Published in Knowledge-based Systems
Volume 216 (2021)
DOI: https://doi.org/10.1016/j.knosys.2021.106806

Context-aware Graph-based Recommendations Exploiting Personalized PageRank

Cataldo Musto[a,*], Pasquale Lops[a], Marco de Gemmis[a], Giovanni Semeraro[a]

[a] *Universita degli Studi di Bari "Aldo Moro", Department of Computer Science*

Abstract

In this article we present a *context-aware recommendation method* that exploits graph-based data models and Personalized PageRank to provide users with recommendations.

In particular, our approach extends the basic graph-based representation that relies on *users* and *items* nodes by introducing a third class of nodes, that is to say, *context* nodes, whose goal is to model the different contextual situations in which an item can be consumed. Given such a data model, we used Personalized PageRank to identify the most suitable recommendations for each user: in a nutshell, our model is based on the intuition that *context nodes* shall be used to influence *random walks*, in order to assist the algorithm in identifying the items that are relevant in a particular contextual setting.

In the experimental evaluation we investigated the effectiveness of the approach on three different datasets. The results showed that our context-aware graph-based approach overcame the baselines in most of the experimental settings and obtained the best overall results in cold-start situations, thus confirming the validity of the methodology.

Keywords: Recommender Systems, Context, Graphs, PageRank

*Corresponding author

Email addresses: cataldo.musto@uniba.it (Cataldo Musto), pasquale.lops@uniba.it (Pasquale Lops), marco.degemmis@uniba.it (Marco de Gemmis), giovanni.semeraro@uniba.it (Giovanni Semeraro)

1. Introduction

Recommender Systems (RS) [1] represent one of the most disrupting technologies that appeared on the scene in the last decade [2]. Basically, such systems acquire information about user needs, interests and preferences, and tailor their behavior on the ground of such information, thus supporting people in several decision-making tasks in a personalized manner.

The effectiveness of these systems is confirmed by several evidences: as an example, 35% of Amazon's revenues are generated through its recommendation engine[1], and many companies frequently report claims that RS contribute from 10% to 30% of total revenues [3]. Similar success stories regarding other web companies such as Facebook, Spotify and Netflix further confirm that RS nowadays play a central role to improve the effectiveness of web platforms.

However, this effectiveness is the consequence of a very long path run by both researchers and practitioners of RSs. Indeed, most of the models that were originally proposed in the area tended to *oversimplify* the recommendation process, since they did not take into account the concept of *context*. Conversely, context plays a key role in every decision-making process, and RS were no exception. As an example, the mood can direct the choice of the movie to be watched. Similarly, a different company (*friends, family, children*) drives the choice of a restaurant, and so on.

This intuition is investigated from the early 2010s by the research line regarding context-aware recommender systems (CARS) [4], whose goal is to evaluate how the injection of contextual data and contextual factors impacts on recommendation processes.

State of the art approaches for CARS are broadly split into three main groups: *Pre-filtering* approaches use the contextual information to filter out from the recommendation process all the ratings that are provided in contex-

[1] `http://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers`

tual situations different from the current one: as an example, if a user needs a recommendation for a place to go with friends, pre-filtering techniques ignore the preferences she expressed when she spent time with her family or her colleagues. Differently, *post-filtering* techniques use contextual information *after* the standard non-contextual recommendation methods are applied on the recommendation data. In this case, a standard approach is to use information about context as a weighting factor to re-rank a non-contextual recommendation list. Finally, *contextual modeling* assumes that contextual information is used *inside* the recommendation algorithm together with the user and item data.

Regardless of the contextualization strategy which is used, several work confirmed effectiveness of CARS, since these techniques usually outperform merely collaborative or content-based recommendations [5]. However, it is worth to state that these methodologies are not completely free from problems: given that for each item a larger number of different ratings (one for each contextual situation, actually) need to be expressed, CARS are even more prone to *cold-start* and *sparsity* issues [6], which unavoidably hurt the overall effectiveness of the algorithms.

In this article we fit into this research line and we present a new approach to provide users with context-aware recommendations. In particular, our methodology relies on a graph-based data model and exploits Personalized PageRank (PPR) [7] to generate recommendations.

The intuition behind this work is to extend the basic graph-based representation that relies on *users* and *items* nodes by introducing a third class of nodes, that is to say, *context* nodes, whose goal is to model the different contextual conditions in which the items can be consumed. Next, given such a data model, we used PPR as recommendation algorithm and we suggested the items with the highest PageRank score. In this case, we biased PPR by modifying the personalization vector p (as shown in [7]) through the adoption of different strategies to distribute the weights. More details and a more precise formalization of the approach are reported in Section 3.2.

To make a long story short, our model is based on the intuition that *context*

nodes shall be used to influence the *random walks* that connect the user to the most suitable items, by taking more into consideration the items that have been previously enjoyed by other users in a particular contextual setting.

The strength and the novelty of the approach lies in the following aspects, that will be further discussed in Section 3.3:

- Differently from related literature (see [8] and [9]), we avoided the exponential growth of the context nodes encoded in the graphs by introducing just *one* node for each user and *one* node for each contextual setting. This choice allows to obtain a *more connected* graph where a higher amount of paths connecting the target user with relevant items exist.

- Based on the previous design choice, we can also state that we developed an approach which is less prone to cold start. Indeed, even if a user *did not express any preference in a specific contextual setting*, our strategy can still provide recommendations by exploiting: *(i)* the paths that connect the preferences of the current user in different contextual settings; *(ii)* the paths that encode the preferences that *other* users in the same contextual settings. This is a clear hallmark of our work, since most of the traditional techniques to provide users with context-aware recommendation (even complex matrix factorization technique) significantly suffer of the cold start.

- We introduced different strategies to influence random walks, which are expected to provide high flexibility to our recommendation strategy. Indeed, our framework allows to tune the behavior of the algorithm on the ground of the characteristics of the dataset, by making the strategy suitable to very different situations and recommendation scenarios.

In the experimental evaluation we showed that our approach overcame several state-of-the-art baselines in most of the experimental settings. Moreover, the results emerged from the experiments showed that a proper tuning of the weights in the personalization vector p tend to improve the accuracy of the

framework. In particular, we noted that the best performance are often obtained by inducing the algorithm in running across the paths that pass through both the *current contextual setting* as well as the *items* the user previously enjoyed. To sum up, through this article we provide the following contributions:

1. We propose a tripartite graph-based data model encoding *users*, *items* and *context* nodes, that allows to provide users with context-aware recommendations;
2. We present different strategies to influence *random walks* (and, in turn, the recommendations returned by the framework) based on different distribution of the weights assigned to the different classes of nodes;
3. We validated our methodology through several experiments, and we showed that our approach obtains competitive results, which are in line (or above, as in cold-start scenarios) with respect to most of the baselines.

The rest of the article in organized as follows: in Section 2 we provide an overview of related work in the area. Next, Section 3 focuses on the methodology and Section 4 evaluates the effectiveness of our framework. Finally, Section 5 presents the conclusions and sketches directions for future work.

2. Related Work

This work investigates two different research lines, that is to say, *graph-based recommendations* and *context-aware recommendations*. In this section we provide an overview of related work in the areas, by emphasizing the hallmarks of the current work with respect to relevant literature.

2.1. Graph-based Recommendations

Most the approaches that have been presented in the area of graph-based recommendations can be roughly split into two classes: *(i)* approaches that exploit spreading activation techniques; *(ii)* approaches inspired by PageRank (PR) and random walk [10].

The use of spreading activation for recommendations purposes is investigated from the early 2000s. As an example, Kovacs et al. present in [11] a domain-indipendent framework for context-aware recommendations which is based on spreading activation. Next, Zhang et al. [12] confirmed the effectiveness of such methodology in the task of recommending research papers. Recently, the approach has been exploited in a context-aware recommendation setting by Papneja et al. [13] and by Bahramian et al. [14], who designed a RS for the tourism domain that relied on spreading activation. In particular, the authors formally designed an ontology to encode both contextual situations (*e.g.*, weather, time, etc.) as well as knowledge about the items and run spreading activation to identify the most relevant POI for the user.

As for the use of PR and random walk, one of the early work in the area is due to Hotho et al. [15], who propose an adaptation of PR to tackle the problem of providing users with *tag recommendations*. In particular, in this work the authors create a graph connecting resources with the tags used in the community to annotate them, and run PR to identify the most suitable tags that can be used to annotate a resource.

Similar intuitions were proposed by Baluja et al. [16], who presented a RS for YouTube that is based on random walks run on the bipartite user-video graph, and by Bogers [17], who used PageRank over the graph that connects movies to tags, genres and actors. A more sophisticated approach is presented in [18], where the authors exploit random walks to calculate similarities between users and items, which are used to tackle the recommendation problem as a *link prediction* one. The flexibility and the effectiveness of graph-based representations is also confirmed by de Gemmis et al. [19], who showed that recommendations based on *random walk with restart* can provide users with *serendipitous encounters*.

Recently, the effectiveness of deep learning models has been demonstrated in graph-based recommendations as well. As an example, Xie et al. [20] exploited graph-based data models to learn embeddings that are then exploited to provide users with *location* recommendations. Another interesting attempt has been

proposed by Wang et al. [21], who proposed a recommendation framework exploiting a Knowledge Graph Attention Network. In particular, their model falls into the category of knowledge-aware recommendations since they rely on a tripartite graph connecting users, items and entities (descriptive properties of the items) and design a deep architecture based on attention mechanisms to provide users with recommendations. Unfortunately, both these models did not model nor encode the concept of *context*, thus it was not possible to take them into account in our experimental evaluation.

Generally speaking, differently from these work, we preferred to directly run PPR over our graph-based data model, without any other processing and without the application of any other algorithm. This is choice is motivated by the findings emerging from previous research [22], where it is shown that recommendation strategies based on PPR can provide state-of-the-art recommendation accuracy.

In particular, through this paper we try to make one step further with respect to our previous work [23], where we investigated how the distribution of the weights in PPR can influence the overall performance of a recommendation framework. Specifically, the findings discussed in [23] are here extended by evaluating PPR in a context-aware recommendation scenario, which is a poorly investigated research line.

Finally, it should be pointed out that the in this work we only modeled *users*, *items* and *contexts*, without encoding descriptive features of the items. The injection of such information into graph based representation is typically covered by *knowledge graphs* [24], whose positive impact in recommendation tasks has been already investigated in [25], in [26], and in the recent work by Caro-Martinez et al. [27] , where a node similarity measure inspired by social network analysis is used to identify relevant problems to be solved.

By the way, this research line is out of the scope of this research, since we just aim to investigate to what extent graphs can be used to provide users with *context-aware recommendations*. However, the easiness of integration of knowledge graphs in the current model already provides us with a clear direction

for future work.

2.2. Context-aware Recommendations

The area of context-aware RSs (CARS) is quite recent, and relies on the idea that *contextual factors*, such as mood, company, time of the day and so on need to be taken into account in recommendation scenarios, since they have a significant impact on decision-making processes. One of the first attempts in the area is due to Herlocker and Konstan, who proposed in [28] to adapt the recommendation list to the specific task of the user. Next, Adomavicius et al. introduced in [5] a *multi-dimensional model* where the user-item matrix is enriched with contextual information. As expected, in both these cases the experiments showed that the introduction of data about context significantly improves the accuracy of the recommendations.

Accordingly, the research in the area gained more and more interest in the recent years. An important research effort was stimulated by the series of CARS[2] and CAMRa[3] workshops. These workshops triggered the development of many approaches, mostly based on the manipulation of the *user* $\times$ *item* $\times$ *contexts* matrix. As an example, Karatzoglou et al. [29], proposed the *multiverse recommendation* framework, in which different types of context are considered as additional dimensions in the representation of the data as a tensor. Similar methods have been proposed by Baltrunas [30] and by Hidasi et al. [31]. Even if the most recent approaches are based on the use of deep learning techniques, as those proposed by Kim et al. [32], by Xin et al. [33], and Liu et al. [34], who recently used attention mechanisms over a graph connecting users and items, the use of matrix factorization is still considered as state of the art in the area of context-aware recommendations. For further reading about CARS, we suggest to refer to the recent survey by Zheng [35].

The distinctive trait of our work with respect to the above mentioned ap-

[2] `http://cars-workshop.org/`

[3] `http://camrachallenge.com/`

proaches lies in the adoption of a *graph-based data model* to encode information about *users, items and contexts* and the exploitation of PPR to rank the item nodes and provide the user with context-aware recommendations.

The use of PPR is also investigated by Bagci et al. [36], who use the preferences of the users to influence *random walks* over a graph that connects users and points of interests, and by Wu et al. [37], who exploited PPR scores as a *weighting factor* to re-rank a list of recommendations in a post-filtering context-aware recommendation strategy. The use of PPR is also a distinctive characteristic of the model proposed by Lee et al. [8]. In this case, the authors extend the classical bipartite user-item representation by creating a graph where each combination *user-context* is encoded as a single node. As an example, let n be the number of different contextual settings, each user is modeled through n different nodes. Another similar attempt is proposed by Yao et al. [9], who introduce the concept of multi-layer context-graph. In this case, the authors present a data model that includes a context node for each combination of contextual settings, and use PPR to bias random walks and to obtain the recommendations. The distinctive traits of the current work with respect to such literature will be discussed in next section.

3. Methodology

In this section we describe our graph-based recommendation framework. First, we show how we extended the basic *bipartite representation* that relies on users and items by introducing *context nodes*, next we provide some basics of PPR algorithm and we introduce the different strategies to distribute weights in PPR to influence random walks over the *tripartite* context-aware representation.

3.1. Graph-based Representations

The main idea behind a basic graph-based model is to represent *users* and *items* as *nodes* in a graph. Formally, given a set of users $U = \{u_1, u_2, \dots u_n\}$ and a set of items $I = \{i_1, i_2, \dots i_m\}$, an *undirected* graph $G =< V, E >$ is

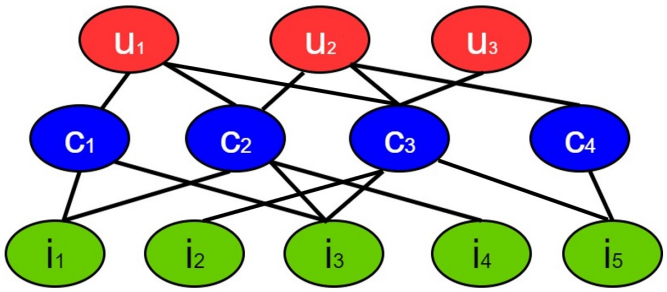

Figure 1: A *toy example* showing our graph-based data model. Users, contexts and items are represented in red, blue and green respectively. Every rating is expressed as a *path* that connects a user to an item passing through the contexts in which the evidence has been collected.

instantiated, where $|V| = |U| + |I|$. Of course, G is a bipartite graph, since it models two different kind of entities (that is to say, *users* and *items*).

Differently from such a data model, our novel *context-aware data model* introduces a third class of nodes, that is to say, context nodes, whose goal is to model the different contextual situations in which an item can be consumed. Formally, given k different contextual dimensions (*mood, company, etc.*) each of which is described through v different values (*mood=good, mood=neutral, mood=bad, etc.*), a set of *context nodes* C, where $|C| = k * v$ is introduced in the graph. As previously stated, we encode just *one* node for each contextual setting, without considering their combinations. Of course, by introducing context nodes we obtain a *tripartite* graph, since three different kind of entities are now modeled.

Figure 1 provides a *toy example* of the graph-based data model we propose in this work. In this case, users are represented with *red* nodes, contexts are represented with *blue* nodes and items are represented with *green* nodes. It is important to emphasize that each context node represents a *contextual situation*, not a *contextual dimension*. This means that a node it is used to represent 'mood=good' rather than just 'mood'. Accordingly, if two different values can be used to describe the contextual dimension *mood*, two different nodes are

included in the graph. For the sake of completeness, we want to emphasize that Figure 1 just reports a the *static* representation our graph-based data model, showing how a traditional bipartite graph is extended to introduce a new class of nodes. The details concerning how to run PR over this graph and how exploit such a representation to provide users with context-aware recommendations are provided in Section 3.2.

Once the *context nodes* have been injected in the graph, it is necessary to encode user preferences as well. In graph-based data models the preferences are modeled as *edges* that connect the users to the items they like. However, differently from classical recommendation methods where each preference can be expressed as a function *like(user,item)*, in CARS the preferences depend on the different contextual situations in which an item can be consumed, thus the previous function shall be represented as *like(user,item,context)*.

In order to extend the representation of user preferences in a context-aware fashion, we tackled the problem as follows: for each user u_i who liked an item i_j in a specific contextual setting c, with $c \in C$ [4] (since c can be a combination of contexts $c_1 \dots c_s$) we create $2s$ edges that connect u_i with each c_k, and each context c_k with the item i_j, with $k = 1 \dots s$.

By referring to Figure 1 again, if the evidence that u_1 liked i_1 in the contextual setting $C = \{c_1, c_2\}$ is collected, we create four different edges. The first two edges connect u_1 to c_1 and c_2, while the other two connect both the contexts with i_1. Of course, this process is repeated over all the preferences expressed by all the users. The merge of all the edges modeling the preferences of the users provides us with the *context-aware data model* that we used to generate recommendations. For the sake of completeness, we want to point out that whether ratings are expressed on a *discrete* scale, they have to be preliminary mapped to a *binary* scale (like/dislike) before being encoded in the graph. As an example, this can be done by setting a *threshold rating*.

[4] C is the power set of C, containing the single contexts.

3.2. Running Personalized PageRank

Given our context-aware representation, we need an algorithm to provide each node $i \in I$ with a relevance score, in order to rank the available items and provide users with recommendations.

A very popular strategy to rank nodes in a graph is to adopt PR algorithm [10], which is able to assign a score to every node in the graph depending on its link structure. If a node i has a link to a node j, then i is implicitly conferring some importance to j, and precisely it confers an importance which is inversely proportional to the outdegree of node i. Consider a random surfer who begins at a node i and executes a random walk on the graph as follows: at each time step, the surfer proceeds from his current node i to a randomly chosen node i is linked to. If i has N outlinks, the surfer proceeds at the next time step to one of these nodes, with equal probabilities $1/N$. As the surfer proceeds in this random walk from node to node, he visits some nodes more often than others; intuitively, these are nodes with many links coming in from other frequently visited nodes. The idea behind PR is that pages visited more often in this walk are more important. Instead of following the link structure, the random surfer can also decide to directly jump to any other node in the graph. This operation is called *teleport*, and also allows to deal with nodes with no out-links. In the classical PR the destination of a teleport operation is modeled as being chosen uniformly at random from nodes. In other words, if N is the total number of nodes in the graph, the teleport operation takes the surfer to each node with probability 1/N. This means that the random surfer at any node can invoke the teleport operation with probability $0 < \alpha < 1$, and the standard random walk, i.e. follow an out-link chosen uniformly at random, with probability $1-\alpha$, where α is a fixed parameter chosen in advance, called *dumping factor*.

Formally, let M be the column normalized adjacency matrix of the graph

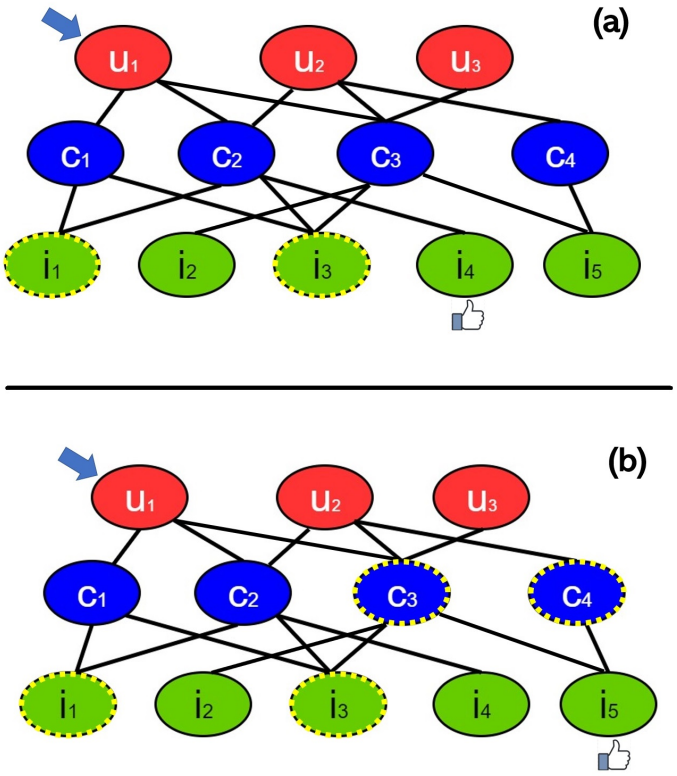

Figure 2: Running PPR with two different strategies to distribute the weights. In Figure 2-a we only bias random walk in order to pass through the items the user liked, represented with a dotted yellow line. In Figure 2-b we also distribute a part of the weights to the nodes representing the current contextual situation. As expected, by changing the distribution of the weight the most suitable recommendation changes as well.

G, built in this way:

$$M_{ji} = \begin{cases} \frac{1}{outdegree(i)} & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

M is a column stochastic matrix (columns sum to 1) and is interpreted as the *transition probability matrix* over the graph G.

Consider the Markov chain induced by the random walk on G, where the states are given by the nodes in G, and the stochastic transition matrix describing the transition from i to j is given by M. If M is aperiodic and irreducible, then the Ergodic Theorem guarantees that the stationary distribution of the random walk is unique [38]. The standard way of ensuring that M is irreducible is to add a new set of complete outgoing transitions with small transition probabilities to all nodes, creating a complete, and thus an aperiodic and strongly connected, transition graph. The new resulting matrix M' is defined as:

$$M' = (1 - \alpha)M + \alpha[\frac{1}{N}]_{N \times N} \quad (2)$$

PR can be thus viewed as the stationary probability distribution for the Markov chain induced by the random walk on the graph. Given the column vector $Rank^t$, where $Rank^t_i$ denotes the probability that the random walk at step t is at node i:

$$Rank^{t+1} = M' \times Rank^t = (1 - \alpha)M \times Rank^t + \alpha p \quad (3)$$

with $p = [\frac{1}{N}]_{N \times 1}$, which is the n-dimensional column vector representing a uniform probability distribution over all nodes.

The stationary probability distribution can be obtained by iterating Eq. 3 until convergence, that is, until $\left\|Rank^{t+1} - Rank^t\right\|_1$ (i.e. L_1 norm of the difference between two successive estimates) is below a certain threshold, or a maximum number of iterations is reached.

Unfortunately, classical PR does not represent the best choice for a recommendation scenario for a twofold reason: *(i)* highly connected nodes (that is

to say, items liked by many users) are provided with very high PR scores, thus the algorithm almost behaves as a simple *popularity-based approach*; *(ii)* the approach is not personalized, that is to say, two different users are provided with the same list of recommendations since the preferences of the single user are not taken into account.

Accordingly, the most suitable strategy to adopt PR and to consider also users' preferences is to exploit the previously mentioned variant of the algorithm, called PPR [7].

The gist of the algorithm is to *bias* the computation of PR to increase the importance of specific nodes by using a non-uniform $N \times 1$ personalization vector for p (see Eq. 3). This allows the random surfer to teleport towards a specific set of nodes, whose weights can be distributed according to a specific strategy. Intuitively, a different distribution of the weights can be used to induce *random walks* to different paths in the graph. As an example, a strategy could distribute 80% of the total weight among the items liked by the target user u, while the remaining 20% could be evenly distributed among the remaining nodes. By following this strategy, that we effectively adopted in our previous research [22], PPR tends to give a higher score to the items that are highly connected with those the user already liked.

In order to compute PPR, in Eq. 3, we replace the uniform vector $p = [\frac{1}{N}]_{N \times 1}$, with a non-uniform vector which allows the random surfer to teleport towards specific nodes, for example the nodes representing the items the user liked, or in a context-aware recommendation scenario the context nodes (the *blue* ones in Figure 2).

The examples provided in Figure 2-a and Figure 2-b better explain our intuition. In both the examples we generate recommendations for u_1 in the contextual setting $C = \{c_3, c_4\}$. In Figure 2-a we assume to run the algorithm by only distributing the weights over the items the user liked (in this case, i_1 and i_3, highlighted with a yellow dotted line), as in the traditional PPR. By following this strategy, it is likely that items that are highly connected to i_1 and i_3, regardless the specific contextual setting in which they have been consumed,

have a high probability of being recommended. In this case, i_4 can be a probable recommendation.

Conversely, in Figure 2-b we also distribute a part of the weight to the nodes representing the *current* contextual situation. As expected, by changing the distribution of the weights the most likely recommendation changes as well. In this case, by distributing some weights to c_3 and c_4 the item i_5 becomes the most probable recommendations.

It should be pointed out that, thanks to our novel strategy to distribute weights in the graph, item i_5 can be now reached by *random walks* even if u_1 did not express any preference in contextual setting c_4. In other terms, we can state that by distributing some of the weights to context nodes we introduce some *collaborative* information in our recommendation strategy, since the algorithm benefits of the paths built by taking into account the preferences of *other users*.

Accordingly, in this work we identified three main strategies to distribute the weights in our graph-based data model, in order to bias PR:

- `PPR-I:` *random walks* biased towards the nodes representing the items the user already liked. This kind of distribution correspond to that depicted in Figure 2-a;

- `PPR-C:` *random walks* biased towards *context* nodes. In this case, our intuition is to bias random walks to pass through the *context nodes* representing the current context with a higher probability. This will give a high PR score to the items that are highly connected to the current context nodes (that is to say, items that many users liked in the current contextual setting), without considering user preferences;

- `PPR-I-C:` combination of the two previous strategies, with a bias towards both context and item nodes. In this case, we aim to bias random walk in order to take into account at the same time the preferences of the current user as well as the current contextual situation. This strategy corresponds to the situation depicted in Figure 2-b.

In the experimental evaluation, we investigated whether different distribution of the weights in our *context-aware graph* leads to an improvement of the accuracy of our recommendation strategy.

3.3. Discussion

Our methodology is based on a *tripartite* graph-based data model that uses PPR to provide users with recommendation. In this section, we justify our design choices and we further emphasize the strengths of our methodology and the elements that distinguishes it from other approaches already presented in literature.

1. **Adoption of a graph-based representation.** Graphs provide a uniform representation based on *nodes* and *edges*, that allows to model the entities involved in the recommendation problem in a very natural fashion. Moreover, such a representation can be easily extended by encoding extra features, such as *descriptive features* of the items and - as in our case - *context nodes* modeling the different contextual situations. In our opinion, the adoption of a uniform representation to model heterogeneous data is a *strong point* that further motivates the choice of adopting a graph-based representation;
2. **Robustness in highly sparse experimental settings.** As shown in Section 2, the techniques presented in [8] and [9] show some overlap with the methodology we introduce in this work. However, some relevant difference still distinguishes our framework: in both the approaches, the authors include a new context node for each *combination* of different contextual situations. This choice leads to an exponential growth of the context nodes encoded in the graphs, and this makes difficult to identify some suitable paths connecting users and items via the (combination of) contextual settings, especially when the number of contexts is particularly huge. Conversely, we introduce just *one* node for each user and *one* node for each contextual setting. Consequently, a lower number of nodes is encoded in the graph.

Even if our design leads to a slight simplification of the way we encode contextual ratings (since ratings are based on a partial description of the contextual situations), this choice allows to obtain a *more connected* graph where a higher amount of paths connecting the target user with relevant items exist, and this leads to a recommendation strategy that is expected to be more effective and less prone to cold start.

Moreover, our approach allows to provide recommendations even if a user *not express any preference in a specific contextual settings*. Indeed, our strategy can still provide recommendations by exploiting: *(i)* the paths that connect the preferences of the current user in different contextual settings; *(ii)* the paths that encode the preferences that *other* users in the same contextual settings. This is a clear hallmark of our work, which is borrowed from traditional user-based collaborative filtering algorithms. Conversely from our work, most of the approaches that provide users with context-aware recommendation (even complex matrix factorization techniques) suffer of the cold start. To sum up, we expect that our design choices make our methodology less prone to *sparsity* and *cold-start*, and this will in turn lead to an improvement of the overall performance.

3. **Flexibility of the recommendation framework.** Another strength of the framework lies in the introduction of different strategies to bias random walks. In our opinion, this is expected to provide high flexibility to our recommendation strategy since the behavior of the algorithm can be tuned on the ground of the characteristics of the dataset. As an example, when the amount of contextual dimensions is huge and the dataset is particularly sparse, we expect that weight distributions based on `PPR-C` and `PPR-I-C` schemes provide the best results, since they induce PPR to pass through context nodes, and this gives in turn high scores to the items that are frequently liked in that particular contextual setting. Conversely, when the sparsity is lower or when the number of contextual situations is smaller, distributions based on `PPR-I` can be more helpful. To sum up, we think that our strategies to distribute weights can provide our framework with

the ability to adapt to different situations and recommendation scenarios, and this is another *strong point* of the methodology.

4. Experiments

In the experimental evaluation we carried out two experiments, whose goal was to answer to the following research questions:

1. How effective are the different strategies to distribute the weights in our context-aware data model? Which one leads to the best recommendations? *(Experiment 1)*
2. How does our framework perform with respect to state-of-the-art techniques to provide users with context-aware recommendations? *(Experiment 2)*

4.1. Experimental Design

In this section we provide all the details concerning the design of our experimental session, to allow to reproduce the experiments and to guarantee the reproducibility of the results.

4.1.1. Description of the Datasets

The evaluation was carried out on three state of the art datasets for context-aware recommendations[5], that is to say, *Tripadvisor* dataset, *DePaul movies* dataset and *Tijuana restaurants* dataset. All the datasets are reported in [39], which provides an overview of standard datasets for evaluating CARS. Thus, we can state that we evaluated our framework against state-of-the-art datasets. Some statistics is reported in Table 1.

As shown in the table, we chose datasets having very different characteristics: *TripAdvisor* is a very sparse dataset having a high number of contextual

[5] `https://github.com/irecsys/CARSKit/tree/master/context-aware_data_sets`

	TripAdvisor	**DePaul-Movies**	**Tijuana**
#Users	2371	97	50
#Items	2269	79	40
#Ratings	14063	5039	1413
#Dimensions	6	3	2
#Conditions	261	10	7
Density	0,00%	1,62%	8,91%

Table 1: Statistics about the datasets. *'Contextual dimensions'* refer to the high-level contextual facets (e.g., mood, company, etc.) while *'contextual conditions'* refer to the values that each contextual dimension can assume.

conditions. *Tijuana restaurants* is a dataset having a very small number of ratings and contexts, while *DePaul movie* has a lower density than Tijuana and a lower number of contexts as than TripAdvisor.

As for the contextual dimensions encoded in each dataset, Tijuana Restaurants dataset includes two contextual dimensions (time and location), where $time = \{weekend, weekday\}$ and $location = \{school, home, work\}$. Next, DePaul movie dataset encodes three contextual dimensions (time, location and company), where $time = \{weekend, weekday\}$ and $location = \{home, cinema\}$ and $company = \{family, partner, alone\}$. Finally, TripAdvisor dataset includes six different contextual dimensions: five location-related dimensions, such as User State, User Timezone, Item City, Item State, Item Timezone and $trip - type = \{family, couples, business, solo - travel, friends)\}$.

4.1.2. Parameters of the Experiment

In Experiment 1 we carried out several runs of the algorithm in order to compare the effectiveness of the recommendations on varying of the strategies to distribute the weights. Clearly, as recommendation algorithm we used PPR (dumping factor equal to 0.85, as in [10].

As for `PPR-I` and `PPR-C`, we evaluate three different combinations. In par-

ticular, we distribute to item nodes (or context nodes, respectively[6]) `30%, 50%` or `80%` or the total weight. The remaining weight is evenly distributed among the other nodes.

As for `PPR-I`, the $N \times 1$ personalization vector p in Eq. 3 is defined as follows:

$$p_i^{PPR-I} = \begin{cases} \frac{\beta}{|I_u|} & \text{if } i \in I_u \\ \frac{1-\beta}{N-|I_u|} & \text{otherwise} \end{cases} \tag{4}$$

where N is the total number of nodes in the graph and I_u is the set of nodes corresponding to items liked by user u, and β allows to define the percentage of the total weight to distribute to item nodes. Setting $\beta = 0.3, 0.5, 0.8$ allows to distribute `30%, 50% or 80%` of the total weight to the specific nodes.

Similarly for `PPR-C`, the $N \times 1$ personalization vector p in Eq. 3 is defined as follows:

$$p_i^{PPR-C} = \begin{cases} \frac{\beta}{|C|} & \text{if } i \in C \\ \frac{1-\beta}{N-|C|} & \text{otherwise} \end{cases} \tag{5}$$

where C is the current contextual situation, and $|C|$ represents the number of nodes in the graph corresponding to the current contextual situation.

Similarly, we propose three different configurations also for `PPR-I-C`: in the first, we give the same importance to context and item nodes (`40%` of weight to context and item nodes, `20%` to the others), while in the second and in the third we give a slightly higher weight to a class of nodes (`50%` of the weight to context node and `30%` to item nodes, and the opposite) and `20%` to the others. These values were set in an empirical fashion, and took inspiration from the weights distribution presented in [23]. Automatic tuning of the weights will be considered as future work.

As for `PPR-I-C`, the $N \times 1$ personalization vector p in Eq. 3 is defined as

[6] When we talk about item nodes or context nodes, respectively, we refer to the nodes representing the items the user previously liked and the nodes representing the *current* contextual situation.

follows:

$$p_i^{PPR-I-C} = \begin{cases} \frac{\beta_1}{|I_u|} & \text{if } i \in I_u \\ \frac{\beta_2}{|C|} & \text{if } i \in C \\ \frac{1-\beta_1-\beta_2}{N-|I_u|-|C|} & \text{otherwise} \end{cases} \quad (6)$$

where β_1 and β_2 are the percentage of the total weight to distribute to item nodes and context nodes, respectively.

To sum up, we evaluated the effectiveness of our framework with nine different combinations of weights (three strategies and three weighting schemes for each group).

For each experimental session, the performance of our weighting schemes was evaluated through a 5-cross fold validation by exploiting *F1@5, F1@10, NDCG@5 and NDCG@10* as evaluation metrics. Metrics were calculated by folowing the *"All Unrated Items"* strategy defined in [40]. By following this strategy, which is very widespread to evaluate RSs, recommended items that are not appearing in the test set of the user are negative examples. Even if this is strong assumption, it allows to reproduce well a real environment, where the goal of the RS is to rank all the items in the catalog. In [41], this strategy is referred to as *AllItems* methodology.

As for the choice of the metrics, Precision and Recall are not reported for the sake of brevity, since the F1 measure is a combination of both of them, while NDCG was preferred over MRR since the latter focuses on the position of the first relevant item returned to the user, while NDCG is able to provide a more comprehensive evaluation based on the overall list.

4.1.3. Selection of the Baselines

In Experiment 2 we compared the best-performing configuration emerging from Experiment 1 to several state-of-the-art techniques. In this case, we compared our method to the implementation of CARS available in the CARSKit library [42]. Moreover, we also compared our results to a non-personalized version of PR.

In particular, we compared our methodology to some of the most popular and well-performing techniques to provide users with context-aware recommendations, such as Context-aware Matrix Facatorization (CAMF) [30] and Contextual Sparse Linear Method (CSLIM) [43] in all their variants. By following the classification provided in [35], we exploited approaches for *dependent context modeling* which exploits the dependencies among users, items and contexts to provide users with recommendations.

As for CAMF, it represents an extension of the method traditional matrix factorization that includes new parameters to represent the interaction of contextual conditions with users and items. In particular, we compared out framework to four different variants of this method:

- **CAMF_C**. The simplest CAMF implementation, that assumes that all the contextual dimensions have a uniform influence on all ratings, without differentiating by item;
- **CAMF_CI**. This model assumes that each contextual factor influences in a way different to each item, so for each couple *(item, context)* a new parameter has to be learnt;
- **CAMF_CU**. This model is similar to the previous one, but it considers the user. Accordingly, it assumes that each contextual factor influences the users in a different way, so for each couple *(user, context)* a new parameter has to be learnt;
- **CAMF_CUCI**. This model combines the previous approaches, since it assumes that each contextual factor influences both the items and the users differently. Thus, (many) different parameters for each combination of user, item and context have to be learnt.

Differently from CAMF, CSLIM is derived from the sparse linear method (SLIM) [44] which was designed for Top-N recommendations in traditional RSs. The different implementations for CSLIM (CSLIM_C, CSLIM_CI and CSLIM_CU) follow the same intuition we have previously explained for CAMF.

Moreover, we also considered simpler contextual pre-filtering approaches, such as UserSplitting and ItemSplitting, inspired by those proposed in [45]. These methods perform a pre-processing of the data according to the contextual conditions, and then apply traditional recommendation methods that do not take into account the context over the already filtered data.

For all the baselines, we run the experiments by using the default parameters.

To conclude, we want also point out that it was not possible to further extend the set of the baselines since almost all the related work we previously mentioned verified (at least one of) the following conditions: *(i)* the authors did not provide an implementation of their method; *(ii)* the evaluation focused on error metrics, such as MAE, RMSE, etc., while we focused on ranking; *(iii)* the authors used different datasets to evaluate their approach, thus the results are not comparable. However, regardless this clarification, we can state that our experimental protocol can provide solid findings since we compared our methodology to nine different baselines, each of which can be considered as state-of-the-art in the area of CARS. Moreover, we made available our source code and we thoroughly explained our protocol, in order to make this work extensible and reproducible.

4.1.4. Implementation details

We integrated our algorithm with CARSKit and we run the experiments by using the same splits of the data as well as the same protocol. This guarantees the consistency between the different runs of the experiments. As previously stated, the source code of our recommendation framework is available online[7]. As indicated in [46], the time complexity of the approach is $O(n * k)$, where n is the number of edges in the graph and k is the number of iteration steps. Time complexity is the same for all the configurations, since the amount of edges that are encoded in the graph does not change on varying of the personalization vector.

[7] `https://github.com/swapUniba/CARSKit_PageRank`

Config.	Weights	F1@5	F1@10	NDCG@5	NDCG@10
PPR-I	30/0/70	0.0047	0.0034	0.0085	0.0100
PPR-I	50/0/50	0.0063	0.0069	0.0104	0.0166
PPR-I	80/0/20	0.0035	0.0048	0.0043	0.0091
PPR-C	0/30/70	0.0056	0.0041	0.0127	0.0145
PPR-C	0/50/50	0.0064	0.0053	0.0152	0.0181
PPR-C	0/80/20	**0.0117**	**0.0118**	**0.0178**	**0.0271**
PPR-I-C	50/30/20	0.0031	0.0017	0.0050	0.0050
PPR-I-C	30/50/20	0.0075	0.0066	0.0174	0.0217
PPR-I-C	40/40/20	0.0076	0.0041	0.0121	0.0121

Table 2: Results of Experiment 1 on *TripAdvisor data*. The best-performing configuration is reported in bold. Values in the *weight* column refer to item nodes, context nodes and other nodes, respectively.

4.2. Results of the Experiments

Experiment 1. Results of the first experiment are provided in Table 2, 3 and 4, and report the results obtained on *TripAdvisor*, *DePaul-movie* and *Tijuana* datasets, respectively.

As for TripAdvisor data, we obtained the best F1 and NDCG with the `PPR-C` configuration that gives the greater weight (`80%`) to context nodes. This results suggests that *context* is the most important source of information for this dataset, which plays a key role to provide users with relevant recommendations. Indeed, the algorithm obtained the best performance by introducing a bias towards the nodes that describe the current contextual situation.

It is worth to note that such a configuration does not take into account the specific preferences of the current user. Indeed, `PPR-C` only biases random walks by inducing the algorithm to pass through context nodes. This gives a higher PR score to the items that are highly connected to the nodes that model the active context (that is to say, other items the users liked in that specific contextual setting). Even this can seem as *counter-intuitive* behavior, it is perfectly in line

Config.	Weights	F1@5	F1@10	NDCG@5	NDCG@10
PPR-I	30/0/70	0.0433	0.0512	0.0464	0.0689
PPR-I	50/0/50	0.0485	0.0596	0.0582	0.0863
PPR-I	80/0/20	0.0457	0.0544	0.0524	0.0799
PPR-C	0/30/70	0.0568	0.0582	0.0568	0.0781
PPR-C	0/50/50	0.0529	0.0634	0.0593	0.0896
PPR-C	0/80/20	0.0556	0.0736	**0.0641**	0.0985
PPR-I-C	50/30/20	**0.0675**	**0.0837**	0.0614	**0.1003**
PPR-I-C	30/50/20	0.0525	0.0603	0.0536	0.0800
PPR-I-C	40/40/20	0.0505	0.0671	0.0542	0.0887

Table 3: Results of Experiment 1 on *DePaul-Movie data*. The best-performing configuration is reported in bold. Values in the *weight* column refer to item nodes, context nodes and other nodes, respectively.

with the intuitions behind our methodology.

Indeed, as previously stated, one of the advantages that follow the adoption of PPR lies is the ability of providing good recommendations in *cold-start* situations.

This claim perfectly explains the behavior we observed on TripAdvisor data, since this dataset has the largest number of contextual dimensions and the highest sparsity. Accordingly, to distribute a part of the weight to the nodes that represent the items the user previously liked is not helpful to correctly address random walks, since the paths connecting that nodes to other relevant item nodes are typically too scarce. Conversely, through the `PPR-C` weighting scheme the choices of *other* users that enjoyed an item in the *same* contextual settings can be easily taken into account, and this leads to an improvement of the overall performance of the method.

Next, we can state that an interesting behavior emerges by analyzing the results regarding *DePaul-movie* data that we presented in Table 3. In this case, the overall best results are obtained by the `PPR-I-C` weighting scheme, that gives

importance to both item nodes (50%) and context nodes (30%). This suggests that, differently from what we noted on TripAdvisor data, the preferences of the target user have to be taken into account.

Indeed, by following this strategy, the algorithm is biased in running across random walks that give high consideration to both the *current contextual situation* as well as to the previous *preferences of the user*, encoded in item nodes. These results confirm the idea underlying our methodology, since they showed that the extension of the data model through the introduction of *context nodes* and the distribution of part of the weight to that nodes in order to bias random walks leads to an improvement of the predictive accuracy of recommendation framework.

Furthermore, in this case we can note that the PPR-C weighting schemes obtained worse results. This suggests that when the data are less sparse and when a smaller number of contextual dimension exists, as in DePaul-Movie dataset, it is better to take into consideration also the information that comes from users' preferences. These findings are also confirmed by considering the results we got on *Tijuana Restaurant* data.

However, as for this dataset, the most interesting outcome of the experiment regards the performance of the PPR-I weighting scheme, which resulted as the best-performing one on both NDCG@5 and NDCG@10. By recalling the behavior of such a scheme, it *only* biases the nodes that represent the previous preferences of the users without taking into account the information about the context.

In this case, our conjecture is that PPR-I obtained the best results since the number of contextual dimensions and contextual settings in Tijuana restaurant dataset is very low and the density in the data is already high. Accordingly, it is not necessary to take into consideration the information that comes from the context and the users preferences can drive the process alone.

Overall, we can state that these results validate one of the conjectures this work relies on, since it confirmed how effective PPR can be to provide users with good recommendations in cold-start settings. Moreover, the strategy we

Config.	Weights	F1@5	F1@10	NDCG@5	NDCG@10
PPR-I	30/0/70	0.1975	0.1722	0.2380	0.3007
PPR-I	50/0/50	0.1651	0.1927	0.1819	0.2836
PPR-I	80/0/20	0.2010	0.2038	**0.2586**	**0.3567**
PPR-C	0/30/70	0.1888	0.1403	0.2342	0.2707
PPR-C	0/50/50	0.1960	0.1648	0.2449	0.3075
PPR-C	0/80/20	0.2155	0.1765	0.2513	0.3049
PPR-I-C	50/30/20	**0.2163**	**0.2095**	0.2511	0.3125
PPR-I-C	30/50/20	0.1959	0.1771	0.2471	0.3047
PPR-I-C	40/40/20	0.1853	0.1856	0.2094	0.3106

Table 4: Results of Experiment 1 on *Tijuana Restaurants* data. The best-performing configuration is reported in bold. Values in the *weight* column refer to item nodes, context nodes and other nodes, respectively.

adopted to distribute the weights makes the algorithm solid and capable of providing good suggestions even when more data are available.

Experiment 2. Next, we compared the best-performing configurations that emerged from Experiment 1 to the baselines implemented in CARSKit library[8] and to a non-personalized version of PR that does not use any weighting scheme to bias random walks.

As shown in Table 5, the first finding of this experiment is that our framework based on PPR obtained the best results on *TripAdvisor* data.

Indeed, our configurations obtained a 216% increase (from *0.0037* to *0.0117*) with respect to traditional PR and a 85% increase (from *0.0065* to *0.0117*) with respect to the best-performing baseline, that is to say, CSLIM.

This outcome further confirms the goodness of the intuition behind our approach. Indeed, as we already explained, context-aware techniques are really prone to cold-start in highly sparse recommendation scenarios, such as TripAdvisor. Conversely, our approach showed its robustness even in these extreme

[8]https://github.com/irecsys/CARSKit

	TripAdvisor		**DePaul-Movie**		**Tijuana Rest.**	
Config.	F1@5	NDCG@5	F1@5	NDCG@5	F1@5	NDCG@5
PPR (best)	**0.0117**	**0.0178**	0.0675	0.0641	**0.2155**	**0.2586**
PR	0.0037	0.0038	0.0518	0.0546	0.1900	0.2278
CAMF_CI	0.0000	0.0000	0.0704	0.0710	0.1043	0.1048
CAMF_CU	0.0002	0.0003	0.0522	0.0509	0.1165	0.1331
CAMF_C	0.0017	0.0040	0.0756	0.0869	0.1737	0.2287
CAMF_CUCI	0.0049	0.0090	0.0806	0.0837	0.1207	0.1566
CSLIM_C	0.0065	0.0110	**0.0998**	**0.1202**	0.1105	0.1142
CSLIM_CI	0.0018	0.0040	0.0369	0.0376	0.1061	0.1078
CSLIM_CU	0.0017	0.0030	0.0686	0.0845	0.1147	0.1711
UserSplitting	0.0018	0.0013	0.0528	0.0460	0.1335	0.1472
ItemSplitting	0.0011	0.0016	0.0530	0.0525	0.1738	0.2101

Table 5: Results of Experiment 2 for all the datasets. The best-performing configurations are reported in bold.

cases, since the results clearly show that our graph-based representation along with the adoption of particular weighting schemes to bias random walks can lead to a large improvement to the overall results.

Conversely, when the data are less sparse, as it happens on *DePaul-Movie* dataset, techniques such as CSLIM are still more precise. In this case it is likely that the ability of learning some relationships and dependencies between contexts and preferences, that is one of the characteristics of more sophisticated algorithms such as CSLIM, allows the algorithm to outperform PPR. However, it is worth to state that our results are still better than those obtained by simple pre-filtering techniques based on UserSplitting and ItemSplitting, even on these data.

Finally, the results regarding *Tijuana Restaurant* provide other interesting outcomes. By referring to the statistics of the dataset we reported in Table 1, it emerges that this dataset is the less sparse one. Accordingly, we would have expected a behavior similar to that we noted on DePaul-movie data, that is to say, to have a matrix factorization-based approach obtaining the best results. Contrary to what was expected, our framework obtained the best results.

This can be justified by the fact that the number of contextual dimensions encoded in this dataset is very small, thus the adoption of a very sophisticated recommendation algorithm based on machine learning is not necessary and easier strategies to provide recommendation can be equally effective. Such an intuition is confirmed by analyzing the results of the baselines, that showed that the second best results are obtained by adopting simpler *splitting* approaches.

However, for our part, these results further confirm the robustness and the flexibility of our framework, which is able to provide good recommendations in experimental settings that are very different one from each other. In this case, our framework has the ability of biasing random walks in order to mainly consider user preferences and to give less importance to the context, and this choice leads to the overall best recommendations among the available methods.

4.3. Take-Home Messages

In this section we try to summarize and outline the main findings emerging from the experiments. In order to better emphasize the effectiveness of the framework, we link our discussion to the hallmark of the current work we already presented in Section 3.3.

1. **Adoption of a graph-based representation.**As previously stated, the adoption of a uniform representation based on graphs was a strong point of our methodology. As expected, experiments showed the effectiveness of the approach, which obtained results better (or at least in line) than several state of the art techniques. The capability of being effective in a broad range of experimental settings confirmed the intuition behind our strategy and supported our claims.
2. **Robustness in highly sparse experimental settings.** In *highly sparse* recommendation scenarios, our algorithm provides the best results. This outcome mainly depends on the novel *data model* we introduced in this work, which is based on a tripartite representation that models users, items and contexts. The paths encoded in this graphs allow to identify relevant items even when the connections in the graph are just a few. It is likely that our choice of modeling each context through one *single* node (rather than modeling the *combination* of contexts, as other related work do) has been helpful to obtain these results, since it leads to a *more connected* graph where more paths connecting the nodes exist;
3. **Flexibility of the recommendation framework.** As previously stated, in highly sparse recommendation scenarios our strategy largely overcomes the other baselines. Conversely, in *less sparse* recommendation scenarios, our algorithm still overcomes simple context-aware recommendation methods (as splitting methods) but is beaten by more sophisticated techniques. Finally, when the number of contextual dimensions is small, regardless of the sparsity, our framework beat all the baseline. Overall, the experimental session confirmed the flexibility of the framework. Indeed, our method-

ology allows to set a different bias based on the different characteristics of the dataset, by deciding whether it is better to give more importance to the contextual settings alone, as in highly sparse recommendation scenarios, or to the preferences of the users, regardless of the specific context in which they have been expressed. This large space of option let our framework adapting to different settings and to obtain satisfying results in a broad range of scenarios. This distinguishes our methodology from other traditional CARS techniques.

5. Conclusions and Future Work

In this work we presented a novel graph-based context-aware recommendation framework that uses PPR to provide users with recommendations. Our methodology is based on the following intuitions: *(i)* first, we extend the original bipartite graph-based data model that relies on users and items nodes by introducing *context* nodes. *(ii)* next, we introduce different strategies to distribute the weights in PPR, in order to bias random walks towards specific nodes.

Clearly, in our case we try to induce the algorithm to give high scores to nodes highly connected to the previous preferences of the user as well as to the current contextual situation. As shown in the experiments, our strategy obtained results in line with the current literature and resulted as more effective in most of the cold-start settings.

As future work, we will try to introduce in our data model *descriptive* features encoding some characteristics of users and items, such as user-generated content [47, 48], in order to evaluate also the impact of these information on the overall performance of the framework. Moreover, we will introduce weighted edges, in order to reward items that are more frequently consumed in specific contextual settings and we will assess to what extent PR and PPR algorithms are prone to *popularity bias* [49]. Finally, another interesting research direction lies in the adoption of more accurate representations, such as those based on *binary codes* [50], and the introduce of elements borrowed from *cognitive*

computing [51] in the overall framework.

References

[1] P. Resnick, H. R. Varian, Recommender systems, Communications of the ACM 40 (3) (1997) 56–58.

[2] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, Recommender systems: an introduction, Cambridge University Press, 2010.

[3] J. Grau, Personalized product recommendations: Predicting shoppers' needs (2009).

[4] G. Adomavicius, A. Tuzhilin, Context-aware recommender systems, in: Recommender systems handbook, Springer, 2011, pp. 217–253.

[5] G. Adomavicius, R. Sankaranarayanan, S. Sen, A. Tuzhilin, Incorporating contextual information in recommender systems using a multidimensional approach, ACM Trans. Inf. Syst. 23 (1) (2005) 103–145. doi:10.1145/1055709.1055714.
URL http://doi.acm.org/10.1145/1055709.1055714

[6] F. Ricci, L. Rokach, B. Shapira, Recommender systems: introduction and challenges, in: Recommender systems handbook, Springer, 2015, pp. 1–34.

[7] T. H. Haveliwala, Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search, IEEE Trans. Knowl. Data Eng. 15 (4) (2003) 784–796.

[8] S. Lee, S.-i. Song, M. Kahng, D. Lee, S.-g. Lee, Random walk based entity ranking on graph for multidimensional recommendation, in: Proceedings of the fifth ACM conference on Recommender systems, 2011, pp. 93–100.

[9] W. Yao, J. He, G. Huang, J. Cao, Y. Zhang, A graph-based model for context-aware recommendation using implicit feedback data, World wide web 18 (5) (2015) 1351–1371.

[10] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: bringing order to the web.

[11] A. I. Kovacs, H. Ueno, Recommending in context: A spreading activation model that is independent of the type of recommender system and its contents, in: Proc. 2nd International Workshop on Web Personalisation, Recommender Systems and Intelligent User Interfaces (WPRSIUI 06), Citeseer, 2006.

[12] Z. Zhang, L. Li, A research paper recommender system based on spreading activation model, in: The 2nd International Conference on Information Science and Engineering, IEEE, 2010, pp. 928–931.

[13] S. Papneja, K. Sharma, N. Khilwani, Context-aware personalized content recommendation using ontology based spreading activation, International Journal of Information Technology 10 (2) (2018) 133–138.

[14] Z. Bahramian, R. A. Abbaspour, C. Claramunt, A context-aware tourism recommender system based on a spreading activation method, International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences 42.

[15] A. Hotho, R. Jäschke, C. Schmitz, G. Stumme, K.-D. Althoff, Folkrank: A ranking algorithm for folksonomies, in: LWA, Vol. 1, 2006, pp. 111–114.

[16] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, M. Aly, Video suggestion and discovery for YouTube: taking Random Walks through the view graph, in: Proceedings of the 17th International Conference on World Wide Web, ACM, 2008, pp. 895–904.

[17] T. Bogers, Movie recommendation using Random Walks over the contextual graph, in: Proc. of the 2nd Intl. Workshop on Context-Aware Recommender Systems, 2010.

[18] X. Li, H. Chen, Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach, Decision Support Systems 54 (2) (2013) 880–890.

[19] M. de Gemmis, P. Lops, G. Semeraro, C. Musto, An investigation on the serendipity problem in Recommender Systems, Information Processing and Management 51 (5) (2015) 695 – 717. doi:http://dx.doi.org/10.1016/j.ipm.2015.06.008.
URL http://www.sciencedirect.com/science/article/pii/S0306457315000837

[20] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, S. Wang, Learning graph-based poi embedding for location-based recommendation, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 15–24.

[21] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, KGAT: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 950–958.

[22] C. Musto, P. Lops, M. de Gemmis, G. Semeraro, Semantics-aware recommender systems exploiting linked open data and graph-based features, Knowledge-Based Systems 136 (2017) 1–14.

[23] C. Musto, G. Semeraro, M. de Gemmis, P. Lops, Tuning personalized pagerank for semantics-aware recommendations based on linked open data, in: European Semantic Web Conference, Springer, 2017, pp. 169–183.

[24] S. Ji, S. Pan, E. Cambria, P. Marttinen, P. S. Yu, A survey on knowledge graphs: Representation, acquisition and applications, arXiv preprint arXiv:2002.00388.

[25] E. Palumbo, G. Rizzo, R. Troncy, E. Baralis, M. Osella, E. Ferro, Knowl-

edge graph embeddings with node2vec for item recommendation, in: European Semantic Web Conference, Springer, 2018, pp. 117–120.

[26] W. Song, Z. Duan, Z. Yang, H. Zhu, M. Zhang, J. Tang, Explainable knowledge graph-based recommendation via deep reinforcement learning, arXiv preprint arXiv:1906.09506.

[27] M. Caro-Martinez, G. Jimenez-Diaz, Similar users or similar items? comparing similarity-based approaches for recommender systems in online judges, in: International Conference on Case-Based Reasoning, Springer, 2017, pp. 92–107.

[28] J. Herlocker, J. Konstan, Content-independent task-focused recommendation, IEEE Internet Computing 5 (6) (2001) 40–47.

[29] A. Karatzoglou, X. Amatriain, L. Baltrunas, N. Oliver, Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering, in: Proceedings of RecSys 2010, ACM, 2010, pp. 79–86.

[30] L. Baltrunas, B. Ludwig, F. Ricci, Matrix factorization techniques for context aware recommendation, in: Proceedings of the fifth ACM conference on Recommender systems, 2011, pp. 301–304.

[31] B. Hidasi, D. Tikk, Fast als-based tensor factorization for context-aware recommendation from implicit feedback, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2012, pp. 67–82.

[32] D. Kim, C. Park, J. Oh, S. Lee, H. Yu, Convolutional matrix factorization for document context-aware recommendation, in: Proceedings of the 10th ACM Conference on Recommender Systems, 2016, pp. 233–240.

[33] X. Xin, B. Chen, X. He, D. Wang, Y. Ding, J. Jose, Cfm: convolutional factorization machines for context-aware recommendation, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 3926–3932.

[34] Y. Liu, F. Shen, J. Zhao, Pairwise interactive graph attention network for context-aware recommendation, arXiv preprint arXiv:1911.07429.

[35] Y. Zheng, B. Mobasher, Context-aware recommendations, in: Collaborative Recommendations: Algorithms, Practical Challenges and Applications, World Scientific Publishing Company, 2018, pp. 173–202.

[36] H. Bagci, P. Karagoz, Context-aware friend recommendation for location based social networks using random walk, in: Proceedings of the 25th international conference companion on world wide web, 2016, pp. 531–536.

[37] H. Wu, K. Yue, X. Liu, Y. Pei, B. Li, Context-aware recommendation via graph-based contextual modeling and postfiltering, International Journal of Distributed Sensor Networks 11 (8) (2015) 1–16.

[38] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995. doi:10.1017/cbo9780511814075.
URL https://doi.org/10.1017/cbo9780511814075

[39] S. Ilarri, R. Trillo-Lado, R. Hermoso, Datasets for context-aware recommender systems: Current context and possible directions, in: 2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW), IEEE, 2018, pp. 25–28.

[40] H. Steck, Evaluation of recommendations: rating-prediction and ranking, in: Proceedings of the 7th ACM conference on Recommender systems, 2013, pp. 213–220.

[41] A. Bellogin, P. Castells, I. Cantador, Precision-oriented evaluation of recommender systems: an algorithmic comparison, in: Proceedings of the fifth ACM conference on Recommender systems, 2011, pp. 333–336.

[42] Y. Zheng, B. Mobasher, R. Burke, Carskit: A java-based context-aware recommendation engine, in: 2015 IEEE International Conference on Data Mining Workshop (ICDMW), IEEE, 2015, pp. 1668–1671.

[43] Y. Zheng, B. Mobasher, R. Burke, Cslim: Contextual slim recommendation algorithms, in: Proceedings of the 8th ACM Conference on Recommender Systems, 2014, pp. 301–304.

[44] X. Ning, G. Karypis, Slim: Sparse linear methods for top-n recommender systems, in: 2011 IEEE 11th International Conference on Data Mining, IEEE, 2011, pp. 497–506.

[45] L. Baltrunas, F. Ricci, Experimental evaluation of context-dependent collaborative filtering using item splitting, User Modeling and User-Adapted Interaction 24 (1-2) (2014) 7–34.

[46] Y. Fujiwara, M. Nakatsuji, M. Onizuka, M. Kitsuregawa, Fast and exact top-k search for random walk with restart, arXiv preprint arXiv:1201.6566.

[47] P. Lops, M. de Gemmis, G. Semeraro, C. Musto, F. Narducci, M. Bux, A semantic content-based recommender system integrating folksonomies for personalized access, Studies in Computational Intelligence, Volume 229 27.

[48] C. Musto, F. Narducci, M. De Gemmis, P. Lops, G. Semeraro, Star: a social tag recommender system, in: Proceedings of the 2009th International Conference on ECML PKDD Discovery Challenge-Volume 497, 2009, pp. 215–227.

[49] H. Abdollahpouri, R. Burke, B. Mobasher, Managing popularity bias in recommender systems with personalized re-ranking, arXiv preprint arXiv:1901.07555.

[50] Y. Li, S. Wang, Q. Pan, H. Peng, T. Yang, E. Cambria, Learning binary codes with neural collaborative filtering for efficient recommendation systems, Knowledge-Based Systems 172 (2019) 64–75.

[51] C. Angulo, Z. Falomir, D. Anguita, N. Agell, E. Cambria, Bridging cognitive models and recommender systems, Cognitive Computation (2020) 1–2.