IASI-CNR, UNIVAQ at TREC 2011 microblog track. In *Proceedings of The 20th Text REtrieval Conference (TREC 2011)*. Gaithersburg, Maryland, USA: NIST.

[8] Amati, G., & Van Rijsbergen, C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems*, *20*, 357–389. doi:10.1145/582415.582416.

[9] Backstrom, L., & Leskovec, J. (2011). Supervised random walks. In *Proceedings of the 4th ACM international conference on Web Search and Data Mining (WSDM 2011)* (pp. 635–644). Hong Kong, China: ACM. doi:10.1145/1935826.1935914.

[10] Baeza-Yates, R., & Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology behind Search*. (2nd ed.). Harlow, England: Pearson Education Ltd.

[11] Barabàsi, A.-L., & Albert, R. (1999). Emergence of Scaling in Random Networks. *Science*, *286*, 509–512. doi:10.1126/science.286.5439.509.

[12] Belkin, N. J., & Croft, W. B. (1992). Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM*, *35*, 29–38. doi:10.1145/138859.138861.

[13] Bellogín, A., Wang, J., & Castells, P. (2013). Bridging memory-based collaborative filtering and text retrieval. *Information Retrieval*, *16*, 697–724. doi:10.1007/s10791-012-9214-z.

[14] Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, *30*, 107–117. doi:10.1016/S0169-7552(98)00110-X.

[15] Burges, C. (2010). *From RankNet to LambdaRank to LambdaMART: An Overview*. Microsoft Research Technical Report MSR-TR-2010-82 Microsoft.

[16] Cañamares, R., & Castells, P. (2017). A probabilistic reformulation of memory-based collaborative filtering - implications on popularity biases. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)* (pp. 215–224). Tokyo, Japan: ACM. doi:10.1145/3077136.3080836.

[17] Cannistraci, C. V., Alanis-Lobato, G., & Ravasi, T. (2013). From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific Reports*, *3*. doi:10.1038/srep01613.

[18] Carterette, B. A. (2012). Multiple Testing in Statistical Analysis of Systems-Based Information Retrieval Experiments. *ACM Transactions on Information Systems*, *30*. doi:10.1145/2094072.2094076.

[19] Castells, P., Hurley, N. J., & Vargas, S. (2015). Novelty and Diversity in Recommender Systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender Systems Handbook* (pp. 881–918). Boston, MA, USA: Springer. doi:10.1007/978-1-4899-7637-6_26.

[20] Chaney, A. J., Blei, D. M., & Eliassi-Rad, T. (2015). A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys 2015)* (pp. 43–50). Vienna, Austria: ACM. doi:10.1145/2792838.2800193.

[21] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., & Yin, D. (). Graph neural networks for social recommendation. In *Proceedings of the The Web Conference 2019 (WWW 2019)* (pp. 417–426). San Francisco, CA, USA: ACM. doi:10.1145/3308558.3313488.

[22] Ganjisaffar, Y., Caruana, R., & Lopes, C. V. (2011). Bagging gradient-boosted trees for high precision, low variance ranking models. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011)* (pp. 85–94). Beijing, China: ACM. doi:10.1145/2009916.2009932.

[23] Goel, A., Gupta, P., Sirois, J., Wang, D., Sharma, A., & Gurumurthy, S. (2015). The who-to-follow system at Twitter: Strategy, algorithms, and revenue impact. *Interfaces*, *45*, 98–107. doi:10.1287/inte.2014.0784.

[24] Goodman, L. A. (1961). Snowball Sampling. *The Annals of Mathematical Statistics*, *32*, 148–170. doi:10.1214/aoms/1177705148.

[25] Gunawardana, A., & Shani, G. (2015). Evaluating Recommender Systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender Systems Handbook* (pp. 265–308). Boston, MA, USA: Springer. doi:10.1007/978-1-4899-7637-6_26.

[26] Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., & Zadeh, R. (2013). WTF: The Who to Follow Service at Twitter. In *Proceedings of the 22nd international conference on World Wide Web (WWW 2013)* (pp. 505–514). Rio de Janeiro, Brazil: ACM. doi:10.1145/2488388.2488433.

[27] Guy, I. (2015). Social Recommender Systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender Systems Handbook* (pp. 511–543). Boston, MA: Springer. doi:10.1007/978-1-4899-7637-6_15.

[28] Hannon, J., Bennett, M., & Smyth, B. (2010). Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the 4th ACM conference on Recommender Systems (RecSys 2010)* (pp. 199–206). Barcelona, Spain: ACM. doi:10.1145/1864708.1864746.

[29] Hasan, M. A., Chaoji, V., Salem, S., & Zaki, M. J. (2006). Link Prediction Using Supervised Learning. In *Proceedings of SDM 06 Workshop on Link Analysis, Counterterrorism and Security* (pp. 1828–1832). Bethesda, Maryland: IEEE.

[30] Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)* (pp. 263–272). Pisa, Italy: IEEE. doi:10.1109/ICDM.2008.22.

[31] Huang, X. L., Tiwari, M., & Shah, S. (2013). Structural Diversity in Social Recommender Systems. In *Proceedings of the 5th ACM RecSys Workshop on Recommender Systems and the Social Web (RecSys RSWeb 2013) co-located with the 7th ACM conference on Recommender Systems (RecSys 2013)* (pp. 1–7). Hong Kong, China.

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

[33] Jelinek, F., & Mercer, R. (1980). Interpolated estimation of Markov source parameters from sparse data. In E. S. Gelsema, & L. N. Kanal (Eds.), *Pattern Recognition in Practice* (pp. 381–402). Amsterdam, Netherlands: North-Holland.

[34] Lee, J. H. (1997). Analyses of multiple evidence combination. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1997)* (pp. 267–276). Philadelphia, PA, USA: ACM. doi:10.1145/258525.258587.

[35] Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, *58*, 1019–1031. doi:10.1002/asi.20591.

[36] Liu, H., Hu, Z., Haddadi, H., & Tian, H. (2013). Hidden link prediction based on node centrality and weak ties. *Europhysics Letters*, *101*. doi:10.1209/0295-5075/101/18004.

[37] Liu, T.-Y. (2007). Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*, *3*, 225–331. doi:10.1561/1500000016.

[38] Lü, L., & Zhou, T. (2011). Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, *390*, 1150–1170. doi:10.1016/j.physa.2010.11.027.

[39] Macdonald, C., McCreadie, R., Santos, R. L., & Ounis, I. (2012). From puppy to maturity: Experiences in developing terrier. In *Proceedings of the SIGIR 2012 Workshop in Open Source Information Retrieval (OSIR 2012)* (pp. 60–63). Portland, Oregon, USA.

[40] Macdonald, C., Santos, R. L., & Ounis, I. (2013). The whens and hows of learning to rank for web search. *Information Retrieval*, *16*, 584–628. doi:10.1007/s10791-012-9209-9.

[41] Macdonald, C., Santos, R. L., Ounis, I., & He, B. (2013). About learning models with multiple query-dependent features. *ACM Transactions on Information Systems*, *31*, 11. doi:10.1145/2493175.2493176.

[42] MacKay, D. J. C., & Peto, L. C. B. (1995). A hierarchical Dirichlet language model. *Natural Language Engineering*, *1*, 289–307. doi:10.1017/S1351324900000218.

[43] McAuley, J., & Leskovec, J. (2012). Learning to discover social circles in ego networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS 2012)* (pp. 539–547). Lake Tahoe, Nevada: Curran Associates Inc.

[44] Meng, Z., Liang, S., Bao, H., & Zhang, X. (2019). Co-embedding attributed networks. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM 2019)* (pp. 393–401). Melbourne, Australia: ACM. doi:10.1145/3289600.3291015.

[45] Ning, X., Desrosiers, C., & Karypis, G. (2015). A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender Systems Handbook* (pp. 37–77). Boston, MA, USA: Springer. doi:10.1007/978-1-4899-7637-6_2.

[46] Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., & Lioma, C. (2006). Terrier: A high performance and scalable information retrieval platform. In *Proceedings of the 2nd workshop on Open Source Information Retrieval (OSIR 2006) at SIGIR 2006* (pp. 18–25). Seattle, WA.

[47] Parapar, J., Bellogín, A., Castells, P., & Barreiro, A. (2013). Relevance-based language modelling for recommender systems. *Information Processing and Management*, *49*, 966–980. doi:10.1016/j.ipm.2013.03.001.

[48] Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in Information Retrieval (SIGIR 1998)* (pp. 275–281). Melbourne, Australia: ACM. doi:10.1145/290941.291008.

[49] Robertson, S. E. (1977). The Probability Ranking Principle in IR. *Journal of Documentation*, *33*, 294–304. doi:10.1108/eb026647.

[50] Robertson, S. E., & Zaragoza, H. (2009). The Probabilistic Relevance Framework : BM25 and Beyond. *Foundations and Trends in Information Retrieval*, *3*, 333–389. doi:10.1561/1500000019.

[51] Sakai, T. (2018). *Laboratory Experiments in Information Retrieval - Sample Sizes, Effect Sizes, and Statistical Power* volume 40 of *The Information Retrieval Series*. Springer. doi:10.1007/978-981-13-1199-4.

[52] Salton, G., & McGill, M. J. (1983). *Introduction to modern information retrieval.*. New York, NY, USA: McGraw-Hill, Inc.

[53] Salton, G., Wong, A., & Yang, C. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*, 613–620. doi:10.1145/361219.361220.

[54] Sanz-Cruzado, J., & Castells, P. (2018). Contact Recommendations in Social Networks. In S. Berkovsky, I. Cantador, & D. Tikk (Eds.), *Collaborative Recommendations: Algorithms, Practical Challenges and Applications* (pp. 519–569). Singapore: World Scientific Publishing. doi:10.1142/9789813275355_0016.

[55] Sanz-Cruzado, J., & Castells, P. (2018). Enhancing structural diversity in social networks by recommending weak ties. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys 2018)* (pp. 233–241). Vancouver, Canada: ACM. doi:10.1145/3240323.3240371.

[56] Sanz-Cruzado, J., & Castells, P. (2019). Information retrieval models for contact recommendation in social networks. In *ECIR 2019: Advances in Information Retrieval* number 11437 in LNCS (pp. 148–163). Cologne, Germany: Springer. doi:10.1007/978-3-030-15712-8_10.

[57] Sanz-Cruzado, J., Castells, P., & López, E. (2019). A simple multi-armed nearest-neighbor bandit for interactive recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys 2019)* (pp. 358–362). Copenhagen, Denmark: ACM. doi:10.1145/3298689.3347040.

[58] Sanz-Cruzado, J., Pepa, S. M., & Castells, P. (2018). Structural Novelty and Diversity in Link Prediction. In *Companion of the The Web Conference 2018 on The Web Conference 2018 (WWW 2018)* (pp. 1347–1351). Lyon, France: IW3C2. doi:10.1145/3184558.3191576.

[59] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*, 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.

[60] Tang, J., Hu, X., & Liu, H. (2013). Social recommendation: a review. *Social Network Analysis and Mining*, *3*, 1113–1133. doi:10.1007/s13278-013-0141-9.

[61] Valcarce, D. (2015). Exploring statistical language models for recommender systems. In *Proceedings of the 9th ACM conference on Recommender Systems (RecSys 2015)* (pp. 375–378). Vienna, Austria: ACM. doi:10.1145/2792838.2796547.

[62] Valcarce, D., Parapar, J., & Barreiro, Á. (2017). Axiomatic Analysis of Language Modelling of Recommender Systems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *25*, 113–127. doi:10.1142/S0218488517400141.

[63] Valverde-Rebaza, J. C., Roche, M., Poncelet, P., & de Andrade Lopes, A. (2018). The role of location and social strength for friendship prediction in location-based social networks. *Information Processing and Management*, *54*, 475 – 489. doi:10.1016/j.ipm.2018.02.004.

[64] Wang, J., Robertson, S., de Vries, A. P., & Reinders, M. J. T. (2008). Probabilistic relevance ranking for collaborative filtering. *Information Retrieval*, *11*, 477–497. doi:10.1007/s10791-008-9060-1.

[65] Wang, J., de Vries, A. P., & Reinders, M. J. T. (2008). Unified relevance models for rating prediction in collaborative filtering. *ACM Transactions on Information Systems*, *26*, 16:1–16:42. doi:10.1145/1361684.1361689.

[66] White, S., & Smyth, P. (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)* (pp. 266–275). Washington, DC, USA: ACM. doi:10.1145/956755.956782.

[67] Wu, Q., Burges, C., Svore, K., & Gao, J. (2008). *Ranking, Boosting and Model Adaptation*. Microsoft Research Technical Report MSR-TR-2008-109 Microsoft Research.