

# Harnessing the Power of Ego Network Layers for Link Prediction in Online Social Networks

Mustafa Toprak, Chiara Boldrini, Andrea Passarella, and Marco Conti

**Abstract**—Being able to recommend links between users in online social networks is important for users to connect with like-minded individuals as well as for the platforms themselves and third parties leveraging social media information to grow their business. Predictions are typically based on unsupervised or supervised learning, often leveraging simple yet effective graph topological information, such as the number of common neighbors. However, we argue that richer information about personal social structure of individuals might lead to better predictions. In this paper, we propose to leverage well-established social cognitive theories to improve link prediction performance. According to these theories, individuals arrange their social relationships along, on average, five concentric circles of decreasing intimacy. We postulate that relationships in different circles have different importance in predicting new links. In order to validate this claim, we focus on popular feature-extraction prediction algorithms (both unsupervised and supervised) and we extend them to include social-circles awareness. We validate the prediction performance of these circle-aware algorithms against several benchmarks (including their baseline versions as well as node-embedding- and GNN-based link prediction), leveraging two Twitter datasets comprising a community of video gamers and generic users. We show that social-awareness generally provides significant improvements in the prediction performance, beating also state-of-the-art solutions like *node2vec* and *SEAL*, and without increasing the computational complexity. Finally, we show that social-awareness can be used in place of using a classifier (which may be costly or impractical) for targeting a specific category of users.

**Index Terms**—link prediction, social circles, Dunbar’s model, Twitter

## I. INTRODUCTION

In early 2021, the number of social media users worldwide has reached 4.2 billion, up 13% from the previous year [1]. Social media are now large ecosystems, where we gather to interact with people we already know or to join communities of people that share our interests. Clearly, being able to pair together people with similar interests is crucial for social-driven and content-driven platforms like Facebook, Twitter, Instagram, Youtube, etc. This spurred intense research on link

M. Toprak, C. Boldrini, A. Passarella and M. Conti are with IIT-CNR, Via G. Moruzzi 1, 56124, Pisa, ITALY e-mail: first.last@iit.cnr.it

This work was partially funded by the SoBigData++, HumaneAI-Net, MARVEL, and OK-INSAIL projects. The SoBigData++, HumaneAI-Net, and MARVEL projects have received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreements No 871042, No 952026, No 957337, respectively. The OK-INSAIL project has received funding from the Italian PON-MISE program under grant agreement ARS01 00917. This work was also supported by the CHIST-ERA grant CHIST-ERA-19-XAI-010, funded by MUR (grant number not yet available).

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

prediction, whereby algorithms suggest potential new links with users that are *similar* to each other. Solving this problem is equivalent to finding a needle in a haystack: online social networks (OSN) feature millions of users, the number of potential links between them grows with the square of the number of users, but the actually existing links are quite sparse.

Link prediction algorithms can be broadly divided into *feature extraction* methods and *feature learning* methods [2]. The former are based on predefined similarity metrics/heuristics, typically computed using topological information about the network, be it local information (such as the number of common neighbors) or global one (such as the length of paths connecting two nodes). Despite their apparent simplicity, local similarity-based prediction algorithms perform generally well, even when compared to more complex approaches [3], and are also computationally efficient. Recently, various new link prediction methods (leveraging graph neural networks [4], [5], [6] and network embeddings [7], [8]) have been proposed, which automatically learn latent features that reflect the relevant structure of the graph, and, for this reason, are denoted as feature learning methods. These new methods have been shown to perform extremely well on the link prediction task [7], [5].

One of the well-known mechanisms that drive link formation is the *homophily* principle, which states that individuals tend to bond with those that are more similar to them [9]. Common-neighbors-based algorithms leverage this principle for predicting new links. However, findings from anthropology show that not all bonds are equal: some relationships are just acquaintances and do not imply a significant cognitive engagement. With respect to the meaningful relationships, each individual organises them into concentric *social circles* where the intimacy progressively decreases from the innermost to the outermost circle [10], [11]. This model of how humans organise their relationships stems from the *social brain hypothesis*, which links our social life to cognitive constraints related to the size of our neocortex [12]. This hierarchical social structure is known to impact significantly on who we trust [13], on the way information spreads in online social networks [14], and on the diversity of information that can be acquired by users [15]. This layered social structure is typically represented using an ego network graph [16], [17], [18], [10], in which the individual, referred to as *ego*, is at the center of the graph, and the edges connect her to the peers (called *alters*) with which she interacts. Figure 1 illustrates the classical ego network structure. Please note that the second layer includes all the alters in the first layer, the third layer

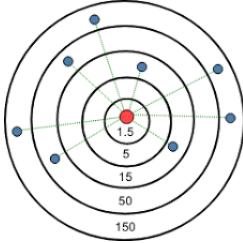


Fig. 1: Layered structure of the ego network. The red dot at the center corresponds to the ego.

includes all the alters in the first and second layers, and so on. The ego-alter tie strength is typically computed as a function of the frequency of *interactions* between the ego and the alter. The ego network structure is characterised by a striking regularity: the typical sizes of the layers are 1.5, 5, 15, 50, 150, with an approximately constant ratio of 3 between the size of consecutive layers. Online relationships have been shown to exhibit similar regularities [19], [20], thus proving that friendship links (which can be in the thousands for online social networks) not associated with online interactions do not change the innate social structure of humans.

Since social links are not all the same, simply considering the set of common neighbors (or analogous topological metrics), disregarding the importance of each social link, might not give a precise estimate of the homophily between two individuals. Thus, in this paper, we set out to improve feature-extraction link prediction algorithms with information about the different importance of social relationships. To this end, we exploit well-established models in the anthropology literature that describe the relative importance of our social bonds by grouping them into different social circles. Please note that the social circles we refer to are very loosely related to the concept of communities in social networks. In fact, social circles are (local) groupings entirely from the perspective of the ego, they include only nodes with which the ego is directly connected, and the links between the alters (which are crucial for regular communities) do not play a role at all here. Note also that our goal is not to propose a new prediction algorithm, but to add social-awareness to popular prediction solutions. To the best of our knowledge, the only other work in the literature that investigates ego networks in relation to link prediction is [21]. However, in the context of that work, an ego network is a different structure from the one studied here (only the links between the alters are considered), social cognitive models are not leveraged, and the effect of using information on the different social circles for link prediction is not investigated.

While the general problem of link prediction entails recommending any of the useful links that a user may want to create with other users, in certain applications we may want to recommend links between users that share a specific common interest together. Predicting links in such contextualised communities might be much more relevant (and valuable) than predicting links between generic users. This could be the case, for example, of a third party service that wants to connect users interested in a specific topic. In this paper, we showcase one application of this concept by focusing

on a set of gamers on Twitter, and we investigate how to make new link recommendations leveraging key properties of the way users allocate their cognitive capacity to social relationships. As discussed in Section III, we believe that focusing on such a target set of users, characterised by a specific type of online engagement, is particularly significant for link recommendation in online social networks. Then, we apply the same method to generic Twitter users, in order to assess the efficacy of the proposed solution also beyond the contextualised communities settings.

The key findings presented in the paper are the following:

- In the vast majority of cases, regardless of the prediction approach (unsupervised or supervised), the specific heuristic or learning algorithm, and the metric (precision, AUC, F1 score) considered, leveraging social circles information outperforms the corresponding baseline in which circles are ignored.
- The contribution of the different social circles to the achieved precision varies with the recommendation algorithm considered. In unsupervised similarity-based approaches, innermost circles help policies with little or no penalization of high degree nodes, while outermost circles boost the performance of algorithms relying on high-degree nodes penalization. The precision of supervised strategies is generally considerably increased when only the innermost circle, corresponding to the small set (composed of 3-5 people typically) of the most intimate relationships, is considered.
- Supervised link prediction that uses circle-aware explicit features is able to beat feature-learning link prediction algorithms like *node2vec* and *SEAL*, which exploit node embedding and graph neural networks, respectively. This happens when the most intimate circle is leveraged, which comprises only the two or three strongest relationships of the ego. Thus, using knowledge on a few common strong ties is more effective than complex black box approaches.
- Leveraging social circles information provides the same performance as using additional classifiers on nodes, which might be impractical or costly to use.

The rest of the paper is organised as follows. We review the related work in Section II. In Section III we present the two datasets that we leverage in our analysis. The ego-aware unsupervised and supervised link prediction approaches based on feature extraction methods that we study, and the feature learning methods that we use as benchmarks are presented Section IV. Their prediction performance is then evaluated in Section V. Finally, in Section VI we conclude the paper. An analysis of the computational complexity of the proposed approach is provided in Appendix F.

## II. RELATED WORK

### A. Dunbar's model and ego networks

According to the *social brain hypothesis* from anthropology [12], the social life of primates is constrained by the size of their neocortex. Specifically, for humans, the typical group size is estimated around an average of 150 members, a limit that goes under the name of Dunbar's number. This

limit is related to the cognitive capacity that humans are able to allocate to nurturing their social relationships. The 150 *friends*, in fact, do not include acquaintances, but only people with which a coherent quality relationship is entertained. In a modern society, this entails, at the very least, exchanging birthday or Christmas cards every year. Unsurprisingly, humans are not fair in how they distribute their cognitive attentions among the 150 important persons they have around them. To the contrary, it is possible to group our social relationships in circles of increasing intimacy, as illustrated in Figure 1. Typically, each of us has at least four circles of intimacy [10], [11]: the innermost one (*support clique*) includes our close family, the *sympathy group* comprises all those people whose death tomorrow would leave us deeply affected, the *affinity group* includes colleagues, extended family, people you hang out with, and finally the *active network* is made up of all the people you meaningfully interact with at least once a year. The circles are conventionally concentric, with the inner layers contained in the outer ones. Many people also feature an additional inner layer (contained in the support clique) comprising on average 1.5 alters for which they have a very high emotional investment [20].

The social brain theory was developed for the offline world, in which relationships were nurtured via face-to-face interactions, letters, or landline phone calls. Many had postulated that this theory would not carry over to the online world, where OSN allow us to engage conveniently with a huge number of people scattered across the globe. However, Dunbar's model stood the test of the cyberworld as well: Dunbar's number hold for email communications [22], mobile phone calls [23], and Twitter [19]. More importantly, the same layered structure of social relationships has emerged on both Facebook and Twitter [20]. These findings are extremely important. In fact, since ego network structures are known to impact significantly on the way information spreads in OSN, and on the diversity of information that can be acquired by users [15], embedding these models of human cognition into services for OSN may drastically improve the quality of service provided to the users.

### B. Link prediction

The literature on link prediction algorithms is large. Here we only summarise the main approaches and we refer the interested readers to [3], [24], [2] for detailed surveys on the topic. Following the taxonomy proposed by [2], we discuss separately feature extraction and feature learning methods

1) *Feature extraction methods*: Similarity-based methods make up the largest class of link prediction algorithms proposed in the literature. The rationale of this approach is that nodes are more likely to form links with nodes that are similar to them. This idea is grounded in the widespread and well-documented social phenomenon of homophily [9]. Algorithms in this class differ in how they define the similarity between nodes. We can broadly distinguish them based on whether they use local information, global information, or a hybrid combination of the two (this approach is one of the least popular, hence we will not treat it further). For a given node pair, local similarity-based solutions rely on node neighborhood-related structural information, such as the number of common

neighbors [25], an inverse function of the degree of common neighbors [26], [27], or a Preferential Attachment index [25]. Local similarity-based approaches are very efficient, even on large networks, due to their easy parallelization. Their main theoretical limitation is that they are able to predict only new links between neighbors-of-neighbors. Their counterpart, global similarity-based approaches, rely on metrics computed considering the whole network topology. These metrics focus typically on the possible paths inside the network, such as the shortest paths [28], paths with different lengths by means of the Katz Index [25], random walks [29], or community membership [30].

Algorithmic methods map the link prediction problem into well-known algorithmic approaches. Within this category, classifier-based methods treat the link prediction problem as a binary classification problem [31], [32], [33], [5]. Each node pair can be characterised with a variety of attributes, including the similarity-based heuristics discussed above. This labelled set can be fed to virtually any classifier (such as decision trees, SVM, k-nearest neighbors, random forest, neural networks). The advantage of this approach is that it can be extended and adapted to any new attribute that one wants to test, and that it generally significantly outperforms purely unsupervised similarity-based methods [31], [33], [34].

Preprocessing methods are considered meta-approaches, as they are intended to be used in conjunction with other algorithms, for which they provide preprocessing/pre-filtering intended to remove some noise in the network. As an example, the clustering method discussed in [28] suggests the removal of the weakest links (those between nodes with few or zero common neighbors). The solution that we propose in this paper falls into this category. The meta-approach that we investigate is based on considerations related to how people distribute their social capacity across their relationships, rather than on pure graph-related properties.

Orthogonally to the above classification, we can also distinguish between approaches that consider the weights of the links or not. Among the former, we mention [27], [35]. As discussed in [36], though, the performance of weighted indices is often worse than their unweighted counterpart. For this reason, in this work we do not consider the weight of links besides what is needed to compute the ego network structure.

2) *Feature learning methods*: Feature learning methods map the graph into a low-dimensional feature space. The difference between feature extraction and feature learning methods is that, in the latter, the features are learnt by the system and not hand-engineered. The mapping, or graph representation, can be learned and optimized via both supervised and unsupervised methods [2]. The random walk methods use graph exploration methods such as breadth-first search (BFS), depth-first search (DFS), and random walks to capture features and node properties such as centrality, being a hub, or community membership. *DeepWalk* [37] and *node2vec* [7] are the most popular methods of this category. For the link prediction task, the node feature vectors (node embeddings) are transformed into edge feature vectors (edge embeddings) using operators like the Hadamard product or cosine similarity. Extracted edge embeddings are fed to learning algorithms

(such as SVM, regression methods, deep neural networks) to train a model that is used to predict future links. These methods are considered semi-supervised, since the information on the existence/non-existence of a link is already present in the studied graph.

Graph neural network (GNN) models leverage neural networks to map the graph structure to a low dimensional vector space. Graph differentiable pooling, graph autoencoders, and graph neural networks have been used for the link prediction algorithms based on neural networks. In this study, we compare against the popular GNN-based link prediction algorithm SEAL [5], which relies on a CNN architecture. Note that network embedding methods and graph neural networks (GNN) are highly related to each other [38], as they both map the network to a lower-dimensional representation space. However, GNNs allow for greater flexibility, by leveraging node features in the encoder, by sharing parameters between nodes in the encoder, and by fixing the transductivity problem of shallow embeddings [39]. From the link prediction perspective, GNNs typically provide end-to-end frameworks (i.e., the learnt presentation is optimised for the link prediction problem), while graph embedding methods capture topology-level properties that are then used to train a separate classifier for link prediction.

### III. THE DATASETS

As discussed in Section I, in this paper we investigate social circle-aware link prediction considering both a community of users with shared interests (gaming, in our case) and generic users. Before presenting the algorithms for predictions in Section IV, here we first introduce the datasets that we use as case study. Without loss of generality, some definitions required in the prediction algorithms will be discussed with reference to these datasets. However, please note that each such definition can be used as-is for any other dataset describing different communities, and the algorithm works unchanged in all such cases.

In the *gaming-related dataset*, we consider as shared topic of interest “indie games”. An independent video game, or indie game, is a video game that is often created without the financial support of a publisher, although some games funded by a publisher are still considered “indie”. Indie games often focus on innovation and rely on digital distribution. Growing a community of interested users is thus a critical success factor for them. The goal of our link prediction task is to recommend users interested in indie games to other users interested in indie games. Our gaming-related dataset consists of 8,932 users (labelled as *gamers*) engaging in game-related conversations. We have collected (June 2018) their timeline (most recent 3200 tweets) using the Twitter Search API. Details on the collection process are provided in Appendix A. In the *generic user dataset*, we focus on a group of users whose interests are not associated with a specific domain. To this aim, we use a dataset presented in [40], containing the timelines (most recent 3200 tweets) of 1,930,802 Twitter users obtained by means of a snowball sampling starting from US President Barack Obama in November 2012. Please refer to [40] for further details on the collection process.

Note that we use two Twitter datasets because Twitter is especially amenable to the computation of ego networks: social interactions are typically public and can be easily downloaded. For this reason, the vast majority of the literature on online ego networks (Section II-A) leverages Twitter data. Facebook interactions would be similarly suitable but they are not public, hence could not be accessed. The social network graphs available to the research community do not typically include information on the frequency of interactions, thus ego networks cannot be extracted.

#### A. Extracting the ego networks

We filter the 8,932 gamers and 1,930,802 generic users and extract their ego network (including their social circles) using the same methodology as in [41], which we briefly recollect here. We simply need a weighted social graph, whose edge weights correspond to the contact frequency between the corresponding nodes. In order to extract reliable information, we filter users according to the same policy used in [41]. We only consider users whose Twitter activity is regular (i.e., they post, on average, at least one tweet every 3 days for at least 50% of the total number of months of their activity) and stationary (i.e., they are not in their initial stage of engagement with the Twitter platform, which typically features a transient spike of activity). The strength of ego-alter relationships is inferred from the frequency of direct tweets (mentions, reply, or retweets) between the ego and the alters. The optimal number of social circles per users is obtained using the Mean Shift clustering algorithms to group such frequencies. The code for computing the ego networks can be found at <https://egonetworks.readthedocs.io/en/latest/>.

Approximately one third of the gamers passed all the filters described above, and this left us with 3,061 gamers with reliable ego network information. On the other hand, approximately 7% of the generic users passed all the filters, amounting to 148,105 generic users with reliable ego network information. We further filter them as suggested in [25] by considering only nodes that are connected to the giant component. Also, in order to have a generic user dataset with a similar size as that of the gaming-related dataset and to reduce its computational intensity, we sampled 3,000 egos from the giant component of the generic users network by snowball sampling [42].

We denote the set of nodes for which we have ego networks as  $V_e$ . In the gaming-related dataset all egos are gamers. Vice versa, in the generic users dataset, all egos are generic users. An exploratory analysis of the ego networks in our datasets, which is substantially in agreement with the results from the related literature [41], [20], can be found in Appendix C. Note that, when we focus on an ego  $i$ , its alters can now be associated with the social circle of  $i$  to which they belong. Note also that we do not typically have the ego network of the alters. In the gaming-related dataset, the alters of ego  $i$  can be generic users, other gamers, or even games. This implies that some nodes are *domain-specific* (in our case, gaming-related), while others are generic. Domain-specific nodes are homogeneous, since they share a common interest. Summarising, we have

TABLE I: Gaming-related dataset

# of gamer nodes $ V_e $	2,995
# of domain-specific nodes $ V_d $	70,859
# of all nodes $ V  =  V_e \cup V_d \cup V_n $	470,485
# of gamer edges $ E_e $	2,614
# of domain-specific edges $ E_d $	154,581
# of edges $ E $	1,004,011

TABLE II: Generic users dataset

# of ego nodes $ V_e $	3000
# of all nodes $ V  =  V_e \cup V_n $	278,510
# of ego edges $ E_e $	7,158
# of edges $ E $	567,739

three classes of nodes in the gaming-related dataset: gamer users with ego network ( $V_e$ ), regular (unlabelled) users without ego network ( $V_n$ ), and domain-specific users that may or may not have an ego network ( $V_d$ , including games and gamers for which ego network information is not available or not reliable because they didn't satisfy the filters discussed previously). On the other hand, in the generic user dataset, all alters of ego  $i$  are also generic users. Generic nodes make prediction more challenging with respect to the case of homogeneous nodes (as, e.g., in [25]), because they tend to differ more from each other. In the generic user dataset, we can identify only two classes of nodes: generic users with ego network ( $V_e$ ) and generic users without ego networks ( $V_n$ ) that are basically alters.

### B. Dataset summary

The final datasets are summarised in Table I and Table II. The imbalance ratio, i.e., the ratio between the negatives (potential links that do not exist in practice) and the positives (links actually existing) in the class of both gamers and generic users is around  $10^4:1$ , which is quite high. This is a well-known problem in link recommendation for online social networks (remember from Section I that links are sparse). This implies that special care should be taken in the evaluation, as explained in detail in Section V-B.

## IV. LINK PREDICTION BASED ON SOCIAL CIRCLES

We denote with  $\mathcal{G} = (V, E)$  the graph modelling the relationships in our dataset. Since our goal is to evaluate the contribution of social circles to link prediction, we will focus on the subset of nodes in  $V$  for which social circles can be computed. Thus, predictions will be made for the set of possible edges between nodes in  $V_e$ .

### A. Similarity-based unsupervised learning with circle-awareness

In order to showcase the effect of social circles for unsupervised link prediction, we focus on the simple, yet effective class of similarity-based approaches. Like for all similarity-based link prediction algorithms, the goal of the proposed algorithm is to associate each non-existing link between two users  $i, j$  in  $V_e$  with a score proportional to the likelihood that the link will actually be formed in the future. Hence, the goal of the link predictor is to associate with each edge

in  $V_e \times V_e - E_e$  a probability/confidence that the link will indeed appear in the future. Qualitatively, this is equivalent to *suggesting users to other users* for possible interactions, friendships, etc.

In order to study the effect of the social circles on link prediction, we propose to *slice* the social graph based on the membership to a specific circle (recall that our concept of social circles is different from that of communities). Let us focus on a user  $i \in V_e$ . We denote with  $C_1^{(i)}, C_2^{(i)}, \dots$ , the sets comprising the neighbors of  $i$  that belong to  $i$ 's first, second,  $\dots$ , social circle, respectively. We have discussed how to obtain these circles in Section III-A. Thus, the graph  $\mathcal{G}$  sliced according, e.g., to circle  $C_3$ , is the graph including only links between egos and their alters up to layer 3 (remember that  $C_3$  includes  $C_1$  and  $C_2$ ). More formally, it can be defined as  $\mathcal{G}_{C_3} = (V, E_{C_3})$ , where  $E_{C_3} = \{e_{ij} \in E : i \in V_e, j \in C_3^{(i)}\}$ . Note that the slicing introduces asymmetry in the graph (e.g., a relationship  $i, j$  can be in  $C_4$  for ego  $i$  and in  $C_3$  for ego  $j$ ). However, it is exactly this edge filtering that helps link prediction, as we will show in Section V.

Social-circles slicing can be extended to include also domain-based slicing for domain-specific datasets. For example, from  $\mathcal{G}_{C_3}$  we can retain only domain-specific nodes and edges (i.e., nodes in  $V_e \cup V_d$  and edges in  $E_e \cup E_d$ ). We generalise the notation related to the slicing preprocessing by denoting with  $\omega$  the specific slicing considered, and with  $\mathcal{G}_\omega$  the resulting graph. In Section V, we will discuss the performance of link prediction both with social-based slicing and with social-based plus domain-based slicing. We anticipate here that when the social circles information is not used for the prediction, leveraging the category information provides a significant advantage. Vice versa, social-circles awareness makes information on the category less relevant (and this is extremely important, since the extraction of categories typically requires an additional, domain-specific, classifier or manual labelling).

In order to perform unsupervised link predictions, for each user-user pair  $i, j$  for which a link does not exist in  $E_e$ , we follow Definition 1 below. Note that each slicing will yield different predicted links. Hence, we will treat each slicing choice as a separate link prediction approach.

**Definition 1** (UNSUPERVISED CIRCLES-AWARE LINK PREDICTION). *For a fixed  $K$ , the social circles-aware link prediction algorithm suggests, to a user  $i$ ,  $K$  users  $j$  (with  $j \in V_e \wedge (i, j) \notin E_e$ ) associated with the top- $K$   $\text{sim}_\omega(i, j)$  values, where  $\omega$  denotes the selected slicing and  $\text{sim}_\omega(i, j)$  is the similarity computed on  $\mathcal{G}_\omega$ .*

The definition of a new similarity function is out of the scope of the paper. What we want to do here is to evaluate popular similarity functions available in the literature when they are enriched with knowledge about the ego network circles. In order to isolate the effect of social-circles awareness, we need approaches that are simple (so it is easy to gauge the role of social circles) yet effective. For these reasons, our choice fell on the strategies described below, often used in the related literature. In summarising them, we denote with  $\Gamma_\omega(i)$  the neighborhood of node  $i$  in  $\mathcal{G}_\omega$ .

- *Common neighbors (CN)* [25]: the similarity is given by the number of common neighbors between users  $i$  and  $j$  in  $\mathcal{G}_\omega$ :

$$sim_\omega(i, j) = |\Gamma_\omega(i) \cap \Gamma_\omega(j)|. \quad (1)$$

- *Jaccard's Coefficient (JC)* [25]: the similarity is computed as the Jaccard similarity of the set of common neighbors of users  $i$  and  $j$  in  $\mathcal{G}_\omega$ :

$$sim_\omega(i, j) = \frac{|\Gamma_\omega(i) \cap \Gamma_\omega(j)|}{|\Gamma_\omega(i) \cup \Gamma_\omega(j)|}. \quad (2)$$

- *Adamic-Adar (AA)* [26]: the Adamic-Adar similarity reduces the importance of common neighbours having high degree:

$$sim_\omega(i, j) = \sum_{z \in \Gamma_\omega(i) \cap \Gamma_\omega(j)} \frac{1}{\log(|\Gamma(z)|)}. \quad (3)$$

- *Resource Allocation (RA)* [27]: the RA score is similar to the Adamic-Adar one, but it penalises even more the common neighbors with high degree:

$$sim_\omega(i, j) = \sum_{z \in \Gamma_\omega(i) \cap \Gamma_\omega(j)} \frac{1}{|\Gamma(z)|}. \quad (4)$$

Among these four link prediction algorithms, RA is the one that consistently performs better in the related literature [36], [43], [35]. In Sections V-D and V-E we will investigate if this is still the case when the policies can leverage the knowledge of the social circles.

### B. Supervised learning with social-circle awareness

Each topological metric discussed in the previous section captures a single possible mechanism yielding to the formation of new links in the network. Ref. [44] discusses how this unsupervised approach can take advantage of adding supervised learning on top of it. They show that simple ranking of heuristics is outperformed by supervised classifiers, because the latter have the capability of identifying multiple differentiating boundaries in the similarity score domain, even when just a single heuristic is used as feature.

Thus, in this section, we cast our social-aware link prediction problem into a supervised learning problem. This entails computing a vector of features for each user pair  $i, j$ . Each of the metrics in Section IV-A becomes a feature that describes the user pair. Each pair is also labelled to mark whether the link exists or not. In order to test the different performance of different supervised approaches, in this work we consider the following learning algorithms (in parentheses we report the link prediction papers in which they have been previously used): logistic regression [45], Random Forest [44], decision trees, naive Bayes, and SVM [31], the latter with both linear and polynomial kernels. We use their R implementations (`glm`, `randomForest`, `rpart`, `klaR`, `kernlab`, respectively) together with the `caret` package for training and test. Parameter optimization is applied on the training set with 10-fold cross validation.

### C. Prediction based on feature learning methods

In the literature, there are newly suggested link prediction methods based on learning *latent features* of the graphs, where these feature vectors are low-dimensional vector representations produced by approaches such as graph representation learning [46] and graph neural networks (GNN) [38]. For a detailed description and classification of the methods, the reader may refer to [38], [2], [47], [46] and Section II-B1. In this work, we select two feature learning algorithms to be used as benchmarks for comparison with our social-aware approach. Specifically, we have selected one of the most popular graph embedding methods, *node2vec* [7], and one most the most popular GNN-based link prediction methods, *SEAL* [5]. For details on the two algorithms and their settings, please refer to Appendix B. Note that, differently from Sections IV-A-IV-B, the selected feature learning approaches are not modified to include social-awareness. In fact, the key idea of feature learning is to autonomously learn the important graph features. Thus, our objective, in this case, is to assess whether the automatically learnt features are better than the social-aware hand-engineered ones in predicting new links.

## V. EVALUATION

In this section we carry out the performance evaluation of the link prediction approach proposed in Section IV using the Twitter datasets described in Section III.

### A. Training and test data

For obtaining longitudinal data, we downloaded (December 2019) the Twitter timeline of the gamers in  $E_e$  one year and a half after the initial download, and timeline of the generic users in  $E_e$  eight years after the initial download. We then identified the links between gamers in the gaming-related dataset and between generic users in the generic users dataset that have appeared in the meanwhile. These new 843 links between gamers and new 1216 links between generic users constitute the set  $E_{new}$  of links to be predicted, while  $E_{old}$  contains the links existing in the first temporal snapshot.

While the above considerations are sufficient for unsupervised link prediction, with supervised learning we also need to assign the negatives (i.e., missing links) to the train and test sets. To this aim, we split them 90%-10% between train and test. Clearly, the set of edges selected for test set may offer only a partial view of the performance (i.e., they may be easier or harder than average to predict). For this reason, we perform a  $k$ -fold cross-validation [24], with  $k = 10$ , each time selecting a new 10% of the negative links for the test set. The performance metrics are aggregated via microaveraging [48].

Using the 90% of negatives for training entails training supervised schemes on millions of negatives and only thousands of positives (Table I, Table II). While the impact of class imbalance is typically limited for similarity-based unsupervised approaches like the ones discussed in Section IV-A, it can create serious problems for the scalability and reliability of supervised learning. In order to mitigate the problem, the common approach in the related literature [31], [49], [5] is to undersample the negative class. Despite its widespread use,

this technique is not without drawbacks [50]. In order to provide reliable measurements, we will show both the results obtained with undersampling (Section V-E) and the results obtained with the complete negative class (Appendix E).

### B. Evaluation metrics

Similarly to the related literature [25], [35], [24], unsupervised link prediction algorithms are evaluated using a top- $K$  analysis, i.e., we compare their performance in predicting  $K$  new links. This allows for a fair comparison among the different approaches, as it avoids fixing a similarity threshold for approaches in which the same threshold may have a different meaning. In order to span a reasonable  $K$  range around the number of positives in the test set (which, as we discussed in Section V-A, are 843 and 1216, for the gaming and generic users datasets respectively), we consider  $K \in \{100, 843, 1000\}$  for the gaming-related dataset and  $K \in \{100, 1216, 1500\}$  for the generic users dataset. For a given slice  $\omega$  and for a fixed  $K$ , each unsupervised link predictor described in Section IV-A outputs a list  $L_\omega^K$  of  $K$  pairs in  $V_e \times V_e - E_{old}$ , which are the newly predicted links (each with its associated confidence in prediction, which, in our case, is the similarity value). Then we can compute classic metrics such as:

- $TP = |L_\omega^K \cap E_{new}|$
- $FP = |L_\omega^K - E_{new}|$
- $FN = |E_{new} - L_\omega^K|$
- $TN = |V_e \times V_e - E_{old}| - |L_\omega^K \cup E_{new}|.$

Supervised approaches automatically label all the elements in the test set (without the need to set the  $K$  or a similarity threshold), hence the number of positives predicted is part of the workings of the supervised approach. In this case, thus, we do not pick the top  $K$  potential edges, but we directly rely on the labelling of the trained predictor for the test edges. It is then straightforward to compute the above metrics (for example,  $TPs$  are those edges in  $E_{new}$  that are marked as *new* by the supervised prediction algorithm).

The related literature on evaluating systems with class imbalance suggests using metrics like precision ( $= \frac{TP}{TP+FP}$ ), recall (aka TPR) and  $F_1$  score. As already argued by [49], in the context of link prediction, precision is more important than the other metrics (such as recall), because if the precision is high, one can live with some false negatives. Therefore, the focus of this evaluation will be mostly on precision. For the completeness of results, we also provide, when relevant, the AUC of the precision-recall curve. The results for the  $F_1$  score can be found in Appendix E.

Precision and recall are evaluated based on metrics (TP, FP, TN, FN) which corresponds to specific realizations of a target phenomenon. Thus, the question arises of how confident we can be about the results obtained on a particular collection that is the result of random sampling (as in the case of cross-validation or subsampling of negatives for supervised prediction). To quantify this confidence, we use credible intervals [51]. For further details, please refer to Appendix D.

### C. Experimental setup

We are interested in comparing the performance obtained using the baseline approach (no social circles information, all edges are considered) and the other strategies in which only edges belonging to the ego network circles are considered. The majority of egos in our datasets feature approximately five circles in their ego networks (Appendix C, Figure A.1). For this reason, in our circle-based slicing we will consider circles from C1 to C5, then we group together into the ACTIVE circle all the circles beyond C5 (for those nodes that have more than five circles). Note that, since social layers are concentric and the  $i$ -th also include the  $(i-1)$ -th up to the first one, for egos with less than five circles the predictions in, e.g. C5, are simply based on the last non-empty social layer (which includes all the innermost ones). Recall that the alters in the ACTIVE circle are all those with a *significant* relationship with the ego, i.e., they interact with the ego at a frequency of at least one contact per year [11]. We denote with ALL the situation in which all relationships are considered, which implies considering both the nodes in the ACTIVE circle (significant bonds) and those outside (acquaintances). This corresponds to the baseline from the related literature: all relationships are treated as equal, without factoring in the role of social circles. As anticipated in Section IV, we will consider two scenarios. In the first one, denoted as ALLEDGES, we consider the full graph  $\mathcal{G}$ , slicing it based on the social circles both in the gaming-related dataset and the generic dataset. In the second one (DOMAINSPECIFICEDGES), we perform both social-based and category-based slicing, by only retaining edges between nodes that are domain-specific (i.e., gamers and games, in our case) with the gaming-related dataset.

### D. Evaluation of unsupervised link prediction

We start the evaluation with the unsupervised case. Note that credible intervals are not provided here, because in the unsupervised case there is no sampling of negatives and the links to be predicted are, deterministically, those that have actually appeared between the first and second download.

1) *The ALLEDGES case with the gaming-related dataset:* The precision<sup>1</sup> for varying  $K$  (number of new links recommended) is shown in Figure 2a. The maximum precision across all policies decreases when we increase  $K$ : the more the links recommended, the more the mistakes (as it is generally the case in the related literature). RA clearly outperforms the others in the gaming-related dataset, and this is consistent with previous findings in the related literature [36]. RA seems to benefit significantly from circle-awareness. Specifically, ignoring the outer circle gives a clear advantage to RA, which at least doubles its precision for all  $K$  values. Thus, the mechanism whereby high-degree penalization and social-awareness work hand in hand is very effective from a link prediction standpoint. It is also interesting to note that this advantage seems lost when only the innermost circle is considered.

<sup>1</sup>Note that the generally low precision values are due to the high class imbalance in the dataset. The benchmark random predictor (the always positive one), in the same settings, would achieve an even worse performance, approximately equal to  $2 * 10^{-4}$ .

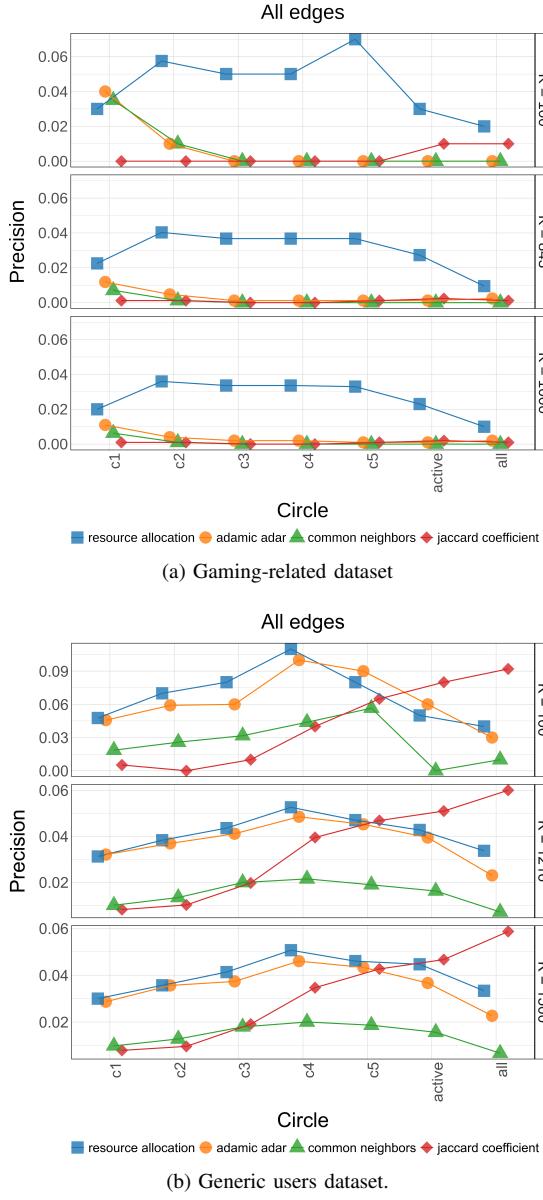


Fig. 2: Precision in unsupervised settings - ALLEDGES.

However, leveraging only the innermost circle C1 means using very little information (only a few links per ego, as can be seen in Figure A.2, Appendix C). This can be an advantage in case of resource limitations (in terms of computational time, computing the similarity is faster on smaller neighborhoods, as we discuss in Appendix F). In such cases, we can have a good prediction (still better than that at ALL) by using only C1. Thus, the predictive power of using only the most intimate relationships unexpectedly outperforms baselines relying on *all* relationships. The drastically different performance of AA with respect to RA in Figure 2a tells us that the way we penalise high-degree nodes impacts on the effect that social circle information has on link prediction effectiveness. However, it is interesting to note that while in general RA is better than AA, this is not the case when resource constraints are considered. When only the most intimate links are kept in the ego networks, AA performs at its best and its precision is not only equivalent to that of RA, but also comparable

TABLE III: ALLEDGES scenario with gaming-related dataset: AUC ( $\times 10$ ) for the precision-recall curve of unsupervised link prediction. We highlight in **bold** the social layer in which each policy performs at its best. We underline the best two AUC overall.

	AUC ( $\times 10$ )						
	C1	C2	C3	C4	C5	ACTIVE	ALL
RA	0.01179	0.03162	0.04699	<b>0.05364</b>	<u>0.05162</u>	0.03641	0.01916
AA	<b>0.02148</b>	0.00749	0.01023	0.01169	<u>0.01270</u>	0.01372	0.01122
CN	<b>0.01914</b>	0.00540	0.00764	0.00902	<u>0.01000</u>	0.01144	0.00924
JC	0.00332	0.00448	0.00648	0.00813	0.00927	<b>0.01007</b>	0.00969

to the precision of RA in the baseline (ALL). In this case, then, strong degree penalization seems to be less important than strong intimacy. The Common Neighbors policy performs quite similarly to AA, even if, in CN, all neighbors contribute the same to the similarity score, regardless of their degree. This is evidence that here AA is working in a regime in which all nodes weight approximately the same (i.e., its logarithmic degree penalization is not enough). Given the similarity with AA, the same considerations we made for AA hold for CN. Finally, we highlight the poor precision of the Jaccard-based strategy, for all  $K$ s and for all circles in the gaming-related dataset. Jaccard similarity gives more weight to neighborhoods that are very similar to each other. This means that not only the overlapping part is considered, but also the number of nodes that are not in common (union of neighbors). As testified by the plots, this restriction does not give an advantage for link prediction in the gaming-related dataset. With respect to social-circles awareness, Jaccard seems to suffer from using only the intimate relationships, while using the ACTIVE layer yields the same precision as the the baseline ALL while saving computational resources.

We report in Table III the Area Under the prediction-recall Curve. Recall that the AUC is not dependent on  $K$ , since it is obtained by exploring the whole range of similarities for making recommendations. When considering the relation between precision and recall through the AUC, we observe that the best overall results (values underlined in the table) are achieved by RA that leverages C4 and C5. Looking at the layers that, for each policy, provide the best AUC, we note that the ALL case (which is the baseline for similarity-based policies from the related literature) is never the best performing. From the circle-awareness standpoint, there seem to exist two classes of policies, those that benefit most from using the outermost circles and those that benefit from the innermost ones.

2) *The ALLEDGES case with the generic users dataset:* Similarly to the gaming-related dataset, we observe that the maximum precision across all policies decreases when we increase  $K$  (Figure 2b). In this case, though, the baseline results (i.e., those without social-awareness) are different: while, for the gamers network, RA was already the best predictor in the ALL case, here we observe a generalised advantage of the JC approach for all  $K$  values. Given this different starting point, let us study how the prediction policies react to circle-awareness. RA preserves the characteristics we

observed for the gamers, and still very much benefits from social-awareness: all the circle-based scenarios have better or comparable precision than the baseline ALL. Note that, as mentioned in the gaming-related dataset results, even if the prediction precision is similar – as it is the case in Figure 2b for C1 and ALL with large  $K$  – using the most intimate links in C1 gives us the advantage of smaller computation time against traditional methods (see Appendix F for a thorough discussion). Turning our attention from RA to AA, Figure 2b shows that AA performs significantly better compared to what we observed for the gaming-related dataset: here, AA substantially mimics the performance of RA, both in terms of achieved precision and in terms of social circles in which it performs best (specifically, around C4). Interestingly, AA was mimicking instead the performance of CN in the gaming-related dataset, with peak performance in C1 and a generally poor precision both in the other social circles and in the baseline ALL. The fact that this difference is due to the different structural properties of the generic users graph with respect to the gamers one follows directly from the definitions of RA, AA, and CN in Section IV-A. Indeed, all the three policies are based on the principle that the more the common neighbors, the better for link prediction. Then, in CN all common neighbors weight the same, while in AA and RA the common neighbors with high degree weight less (much less in RA than in AA, since in RA the penalization is linear while in AA it is logarithmic) than those with small degree. The fact that the performance of AA is approaching that of RA implies that the degree of the common neighbors is not very high (hence the logarithmic penalization of AA and the linear one of RA are in a regime in which they yield similar scores). Taking into account the degree of the common neighbors still provides an advantage, though, as highlighted by the worse performance of CN with respect to AA and RA. As a final remark, note that while JC is performing extremely well with the generic users dataset in the baseline ALL, its performance rapidly deteriorates as we incorporate circle information. This tendency was already present in the gamers datasets (Figure 2a) but was somewhat masked by the generally poor precision achieved by JC regardless of the circle/baseline considered. It is interesting to speculate on why circle-awareness does not help JC in general. To this aim, let us consider the definition of the Jaccard similarity in Section IV-A. JC captures the fraction of common neighbors out of all neighbors for a pair of users  $i, j$ . Thus, its very nature requires “useless” neighbors (i.e. those that are not in common) to be present, in order to differentiate between high and low neighborhood overlap. However, the slicing based on social-circles effectively remove “useless” nodes, hence impairing the discriminating capabilities of the JC score.

In Table IV, we report the Area Under the prediction-recall Curve for the generic uses dataset. Again, recall that the AUC captures the trade-off between precision and recall and that it is not dependent on the specific  $K$  value considered, as it spans the whole  $K$  range. The best overall precision-recall performance is achieve by RA when it leverages social information in C5. The second best AUC is provided by JC in the baseline case (no circle-awareness). AA and CN provides

TABLE IV: ALLEDGES scenario with generic users dataset: AUC ( $\times 10$ ) for the precision-recall curve of unsupervised link prediction. We highlight in **bold** the social layer in which each policy performs at its best. We underline the best two AUC overall.

	AUC ( $\times 10$ )						
	C1	C2	C3	C4	C5	ACTIVE	ALL
RA	0.03028	0.05755	0.10289	0.14233	<b>0.15142</b>	0.13941	0.09313
AA	0.02730	0.04987	0.07930	0.11554	<b>0.12075</b>	0.10057	0.06273
CN	0.01106	0.01914	0.03093	0.04678	<b>0.04714</b>	0.04171	0.03088
JC	0.00927	0.01737	0.03573	0.07824	0.10735	0.12498	<b>0.14390</b>

the highest AUC when leveraging C5 information. Overall, from the results in Table IV, we can thus conclude that, even though the advantage of circle awareness is less evident in the generic users dataset, it still outperforms non-circle-related approaches.

3) *The DOMAINSPECIFIEDGES case:* Recall that, in this case, for any social-based slicing, we also remove all nodes (and associated edges) that are not domain-specific (this is only relevant to the gaming-related dataset). All the nodes left after this additional, category-based slicing are either gamers or games, i.e., they belong to the specific community for which we are making link recommendations. Making a link prediction is now much easier, because the network has been pruned by nodes potentially irrelevant or misleading. Indeed, as shown in Figure 3, the precision of AA and CN significantly improves with respect to the ALLEDGES scenario in Figure 2a. This improvement is also present when no social-circle information is used (ALL circle in Figure 2a). This implies that when social-circle awareness is not used for prediction, leveraging the category information provides a significant advantage. Vice versa, social-circles awareness seem to make information on the category unnecessary. Since the extraction of user categories typically requires a domain-specific classifier or manual labelling, being able to skip this phase without affecting the link prediction quality would be very important. The precision of the Jaccard-based approach remains very low also when considering domain-specific edges only. We conclude this part on the DOMAINSPECIFIEDGES scenario by analysing the AUC of the precision-recall curve, reported in Table V. Table V shows that, also in this case, the best performance is achieved by RA in the outermost circles (specifically, ACTIVE and C5). We observe again the duality of behaviours with respect to the gain from social-circles awareness: RA and AA achieve their best AUC in ACTIVE, CN and JC in C1. The baseline ALL is never optimal (neither from an absolute standpoint nor for specific policies). This further confirms the advantage of using social circle awareness in general.

#### E. Evaluation of supervised link prediction

In this section we assess the advantages brought about by including a supervised classifier in the link prediction approach. As discussed in Section IV-B, our classifier uses as features the similarity-based metrics introduced in Section IV-A. The supervised approaches that we test are logistic regression, decision trees, naïve Bayes, Random Forest, SVM.

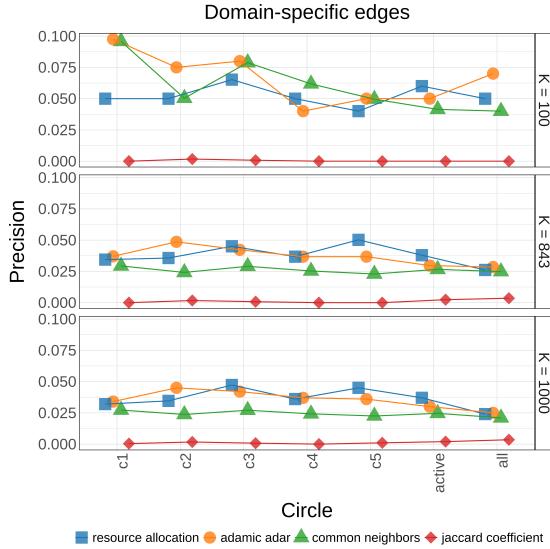


Fig. 3: Precision in unsupervised settings - DOMAINEDGES (only relevant to the gaming-related dataset).

TABLE V: DOMAINEDGES scenario with gaming-related dataset: AUC ( $\times 10$ ) for the precision-recall curve of unsupervised link prediction. We highlight in **bold** the social layer in which each policy performs at its best. We underline the best two AUC overall.

	AUC ( $\times 10$ )						
	C1	C2	C3	C4	C5	ACTIVE	ALL
RA	0.3331	0.5457	0.7275	0.7361	0.8175	<b>0.8807</b>	0.6605
AA	0.6564	0.5754	0.7638	0.6762	0.6834	<b>0.7870</b>	0.5745
CN	<b>0.6523</b>	0.5643	0.5381	0.4289	0.4131	0.5562	0.4028
JC	<b>0.1296</b>	0.1270	0.1353	0.1205	0.1117	0.1053	0.0950

Recall (from Section V-A) that with supervised learning we investigate the performance both on the original dataset and on the undersampled one. Also, with supervised learning we do not carry out a top-K analysis but we allow the algorithms to freely classify all the edges in the test set.

Figures 4a, 4b, and 5 show the results, for the ALLEDGES and the DOMAINEDGES case respectively. Recall that, as explained in Section V-A, these results are obtained with the negatives undersampled to make the positive and negative classes balanced. In Appendix E we also investigate the link prediction performance when no undersampling is carried out (the findings are substantially confirmed). Figure 4a shows the precision of the supervised link prediction approaches in the ALLEDGES scenario of gaming-related dataset. We can identify two important results. First, all approaches that rely on C1 information provide better precision than with any other layer. In particular, using all edges (case ALL) is never better than using C1 alone. Second, all circle-based approaches that leverage information on the outermost social layers perform at least as well as the baseline (ALL). When considering the generic users dataset (Figure 4b), we observe that the peak performance in C1 is confirmed for Random Forest and naïve Bayes. However, for the other learning strategies, the precision drastically drops in C1. For generic users, we can

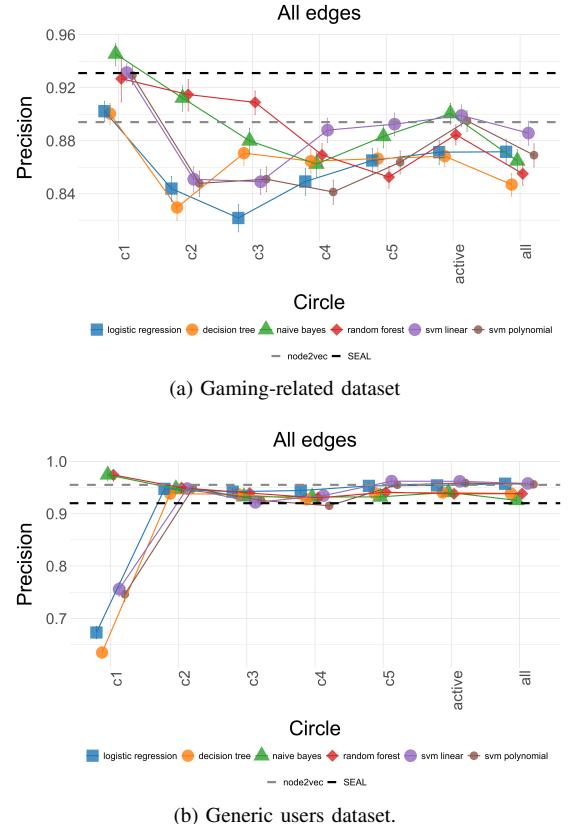


Fig. 4: Precision (with credible intervals) - SUPERVISED ALLEDGES on the undersampled datasets. The ALL scenario of feature learning algorithms is also included (it will be discussed in Section V-F).

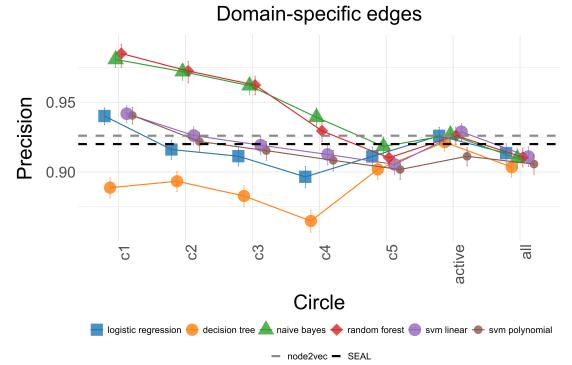


Fig. 5: (Gaming-related dataset) Precision (with credible intervals) - SUPERVISED DOMAINEDGES on the undersampled dataset. The ALL scenario of feature learning algorithms is also included (it will be discussed in Section V-F).

conclude that all the social circles but the first one provide approximately the same precision as the baseline ALL. Finally, please note that, for both datasets, the precision achieved with supervised strategies is much higher than that obtained with unsupervised ones (consistently with the results in the related literature [49]).

In the DOMAINSPECIFIEDGES scenario for the gaming-related dataset (Figure 5), again naïve Bayes and Random

Forest achieve the highest precision when leveraging only C1 relationships. Differently from the previous case, the precision of decision trees in C1 is lower than with other layers. The remaining learning strategies still gain from using C1 information, but the margin is smaller in this case. Differently from the unsupervised case, supervised learning seems able to overcome the difficulties of predicting new links in the ALLEDGES scenario, hence the relative advantage of filtering out links not domain-specific is partially lost.

In summary, supervised learning consistently yields better predictions than their unsupervised counterparts. This may be due to more flexible identification of boundaries between positives and negatives (with respect to the simple threshold-based approach of similarity-based heuristics) or to a smart combination of the similarity-based heuristics. Regardless, social circles again prove very effective in boosting the precision of link prediction strategies, especially in the gaming-related dataset. In particular, supervised strategies seem able to effectively exploit the innermost layer C1 in the domain-dependent dataset and the active network layer in domain-independent dataset much better than the unsupervised cases.

#### F. Comparison against link prediction methods based on feature learning

Until now, we have studied the social-aware link prediction method by embedding the intimacy levels of Dunbar’s ego network model into existing unsupervised and supervised approaches. The ego network model is a way of capturing the local information of relationships of the individuals. Since the aim of the study is to understand the contribution of the social circles to the performance of the link prediction methods, we selected prediction methods where the feature calculation is not a black-box, and the link prediction method is not an end-to-end method. All the methods discussed so far were based on explicit graph features. In this section, we compare the prediction performance of the methods based on latent features (*node2vec* and *SEAL*, discussed in Section IV-C) to feature extraction methods leveraging social circle information.

Table VI shows the precision results of *node2vec* (for the best  $(p, q)$  pair), *SEAL* (whose best precision is achieved with  $h = 1$  without embeddings), and the best supervised social-aware algorithms (whose the best results always occur with C1) on the undersampled<sup>2</sup> generic users and gamers datasets. Recall that we are using *node2vec* and *SEAL* approaches as baseline methods where all edges are included without any slicing (this corresponds to the case ALL in the previous section). Please note that this comparison is fair, since we are studying machine learning-based algorithms trained on explicit features against a machine learning-based algorithm trained on latent features. Vice versa, we do not consider the unsupervised approaches from Section V-D as they do not leverage machine learning techniques. Table VI shows that the best circle-aware link prediction methods always outperform latent feature-based algorithms. Looking at Figures 4a, 4b, and 5 for a circle-by-circle comparison, we observe that either or both

<sup>2</sup>For consistency with the original *node2vec* and *SEAL* papers, where the negatives are undersampled in the evaluation.

TABLE VI: Precision scores of *node2vec* with best  $(p, q)$  combinations, *SEAL* with  $h = 1$  without embeddings (always the best configuration in our scenarios), and the best circle-aware supervised methods (all use C1 only, the learning method is *RF*-Random Forest, *NB*-naïve Bayes). The bold cells represent the best precision score per column/scenario.

	Generic Users	Gamers	
	ALLEDGES	DOMAINEDGES	
<i>node2vec</i>	0.955 ( $p = 1, q = 4$ )	0.894 ( $p = 0.25, q = 0.5$ )	0.926 ( $p = 0.25, q = 4$ )
<i>SEAL</i>	0.920	0.931	0.920
<i>circle-aware link prediction</i>	<b>0.974</b> (C1; RF & NB)	<b>0.945</b> (C1; NB)	<b>0.985</b> (C1; RF)

feature learning strategies always provide the best precision in the ALL case. As expected, feature learning strategies are also generally very competitive in the other cases, outperforming circle-aware approaches for several slicing. However, they are never able to surpass the precision achieved by the best circle-aware strategy.

The implications of the above results are far-reaching: by using knowledge on a few common strong ties, circle-aware link prediction consistently beats black-box approaches. This advantage is not even paid in terms of computational complexity: as we show in Appendix F, all the strategies in Table VI are linear in the number of edges.

## VI. CONCLUSION

In this paper, we have studied the performance of circle-aware feature-extraction link prediction algorithms. Specifically, relying on very well-established models from anthropology, we have considered the social circles in individual ego networks, using the circle as a proxy of intimacy. We have selected four benchmark heuristics and we have modified them to include awareness of the social circles. Our results show that social-circles-based link prediction is generally extremely effective. Specifically, in the vast majority of cases, regardless of the prediction approach (unsupervised or supervised), the specific heuristic or learning algorithm, and the metric (precision, AUC, F1 score) considered, leveraging social circles information outperforms the corresponding baseline in which circles are ignored. In addition, using only information about the innermost social circles guarantees the same performance achieved when using the whole network. Using social circles information also seems to provide the same performance as using additional classifiers on nodes, which might be impractical or costly to set up. Finally, and most importantly, circle-aware supervised link prediction outperformed recent state-of-the-art feature learning-link prediction approaches, including a GNN-based solution. Interestingly, the best-performing circle is C1, which comprises only the two or three strongest relationships of the ego: by using knowledge on a few common strong ties, circle-aware link prediction consistently beats black-box approaches.

## REFERENCES

- [1] Global State of Digital, “The global state of digital in 2021 report,” We are social and Hootsuite, Tech. Rep., 2021.

- [2] E. C. Mutlu, T. Oghaz, A. Rajabi, and I. Garibay, "Review on learning and extracting graph features for link prediction," *Machine Learning and Knowledge Extraction*, vol. 2, no. 4, pp. 672–704, 2020.
- [3] V. Martínez, F. Berzal, and J.-C. Cubero, "A Survey of Link Prediction in Complex Networks," *ACM Computing Surveys*, vol. 49, no. 4, pp. 1–33, dec 2016.
- [4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
- [5] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, vol. 2018-Decem, 2018, pp. 5165–5175.
- [6] L. Cai, J. Li, J. Wang, and S. Ji, "Line graph neural networks for link prediction," *arXiv preprint arXiv:2010.10046*, 2020.
- [7] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.
- [8] S. De Winter, T. Decuyper, S. Mitrović, B. Baesens, and J. De Weerdt, "Combining temporal aspects of dynamic networks with node2vec for a more efficient dynamic link prediction," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 1234–1241.
- [9] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.
- [10] R. A. Hill and R. I. Dunbar, "Social network size in humans," *Human nature*, vol. 14, no. 1, pp. 53–72, 2003.
- [11] W.-X. Zhou, D. Sornette, R. a. Hill, and R. I. M. Dunbar, "Discrete hierarchical organization of social group sizes," *Proceedings. Biological sciences / The Royal Society*, vol. 272, no. 1561, pp. 439–444, 2005.
- [12] R. Dunbar, "The social brain hypothesis," *Evolutionary Anthropology*, vol. 9, no. 10, pp. 178–190, 1998.
- [13] A. G. Sutcliffe, D. Wang, and R. I. Dunbar, "Modelling the role of trust in social relationships," *ACM Transactions on Internet Technology (TOIT)*, vol. 15, no. 4, p. 16, 2015.
- [14] V. Arnaboldi, M. Conti, M. La Gala, A. Passarella, and F. Pezzoni, "Information diffusion in osns: The impact of nodes' sociality," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, ser. SAC '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 616–621.
- [15] S. Aral and M. Van Alstyne, "The diversity-bandwidth trade-off," *American Journal of Sociology*, vol. 117, no. 1, pp. 90–171, 2011.
- [16] M. Everett and S. P. Borgatti, "Ego network betweenness," *Social networks*, vol. 27, no. 1, pp. 31–38, 2005.
- [17] N. Lin, K. S. Cook, and R. S. Burt, *Social capital: Theory and research*. Piscataway, New Jersey, USA: Transaction Publishers, 2001.
- [18] C. McCarty, "Structure in personal networks," *Journal of social structure*, vol. 3, no. 1, p. 20, 2002.
- [19] B. Gonçalves, N. Perra, and A. Vespignani, "Modeling users' activity on twitter networks: Validation of dunbar's number," *PloS one*, vol. 6, no. 8, p. e22656, 2011.
- [20] R. Dunbar, V. Arnaboldi, M. Conti, and A. Passarella, "The structure of online social networks mirrors those in the offline world," *Social Networks*, vol. 43, pp. 39–47, 2015.
- [21] L. M. Aiello and N. Barbieri, "Evolution of ego-networks in social media with link recommendations," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 111–120.
- [22] J. O. Haerter, B. Jamtveit, and J. Mathiesen, "Communication dynamics in finite capacity social networks," *Physical review letters*, vol. 109, no. 16, p. 168701, 2012.
- [23] G. Miritello, E. Moro, R. Lara, R. Martínez-López, J. Belchamber, S. G. Roberts, and R. I. Dunbar, "Time as a limited resource: Communication strategy in mobile phone networks," *Social Networks*, vol. 35, no. 1, pp. 89–95, jan 2013.
- [24] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, mar 2011.
- [25] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *JASIST*, 2007.
- [26] L. A. Adamic and E. Adar, "Friends and neighbors on the Web," *Social Networks*, vol. 25, no. 3, pp. 211–230, jul 2003.
- [27] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *Eur. Phys. J. B*, vol. 71, pp. 623–630, 2009.
- [28] D. Liben-Nowell, "An algorithmic approach to social networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.
- [29] W. Liu and L. Lü, "Link prediction based on local random walk," *EPL (Europhysics Letters)*, vol. 89, no. 5, p. 58007, 2010.
- [30] S. Soundarajan and J. Hopcroft, "Using community information to improve the precision of link prediction methods," in *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web Companion*, 2012, pp. 607–608.
- [31] M. A. Hasan, V. Chaoji, S. Salem, M. Zaki, and N. York, "Link Prediction using Supervised Learning," in *SDM06: workshop on link analysis, counter-terrorism and security*, 2006.
- [32] M. Zhang and Y. Chen, "Weisfeiler-lehman neural machine for link prediction," in *Proceedings of ACM SIGKDD*, vol. Part F1296, 2017, pp. 575–583.
- [33] W. Cukierski, B. Hamner, and B. Yang, "Graph-based features for supervised link prediction," in *Proceedings of the International Joint Conference on Neural Networks*, 2011, pp. 1237–1244.
- [34] Y. Wang and J. Li, "Credible Intervals for Precision and Recall Based on a K -Fold Cross-Validated Beta Distribution," *Neural Computation*, vol. 28, no. 8, pp. 1694–1722, aug 2016.
- [35] B. Zhu and Y. Xia, "Link Prediction in Weighted Networks: A Weighted Mutual Information Model," *PLOS ONE*, vol. 11, no. 2, p. e0148265, feb 2016.
- [36] L. Lü and T. Zhou, "Link prediction in weighted networks: The role of weak ties," *EPL*, vol. 89, no. 1, 2010.
- [37] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.
- [38] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, 2020.
- [39] W. L. Hamilton, *Graph Representation Learning Hamilton*, 2020.
- [40] V. Arnaboldi, M. Conti, A. Passarella, and F. Pezzoni, "Ego networks in twitter: an experimental analysis," in *2013 Proceedings IEEE INFOCOM*. IEEE. New York, NY, USA: IEEE, 2013, pp. 3459–3464.
- [41] C. Boldrini, M. Toprak, M. Conti, and A. Passarella, "Twitter and the press: an ego-centred analysis," in *Companion Proceedings of the The Web Conference '18*, 2018, pp. 1471–1478.
- [42] L. A. Goodman, "Snowball sampling," *The Annals of Mathematical Statistics*, vol. 32, no. 1, pp. 148–170, 1961.
- [43] P. Zhang, X. Wang, F. Wang, A. Zeng, and J. Xiao, "Measuring the robustness of link prediction algorithms under noisy environment," *Scientific Reports*, vol. 6, 2016.
- [44] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in *Proceedings of ACM SIGKDD*, 2010, pp. 243–252.
- [45] J. O'Madadhain, J. Hutchins, and P. Smyth, "Prediction and ranking algorithms for event-based network data," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 23–30, 2005.
- [46] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.
- [47] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, "Link prediction techniques, applications, and performance: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 553, p. 124289, 2020.
- [48] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 1, p. 49, nov 2010.
- [49] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi, "Human mobility, social ties, and link prediction," in *Proceedings of the ACM SIGKDD'11*. New York, New York, USA: ACM Press, 2011, p. 1100.
- [50] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla, "Evaluating link prediction methods," *Knowl Inf Syst*, vol. 45, pp. 751–782, 2015.
- [51] C. Goutte and E. Gaussier, "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation," in *Advances in Information Retrieval. ECIR 2005*. Berlin, Heidelberg: Springer, 2005, pp. 345–359.
- [52] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
- [53] D. Garcia-Gasulla, U. Cortés, E. Ayguadé, and J. Labarta, "Evaluating Link Prediction on Large Graphs," in *Frontiers in Artificial Intelligence and Applications*, 2015.
- [54] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. Cambridge, MA, USA: MIT press, 2009.

- [55] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. New Delhi, India: Pearson, 2019.
- [56] P. Bille, A. Pagh, and R. Pagh, “Fast evaluation of union-intersection expressions,” in *Algorithms and Computation*, T. Tokuyama, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 739–750.
- [57] D. Zhang, J. Yin, X. Zhu, S. Member, C. Zhang, and S. Member, “Network Representation Learning : A Survey,” no. June, pp. 1–25, 2018.

## APPENDIX A DATA COLLECTION

In order to collect an initial community of users interested in indie games, we first had to identify some relevant indie games. To this purpose, we referred to Steam, a digital game distribution platform (<https://store.steampowered.com/>), and SteamSpy (<https://steamspy.com/>) , which provides statistics about the games on Steam. Among the indie games listed as the most popular during October 2017, we were able to identify the Twitter accounts of 133 of them. We then downloaded the timelines of these games using the Twitter Search API. From the downloaded timelines, the 400 most frequently used hashtags have been extracted and used to monitor Twitter using the Twitter Streaming API. This allowed us to identify a set of 8,932 users engaging in game-related conversations. We labelled these users as *gamers*. We have then collected (June 2018) their timeline (most recent 3200 tweets) using the Twitter Search API. In order to enrich the set of game-related nodes in the dataset, we also downloaded all followers of the initial 133 games, thus bringing in additional 25,014 nodes, for a total of 31,091 domain-specific nodes. Note that we do not collect the ego networks of these nodes.

## APPENDIX B

### FEAUTURE LEARNING METHODS USED FOR BENCHMARKS

*node2vec* is a framework that maps the nodes of a graph to low-dimensional feature vectors while preserving nodes’ neighborhoods [7]. It proposes a biased random walk based on breadth-first sampling (BFS) and depth-first sampling (DFS) with transition probabilities  $p$  and  $q$ . BFS captures the embeddings related to structural equivalence (e.g., being a hub node) while DFS captures embeddings related to communities based on homophily. There exist other four parameters for *node2vec*:  $d$  the dimension/length of the feature vector,  $l$  the length of random walk,  $k$  the size of the neighborhood, and  $r$  the number of walks per node. A feature vector is produced per node in the network by combining the Skip-gram architecture of *word2vec* with a flexible neighborhood definition obtained with biased random walks. For the link prediction task, these node embeddings are transformed into edge embeddings by applying binary operators like, e.g., Hadamard. Then, logistic regression is applied on the edge embeddings to train a binary classification model on positive and negative data (existing and missing links). Given that *node2vec* is designed to capture latent features, in this study, we apply *node2vec* method as is, without explicitly adding information on the social circles. Since the aim is to predict the possible future links between the egos whose social circles can be computed, as described in Section IV, we learn the node embeddings of the vertex set  $V_e$  by using all edges  $E_e$ . For the extraction

of node embeddings, we try the different combinations of  $p$  and  $q$  values ( $p, q \in \{0.25, 0.5, 1, 2, 4\}$ ) while keeping other parameters of random walk fixed ( $d = 128$ ,  $l = 80$ ,  $k = 10$ ,  $r = 10$ ) as mentioned in [7]. Then, we calculate the edge embeddings of all existing and missing links in  $V_e \times V_e$ . Given two feature vectors,  $f(i)$  and  $f(j)$ , of egos  $i$  and  $j$  where  $i, j \in V_e$ , the edge embedding of the edge  $e_{ij}$  is calculated as:  $f(i) \circ f(j)$  where  $\circ$  is a binary operator (average, Hadamard, weighted-L1, weighted-L2). We are using Hadamard operator, which is shown to be the most effective one in [7].

*SEAL* [5] is a graph neural network method that focuses on the local enclosing subgraphs of the target links (node pairs). For a node pair  $i, j$ , the local enclosing subgraph is the subgraph induced by the union of  $i$ ’s and  $j$ ’s neighborhoods, up to  $h$  hops. Once the enclosing subgraphs are extracted, the nodes in them are labelled using the Double-Radius Node Labeling (DRNL), whereby the labels are a function of the distance from the target nodes  $i, j$ . Parameter  $h$  defines the order of the neighborhood. Zhang and Chen [5] show that higher-order graph structures and their corresponding heuristics (such as Page Rank, Katz index) can be captured using small  $h$  values. Here, we use all relationships ( $E$ ) to extract the local enclosing subgraph of all ego node pairs on which we apply the prediction task. We also set  $h \in \{1, 2\}$ , like in [5]. In *SEAL*, the extracted local enclosing subgraphs, for the existing and missing links, are then fed to a Deep Graph Convolutional Neural Network (DGCNN) [52] to train the model. For the DGCNN, we use the same parameters of the *SEAL* paper. The core structure of *SEAL* consists of local enclosing subgraphs and DGCNN, which can be extended with node features (which we don’t have) and node embeddings (such as those of *node2vec*). In [5], it was shown that node embeddings may or may not increase the prediction performance. Thus, here we carry out experiments both with and without embeddings.

## APPENDIX C PRELIMINARY EGO NETWORK ANALYSIS

As anticipated in Section III, we extract the ego network structure of the gamers and generic users in  $V_e$  using the methodology described in [41]. The resulting distribution of the optimal circle number of gamer egos and generic user egos can be seen in Figure A.1. The mean value is 4.94 for gamers and 4.73 for generic users, and the median and mode values are equal to 5 for both datasets. The plot shows that the number of social circles of both types of egos is compatible with the findings of previous studies which establish that human ego network can be layered into 5 layers/circles. For further understanding of the ego network structures, we examine the alter distribution through the layers. In previous studies [20], it has been shown that in OSN the external layers are slightly smaller than the reference model discussed in Section II-A. In Figure A.2, we show the distribution of the alter number for the egos with optimal circle number equal to 5 (the case generally studied in the related literature). These results, both for gamers and generic users, are compatible with the typical findings from OSNs analysis in the related literature.

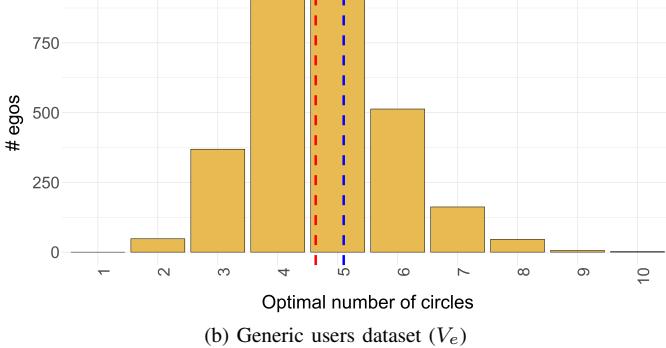
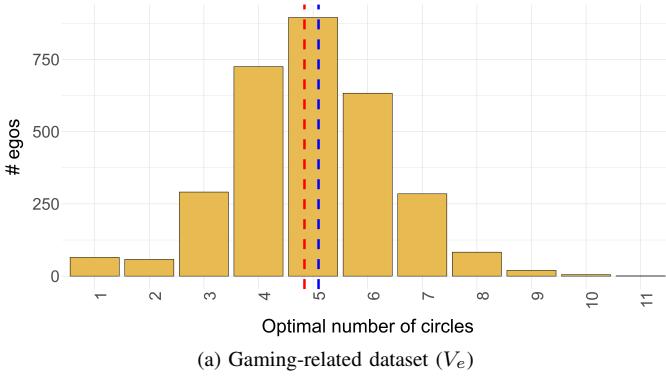


Fig. A.1: Distribution of the optimal number of social circles for the gamer egos and generic user egos. Red and blue lines represent the mean and median values, respectively.

#### APPENDIX D EVALUATION SETUP

Traditionally, binary classifiers are evaluated using the ROC curve, which plots the True Positive Rate ( $\frac{TP}{TP+FN}$ ) against the False Positive Rate ( $\frac{FP}{FP+TN}$ ). However, the scenario we are considering clearly suffers from the class imbalance problem, since the actual edges between nodes are significantly fewer than all the possible edges that there could exist between them. In this situation, the ROC curve, while being insensitive to the class imbalance<sup>3</sup>, tends to provide overoptimistic results [53]. The related literature on evaluating systems with class imbalance suggests using metrics like precision ( $= \frac{TP}{TP+FP}$ ), recall (aka TPR) and  $F_1$  score. As already argued by [49], in the context of link prediction, precision is more important than the other metrics (such as recall), because if the precision is high, one can live with some false negatives. Therefore, the focus of this evaluation will be mostly on precision. For the completeness of results, we also provide, when relevant, the  $F_1$  score and the AUC of the precision-recall curve. Note also that we are interested in comparing the performance obtained using the baseline approach (no social circles information, all edges are considered) and the other strategies in which only edges belonging to the ego network circles are considered.

<sup>3</sup>Differently from the accuracy, which should not be used when imbalance is an issue. In fact, given the accuracy formula  $\frac{TP+TN}{P+N}$ , it is easy to see that correct classification in the most numerous class will hide anything about the sparse class.

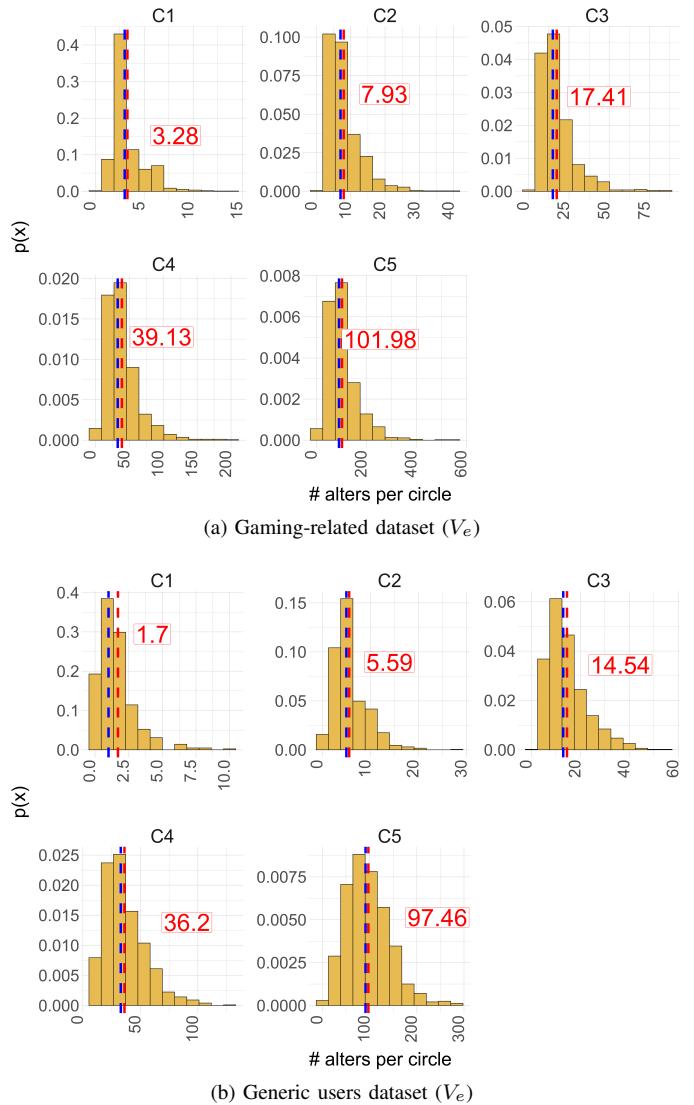


Fig. A.2: Average alter distribution per each circle of gamer egos and generic user egos. For the sake of clarity, egos that have optimal circle number equal to five have been used. Red and blue lines represent the mean and median values, respectively, of the distribution.

Precision and recall are evaluated based on metrics (TP, FP, TN, FN) which corresponds to specific realizations of a target phenomenon. Thus, the question arises of how confident we can be about the results obtained on a particular collection that is the result of random sampling (as in the case of cross-validation or subsampling of negatives for supervised prediction). [51] proposes a Bayesian approach to the problem. The basic idea is to model TP, FP, FN, TN as if they were sampled from a multinomial distribution, with unknown success probability per category  $\pi_i$ . Based on the properties of the multinomial distribution, we know that individual counts are binomial (i.e., equivalent to the number of successes in a sequence of  $n$  independent experiments, each with success probability  $p$ ). Specifically, it can be proved that TP are binomial with parameters  $TP+FP$  (corresponding to the number of trials) and success probability  $p$  (corresponding to

the precision). Using Bayes' theorem and well-known priors associated with binomials, it can be derived that the posterior of the precision  $p$  is Beta distributed:

$$p|\mathcal{D} \text{ (aka the posterior distr.)} \sim \text{Beta}(TP + \lambda, FP + \lambda). \quad (5)$$

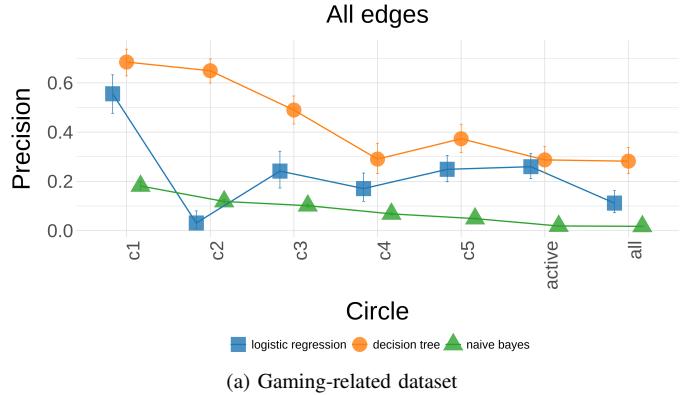
A similar formula is obtained for the recall  $r$ . In these formulas,  $\lambda$  is a parameter whose value depends on the prior considered:  $\lambda = \frac{1}{2}$  corresponds to Jeffrey's non-informative prior (the one we use here),  $\lambda = 1$  to the uniform prior. Credible intervals<sup>4</sup> for  $p$  and  $r$  can then be obtained as the intervals containing 95% of the Beta distribution. The formula for the credible interval of the F1 score is slightly more complicated, so we do not report it here. Microaveraging and credible intervals are also used in [34].

#### APPENDIX E EVALUATION OF SUPERVISED LINK PREDICTION - ADDITIONAL RESULTS

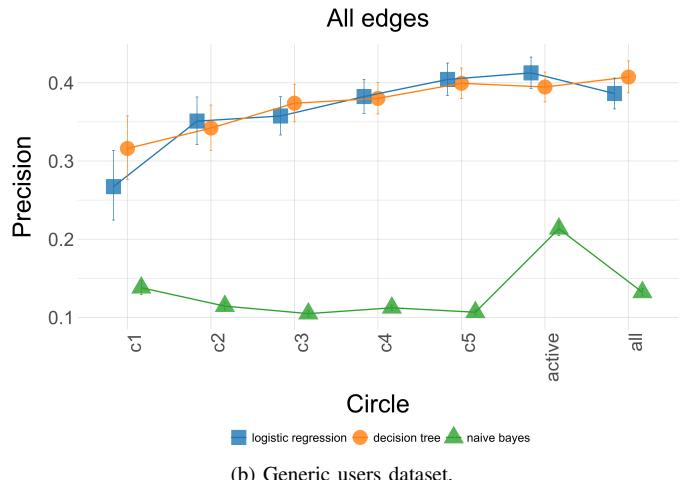
As already mentioned, the precision achieved with supervised learning is generally much higher than that observed with unsupervised algorithms. In order to investigate whether this might be due to an artificial bias introduced by the undersampling of negatives, we have also run the same link prediction experiment using the complete set of negatives (again splitting 90%-10% between train and test). Figures A.3a, A.3b, and A.4 show the precision obtained in this case. Only logistic regression, decision trees, and naïve Bayes are shown, since the others did not converge in a reasonable time due to the new size of the problem. The precision values achieved in this case are significantly smaller than those obtained with the undersampled dataset (thus confirming the limitations of this approach). Still, they improve over the ones obtained with the standalone similarity measures. Again, it is confirmed that the C1 layer is the most useful for making precise predictions in the gaming-related dataset (Figures A.3a). This result is very encouraging, since C1 is the most lightweight in terms of resources consumed. In the generic users dataset (Figures A.3b), the role played by the social circles is reversed: using C5 or ACTIVE, we are able to perform predictions that are at least as good as the baseline, however the precision drops as we move towards the innermost layers.

We conclude the section analysing the F1 score, which is the harmonic mean of precision and recall. We only show the results obtained using the full negative set, as they provide a more reliable evaluation. For both the ALLEDGES and the DOMAINEDGES case (Figures A.5a, A.5b, and A.6), we observe a different ranking with respect to the corresponding precision results, with the highest F1 being achieved by the naïve Bayes approach. Still, since, as discussed in [34], precision is much more important than recall in link prediction, we argue that decision trees should be preferred to naïve Bayes. If we only look at the role played by the social circles, we observe that leveraging social circles is always better or as

<sup>4</sup>Using confidence intervals instead of credible intervals would not be the best choice in this case, because precision, recall, and F1 score belong to  $[0, 1]$ , hence, especially for values close to either 0 or 1, they depart significantly from the normality assumption behind confidence intervals.



(a) Gaming-related dataset



(b) Generic users dataset.

Fig. A.3: Precision (with credible intervals) - SUPERVISED ALLEDGES on the full datasets.

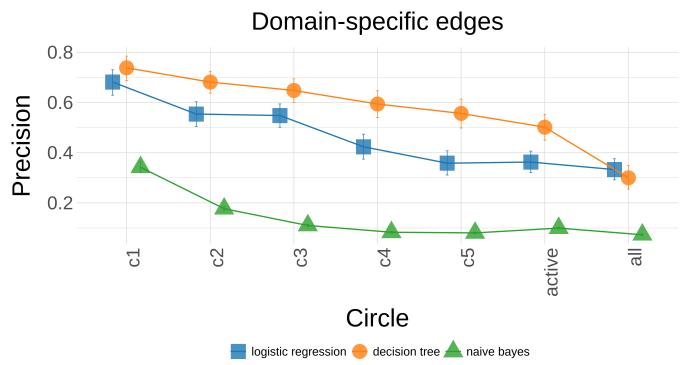
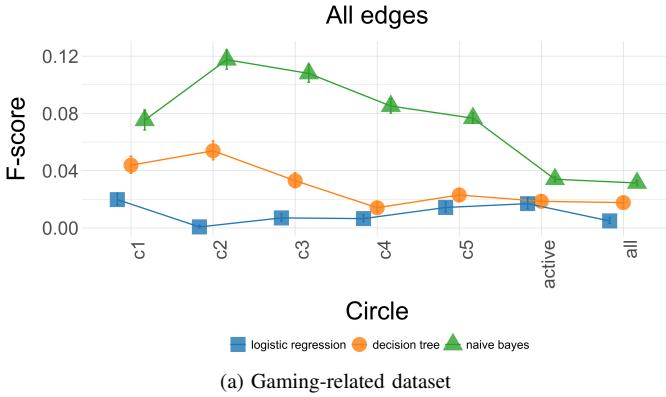
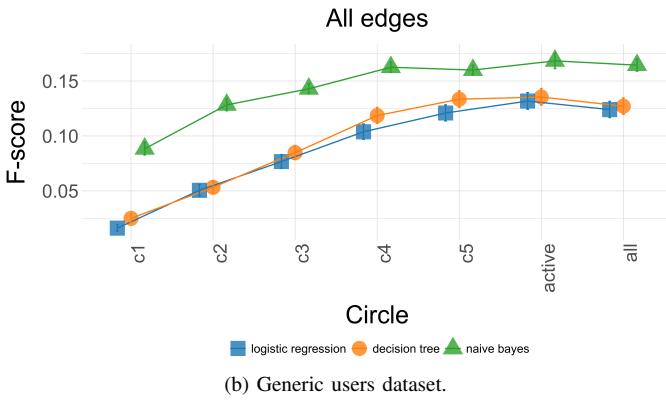


Fig. A.4: Precision (with credible intervals) - SUPERVISED DOMAINEDGES on the full gaming-related dataset.

good as the ALL baselines. However, when aiming at striking a balance between precision and recall, C1 is replaced by C2 as the best performing circle in the gaming-related dataset, probably due to C1's high selectivity for the most intimate relationships. In the generic users dataset, instead, the ACTIVE layer is confirmed as the one providing the best prediction performance.



(a) Gaming-related dataset



(b) Generic users dataset.

Fig. A.5: F1 score (with credible intervals) - SUPERVISED ALLEDGES on the full datasets.

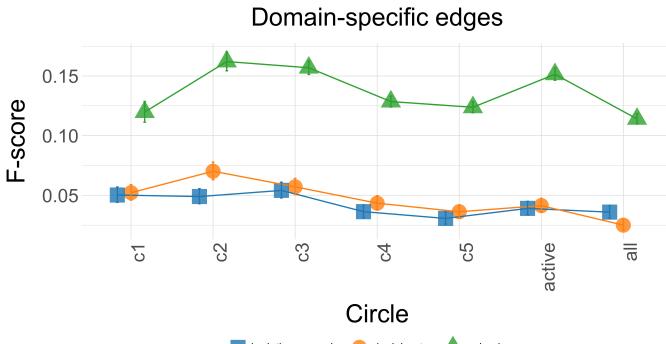


Fig. A.6: F1 score (with credible intervals) - SUPERVISED DOMAINEDGES on the full gaming-related dataset.

## APPENDIX F COMPLEXITY ANALYSIS

In this section, we investigate the computational complexity of ego-aware link prediction and how it compares to the other approaches tested in Section V. For the convenience of the reader, the notation used in this section is summarised in Table A.1.

Let us start with heuristic-based link prediction in the unsupervised case (discussed in Section IV-A). Algorithm 1 illustrates the steps towards computing the similarity between a pair of nodes depending on the specific similarity considered. Please note that the weighted graph discussed in Section III is assumed as input. With respect to the baselines (traditional

TABLE A.1: Notation summary for complexity analysis

Symbol	Description
$V_e$	set of ego nodes
$E_e$	set of existing edges between ego nodes
$\bar{E}_e$	Set of non-existing edges between ego nodes
$E_{train}$	Set of training edges, it is the union set of positive and negative edges used for training
$\Gamma(i)$	Neighborhood of a generic node $i$
$\Gamma_{C_x}(i)$	Social circle $C_x$ of a generic node $i$
$d$	Number of features used for learning
$h$	Depth of the enclosing subgraph computed by SEAL

**Algorithm 1:** Find the similarity between nodes - unsupervised setting

**Data:**  $\mathcal{G} = (V_e, E)$ ;  $C_x$  = the social circle to be considered.

**Notation:**  $\Gamma(i)$  = neighborhood of node  $i$ ;  $\bar{E}_e$  = set of missing edges.

**Result:** Similarity  $sim_{C_x}(i, j)$  for each node pair  $i, j$  in  $V_e$ .

```

1 begin
2   for  $i \in V_e$  do
3     // Compute the ego network and
      slice the graph according to
       $C_x$ 
4     AssignToCircles( $\Gamma(i)$ )
5      $\Gamma_{C_x}(i) = Slice(\Gamma(i), C_x)$ 
6   for  $(i, j) \in \bar{E}_e$  do
7     // Compute the similarity
8     if CN then
9        $sim_{C_x}(i, j) = |\Gamma_{C_x}(i) \cap \Gamma_{C_x}(j)|$ 
10    if JC then
11       $sim_{C_x}(i, j) = \frac{|\Gamma_{C_x}(i) \cap \Gamma_{C_x}(j)|}{|\Gamma_{C_x}(i) \cup \Gamma_{C_x}(j)|}$ 
12    if AA or RA then
13       $sim_{C_x}(i, j) = 0$ 
        for  $z \in \Gamma_{C_x}(i) \cap \Gamma_{C_x}(j)$  do
           $sim_{C_x}(i, j) =$ 
             $sim_{C_x}(i, j) + penalised(\Gamma(z))$ 

```

methods which are represented as ALL in the results) in which no circle information is used, our approach requires two additional steps: (i) computing the ego networks of the nodes for which predictions should be made (line 3) and (ii) pruning the ego networks based on the social circle we want to leverage (line 4). The code from line 5 on is common to both the baselines and the circle-aware approaches, since the specific slicing considered is controlled with  $C_x$  (and  $x = \text{ALL}$  for the baseline). In Lemma 1 below we derive the time complexity for computing the ego networks. Note that, in social networks where nodes are real people, the neighborhood size does not grow significantly with  $|V|$ , hence the complexity of computing an ego network can be considered approximately constant.

**Lemma 1** (Complexity egonets). *The complexity of computing the ego network of a generic node  $i$  is given by the following:*

$$\mathcal{O}(\text{ego}) = \mathcal{O}(|\Gamma(i)| * \log(|\Gamma(i)|)). \quad (6)$$

*Proof.* Line 3 in Algorithm 1 entails a) sorting the neighbors of node  $i$  ( $\Gamma(i)$ ) based on contact frequencies, b) extracting the active part (corresponding to relationship with contact frequency larger than once per year, as discussed in Section II-A), and c) running a unidimensional clustering algorithm on the remaining contact frequencies. The complexity of step (a) is  $\mathcal{O}(|\Gamma(i)| * \log(|\Gamma(i)|))$  (e.g., assuming standard merge sort is used [54]), the complexity of step (ii) is  $\mathcal{O}(\log(|\Gamma(i)|))$  which is equivalent to the complexity of a search algorithm on a sorted input (e.g., assuming binary search is used [54]). After step (b), for each ego  $i$ , we are not working anymore on its full neighborhood but only on the active part  $\Gamma_{active}(i)$ , which is smaller than  $\Gamma(i)$  since it only includes strong relationships. The complexity of step (c) is mainly dependent on the chosen clustering algorithm (note that the classification of alters in circles is robust against the different clustering methods [20]). For example, DBSCAN is quite efficient and it runs in  $\mathcal{O}(|\Gamma_{active}(i)| \log(|\Gamma_{active}(i)|))$  [55]. The time complexity of k-means is linear [55] in the number of points to cluster (i.e.,  $\mathcal{O}(|\Gamma_{active}(i)|)$ ) but it needs additional computations for the selection of the best  $k$  (e.g.,  $\mathcal{O}(k|\Gamma_{active}(i)|)$  for the partition coefficient method,  $\mathcal{O}(|\Gamma_{active}(i)|^2)$  for the Silhouette method). Mean Shift runs in  $\mathcal{O}(|\Gamma_{active}(i)| \log(|\Gamma_{active}(i)|))$ , like DBSCAN. The last step to carry out is the slicing in line 4, whose complexity is equivalent to that of a search in  $\Gamma_{active}(i)$ , hence it is logarithmic in its size. In summary, the complexity of computing an ego network is given by the following (assuming DBSCAN or Mean Shift are used for clustering):

$$\begin{aligned}\mathcal{O}(\text{ego}) &= \mathcal{O}(|\Gamma(i)| * \log(|\Gamma(i)|)) + \\ &\quad + \mathcal{O}(\log(|\Gamma(i)|) + |\Gamma_{active}(i)| \log(|\Gamma_{active}(i)|) + \\ &\quad + \log(|\Gamma_{active}(i)|)) \\ &= \mathcal{O}(|\Gamma(i)| * \log(|\Gamma(i)|)),\end{aligned}$$

where we have leveraged  $|\Gamma(i)| \geq |\Gamma_{active}(i)|$ .  $\square$

Exploiting the above result, we can now derive the time complexity of circle-aware and baseline heuristics with unsupervised link prediction. Theorem 1 below states that in both cases the complexity of unsupervised link prediction grows with  $|V|^2$ , hence it is intrinsically inefficient. Since  $|\Gamma_{C_x}(i)| \ll |\Gamma(i)|$  when  $x$  is small, circle-aware link prediction provides a marginal advantage in this case.

**Theorem 1** (Complexity unsupervised). *The time complexity of link prediction using circle-aware or baseline heuristic  $\phi$  with unsupervised learning is given by the following:*

$$\mathcal{O}(\text{circle-aware } \phi) = \mathcal{O}(|V|^2 * |\Gamma_{C_x}(i)|) \quad (7)$$

$$\mathcal{O}(\text{baseline } \phi) = \mathcal{O}(|V|^2 * |\Gamma(i)|), \quad (8)$$

where  $\phi$  is any of the heuristics defined in Section IV-A (i.e., CN, JC, AA, RA),  $\Gamma(i)$  denotes the neighborhood of node  $i$ , and  $\Gamma_{C_x}(i)$  corresponds to the neighborhood of node  $i$  cut at social circle  $C_x$ .

*Proof.* With reference to Algorithm 1, in order to estimate the time complexity with circle-aware  $\phi$  in the average case, we have to derive the time complexity of computing the ego

network ( $\mathcal{O}(\text{ego})$ , derived in Lemma 1) and that of computing the similarity ( $\mathcal{O}(\text{circle-aware similarity})$ ), as illustrated in Equation 9 below. On the other hand, only the similarity complexity is needed for the baseline case (Equation 10). Note that, in both equations, we have approximated  $|\bar{E}_e|$  (the number of missing edges between egos) as  $|V|^2$ , since social networks are typically sparse.

$$\begin{aligned}\mathcal{O}(\text{circle-aware } \phi) &= \mathcal{O}(|V|)\mathcal{O}(\text{ego}) + \\ &\quad + \mathcal{O}(|V|^2) * \mathcal{O}(\text{circle-aware sim}) \quad (9)\end{aligned}$$

$$\mathcal{O}(\text{baseline } \phi) = \mathcal{O}(|V|^2) * \mathcal{O}(\text{baseline similarity}) \quad (10)$$

Clearly, the complexity of the circle-aware and baseline similarities depends on the specific approach  $\phi$  (CN, JC, AA, RA) considered. We focus first on CN. The complexity  $\mathcal{O}(\text{similarity CN})$  simply corresponds to an intersection between sets (see line 7 of Algorithm 1), hence it is given by Equation 11 for the social-aware case, and by Equation 12 for the baseline. The complexity of a set intersection by using hashing-based dictionaries is  $\mathcal{O}(\min(|Set_1|, |Set_2|))$  [56].

$$\begin{aligned}\mathcal{O}(\text{circle-aware similarity}) &= \mathcal{O}(\min(|\Gamma_{C_x}(i)|, |\Gamma_{C_x}(j)|)) \\ &\approx \mathcal{O}(|\Gamma_{C_x}(i)|) \quad (11)\end{aligned}$$

$$\begin{aligned}\mathcal{O}(\text{baseline similarity}) &= \mathcal{O}(\min(|\Gamma(i)|, |\Gamma(j)|)) \\ &\approx \mathcal{O}(|\Gamma(i)|) \quad (12)\end{aligned}$$

While the above formulas have been derived for CN, it is easy to see that also JC, AA, and RA share the same complexity. Indeed, JC leverages the set union in addition to the set intersection, operation that is still linear in the size of the smaller set. AA and RA are the analogous of CN but with different weights, hence, again, they are linear in the size of the smaller set.

We can now substitute Lemma 1 and Equations 11-12 into Equations 9 and 10 above, thus obtaining:

$$\begin{aligned}\mathcal{O}(\text{circle-aware } \phi) &= \mathcal{O}(|V| * |\Gamma(i)| * \log(|\Gamma(i)|)) + \\ &\quad + \mathcal{O}(|V|^2) * |\Gamma_{C_x}(i)| \\ &= \mathcal{O}(|V|^2) * |\Gamma_{C_x}(i)| \\ \mathcal{O}(\text{baseline } \phi) &= \mathcal{O}(|V|^2) * |\Gamma(i)|.\end{aligned}$$

The right-hand side of the above equation stems from the fact that social networks are sparse, hence  $|\Gamma(i)| \ll |V|$ . Since the number of neighbors in any circle of the ego network is (significantly) smaller than all neighbors in the baseline (because, by definition, social circles only retain the strongest relationships), we have that the time complexity of the ego-aware system is always (significantly) smaller than the baseline. This concludes the proof.  $\square$

In Theorem 2 below, we derive the time complexity for baseline and circle-aware heuristics in the supervised case (discussed in Section IV-B). If negatives are undersampled, supervised learning can provide a much greater efficiency than unsupervised link prediction.

**Theorem 2** (Complexity supervised). *The time complexity of link prediction using circle-aware or baseline heuristics with supervised learning is given by Table A.2. It holds that:*

TABLE A.2: Time complexity of supervised learning algorithms used in Section IV-B.  $d$  denotes the number of features considered (in our case  $d = 4$ , i.e., to the number of heuristics defined in Section IV-A),  $k$  is the number of decision trees used in Random Forest (in our case,  $k = 500$ ).

	Strategy	Undersampled negatives	Full dataset
Training	Logistic regr. Naïve Bayes	$\mathcal{O}( E_e d)$	$\mathcal{O}( V ^2d)$
	Decision Trees	$\mathcal{O}( E_e \log( E_e )d)$	$\mathcal{O}( V ^2\log( V ^2)d)$
	Random Forest	$\mathcal{O}( E_e \log( E_e )dk)$	$\mathcal{O}( V ^2\log( V ^2)dk)$
	SVM	$\mathcal{O}( E_e ^2)$	$\mathcal{O}( V ^4)$
Feature extract.	Baseline	$\mathcal{O}( E_e ) \cdot \mathcal{O}( \Gamma(i) )$	$\mathcal{O}( V ^2) \cdot \mathcal{O}( \Gamma(i) )$
	Circle-aware	$\mathcal{O}( E_e ) \cdot \mathcal{O}( \Gamma_{C_x}(i) )$	$\mathcal{O}( V ^2) \cdot \mathcal{O}( \Gamma_{C_x}(i) )$
Egonet comp.	Baseline	-	$\mathcal{O}( V ) \cdot \mathcal{O}( \Gamma(i) ) \cdot \log( \Gamma(i) )$
	Circle-aware	-	

- When negatives are undersampled, with the exception of logistic regression and naïve Bayes, the training phase ( $\geq \mathcal{O}(|E_e|\log|E_e|)$ ) dominates the complexity, and the effect of ego network computation in the feature extraction phase is negligible.
- When logistic regression and naïve Bayes are used together with negatives undersampling, the overall time complexity is  $\mathcal{O}(|E_e|)$ , with a slightly smaller multiplicative factor for baseline heuristics.
- When negatives are not undersampled, the computation of ego networks (which is  $\mathcal{O}(|V|) \approx \mathcal{O}(|E_e|)$ ) is negligible, and the feature extraction is slightly more convenient for circle-aware heuristics.

*Proof.* The complexity of supervised learning can be decomposed into the complexity for extracting the relevant features and the complexity of solving the learning problem on these features:

$$\begin{aligned}\mathcal{O}(\text{supervised link prediction}) &= \\ \mathcal{O}(\text{feature extraction}) + \mathcal{O}(\text{supervised learning})\end{aligned}$$

It is easy to see that the latter component is the same regardless of the social circle considered, since the features described in Section IV-A, once computed, are simply the four heuristics associated with each edge  $(i, j)$ . Specifically, denoting the number of features with  $d$  (with  $d = 4$  in our case) and the number of training links with  $|E_{train}|$ , the time complexity for training the supervised algorithms used in Section IV-B is:  $\mathcal{O}(d|E_{train}|)$  for logistic regression and naïve Bayes,  $\mathcal{O}(|E_{train}|\log(|E_{train}|)d)$  for decision trees,  $\mathcal{O}(|E_{train}|\log(|E_{train}|)dk)$  for Random Forest ( $k$  is the number of decision trees used in Random Forest),  $\mathcal{O}(|E_{train}|^2)$  for SVM (of the latter, we here consider the kernel version). The number  $|E_{train}|$  of training links depends on whether we consider the undersampled or full training set. In the former case,  $|E_{train}| = 2|E_e|$ , since the negative training links are undersampled to match the number of positive ones. In the latter,  $|E_{train}| \sim \mathcal{O}(|V|^2)$ , since social networks are sparse. Note that the time complexity of the learning phase only depends on the selected supervised learning algorithm.

We now focus on  $\mathcal{O}(\text{feature extraction})$ . We start with the circle-aware feature extraction. In the equation below, we derive the time complexity separating the negatives undersampling and the full dataset case, and leveraging Lemma 1

for  $\mathcal{O}(\text{ego})$  and Equation 11 for  $\mathcal{O}(\text{circle-aware sim})$ . In general, the similarity is computed for all node pairs in  $E_{train} = E_e \cup \bar{E}_e^{train}$ , where  $\bar{E}_e^{train}$  denotes the set of negative training edges. However, with negatives undersampling,  $|\bar{E}_e^{train}| = |E_e|$ , with the full dataset  $\bar{E}_e^{train} \approx \mathcal{O}(|V|^2)$ .

$$\begin{aligned}\mathcal{O}(\text{circle-aware feat. extr.}) &= \\ &= \mathcal{O}(|V|) \cdot \mathcal{O}(\text{ego}) + \begin{cases} \mathcal{O}(|E_e|) \cdot \mathcal{O}(\text{circle-aware sim}) \\ \mathcal{O}(|V|^2) \cdot \mathcal{O}(\text{circle-aware sim}) \end{cases} \\ &\approx \begin{cases} \mathcal{O}(|E_e|) \cdot \mathcal{O}(|\Gamma(i)| \cdot \log(|\Gamma(i)|) \cdot |\Gamma_{C_x}(i)|) \\ \mathcal{O}(|V|^2) \cdot \mathcal{O}(|\Gamma_{C_x}(i)|) \end{cases} \quad (13)\end{aligned}$$

Deriving the complexity of baseline feature extraction is now straightforward. In fact, we can simply neglect the computation of ego networks in Equation 13 and use the formula for  $\mathcal{O}(\text{baseline sim})$  in Equation 12. Then, the equation below follows:

$$\begin{aligned}\mathcal{O}(\text{baseline feature extraction}) &= \\ &= \begin{cases} \mathcal{O}(|E_e|) \cdot \mathcal{O}(\text{baseline sim}) \\ \mathcal{O}(|V|^2) \cdot \mathcal{O}(\text{baseline sim}) \end{cases} = \\ &= \begin{cases} \mathcal{O}(|E_e|) \cdot \mathcal{O}(|\Gamma(i)|) \\ \mathcal{O}(|V|^2) \cdot \mathcal{O}(|\Gamma(i)|) \end{cases} \quad (14)\end{aligned}$$

Comparing Equations 13 and 14, we remark that, when using the full dataset, the time complexity of feature extraction is dominated by  $\mathcal{O}(|V|^2)$ , with circle-aware features being slightly more convenient (since  $|\Gamma_{C_x}(i)| < |\Gamma(i)|$ ). Vice versa, with negatives undersampling, the time complexity of feature extraction is  $\mathcal{O}(|E_e|)$  and baseline features are marginally more advantageous.  $\square$

If we compare the heuristic-based link prediction complexity in Theorems 1-2, we observe that supervised link prediction with negatives undersampled is more efficient than unsupervised link prediction, as it allows to reduce the complexity from  $\mathcal{O}(|V|^2)$  to  $\mathcal{O}(|E_e|)$ . In these settings, the small overhead of computing ego networks is paid off by the outstanding prediction performance. We can now derive the time complexity of node2vec-based link prediction and SEAL.

**Theorem 3.** *The time complexity of link prediction with node2vec, using Hadamard product for edge embedding, is given by the following:*

- $\mathcal{O}(d|E_e|)$  when negatives are undersampled in the training set,
- $\mathcal{O}(d|V|^2)$  when all training negatives are used.

*Proof.* The time complexity for getting the node embedding with node2vec is  $\mathcal{O}(d|V|)$  [57], where  $d = 128$ , i.e., the dimensions of the embeddings. . Then, for each training edge, we have to extract the edge embedding by taking the Hadamard product of the node embedding pairs for training links:

$$\mathcal{O}(d \cdot |E_{train}|) \approx \begin{cases} \mathcal{O}(d|E_e|) & \text{with negatives undersampling} \\ \mathcal{O}(d|V|^2) & \text{with full training data.} \end{cases}$$

Finally, for link prediction, we train a logistic regression model, whose complexity is  $\mathcal{O}(d \cdot |E_{train}|)$ , hence the above

equation also holds for the training phase. Thus, the thesis follows.  $\square$

**Theorem 4.** *The time complexity of link prediction using SEAL is:*

- $\mathcal{O}(d|E_e||\Gamma(i)|^h)$  when negatives are undersampled in the training set,
- $\mathcal{O}(|V|^2|\Gamma(i)|^h)$  when all training negatives are used.

*Proof.* SEAL starts with extracting enclosing subgraphs of depth  $h$  for each training link. We denote the edge set of the enclosing subgraphs of depth  $h$  as  $E_{sub}^h$ . Note that  $|E_{sub}^h| \approx 2|\Gamma(i)|^h$ , and  $|E_{sub}^h|$  approaches  $|E|$  as  $h$  gets larger. This phase requires  $\mathcal{O}(|E_{train}| \cdot |E_{sub}^h|)$ . Then, each enclosing subgraph is fed to DGCNN, whose complexity is linear in the number of edges of the training graph [38]. Since DGCNN is applied to each enclosing subgraph, the complexity of this phase is given by  $\mathcal{O}(|E_{train}| \cdot |E_{sub}^h|)$ . When SEAL is used together with *node2vec* embeddings, we have to also consider an initial  $\mathcal{O}(d|V|)$  (with  $d = 128$ ) for computing the embeddings, which can also be approximated as  $\mathcal{O}(d|E_e|)$ .  $\square$

We conclude this section by comparing the complexity of the best-performing link prediction algorithms according to the results of Section V: circle-aware supervised link prediction (with logistic regression or naïve Bayes), *node2vec*, and SEAL. Considering the case where negatives are undersampled (more effective for all strategies, including self-supervised ones), all the best-performing algorithms are linear in the number of edges. Hence, our circle-aware approach is able to outperform state of the art solutions without worsening the computational complexity.