University of Granada

Department of Computer Science
and Artificial Intelligence

Supervised Data Mining in Networks: Link Prediction and Applications

Doctoral Thesis

Víctor Martínez Gómez

Supervisors

Fernando Berzal Galiano

Juan Carlos Cubero Talavera

Programa de Doctorado en Tecnologías de la Información y la Comunicación

PhD Program in Information and Communication Technologies

Granada, May 2018

Editor: Universidad de Granada. Tesis Doctorales
Autor: Victor Martínez Gómez
ISBN: 978-84-9163-982-4
URI: http://hdl.handle.net/10481/53613

El doctorando Víctor Martínez Gómez y los directores de la tesis Fernando Berzal Galiano y Juan Carlos Cubero Talavera garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, Mayo 2018.

The doctoral candidate Víctor Martínez Gómez and the thesis supervisors Fernando Berzal Galiano and Juan Carlos Cubero Talavera guarantee, by signing this doctoral thesis, that the work has been carried out by the doctoral student under the direction of the thesis directors, and as far as we know, in the realization of the work, the rights to be cited of other authors have been respected when their results or publications have been used.

Granada, May 2018.

Víctor Martínez Gómez

Fernando Berzal Galiano

Juan Carlos Cubero Talavera

Esta tesis doctoral ha sido desarrollada con la financiación de la ayuda con referencia BES-2013-064699 bajo el plan *Ayudas para Contratos Predoctorales para la Formación de Doctores 2013* y con la financiación del proyecto con referencia TIN2012-36951 bajo el plan *Proyectos Nacionales de Investigación*, adscritos al Ministerio de Economía, Industria y Competitividad.

This doctoral thesis is partially supported by the Spanish Ministry of Economy and the European Regional Development Fund (FEDER), under grant *Ayudas para Contratos Predoctorales para la Formación de Doctores 2013* with reference BES-2013-064699 and grant *Proyectos Nacionales de Investigación* with reference TIN2012-36951.

Table of Contents

Resumen

La predicción de enlaces consiste en predecir la existencia de enlaces no observados actualmente o enlaces que aparecerán en el futuro entre pares de nodos en redes complejas. Este problema ha atraído la atención de investigadores en diversas disciplinas debido a su utilidad en una amplia gama de aplicaciones, entre las que se encuentran la identificación de genes asociados a determinadas enfermedades o la mejora de las sugerencias realizadas por los sistemas de recomendación. Esta tesis doctoral comprende diferentes líneas de trabajo, todas ellas estrechamente relacionadas con el problema de la predicción de enlaces.

Por un lado, después de un estudio exhaustivo del estado del arte en predicción de enlaces, se identificaron las principales limitaciones de los enfoques actualmente propuestos. Estas limitaciones se relacionaban con las dificultades asociadas al equilibrio entre la escalabilidad y el rendimiento de las técnicas de predicción de enlaces. Se han propuesto dos técnicas escalables de predicción de enlaces que siguen diferentes enfoques para explotar características locales de la red.

Por otro lado, se han abordado diferentes aplicaciones para las técnicas de predicción de enlaces. Se ha propuesto un nuevo algoritmo para priorización genérica, como la priorización de genes asociados a una determinada enfermedad, que logró mejores resultados que otras técnicas gracias a su capacidad para integrar fuentes de datos heterogéneas. También se ha desarrollado un algoritmo para la desambiguación de los sentidos de las palabras en relaciones semánticas entre conceptos, basado en la predicción de enlaces y que no requiere datos anotados. En este trabajo, mostramos cómo nuestro algoritmo logró una mayor precisión que otras técnicas del estado del arte en diferentes tareas de evaluación y cómo las relaciones extraídas pueden usarse para mejorar el rendimiento de las técnicas de última generación para la desambiguación del sentido de las palabras. Además, dado que la función de los nodos influye en cómo se forman los enlaces en redes complejas, hemos desarrollado una nueva métrica de distancia basada en el concepto de equivalencia automórfica con aplicación al descubrimiento de los roles de los nodos.

Finalmente, hemos desarrollado una herramienta de minería de datos para redes complejas. Esta herramienta, llamada NOESIS, contiene implementaciones eficientes de una extensa lista de algoritmos relacionados con redes, incluyendo una biblioteca completa de técnicas de predicción de enlaces.

Abstract

Link prediction is the problem of predicting the existence of currently-unobserved links or links that will appear in the future between pairs of nodes in complex networks. This problem has attracted a great deal of attention from researchers in diverse disciplines due to its applicability in a wide range of tasks, such as the identification of disease-associated candidate genes or the improvement of recommendations suggested by recommender systems. This PhD dissertation comprises different lines of work, all of them closely related to the link prediction problem.

On the one hand, after an exhaustive study of the state of the art in link prediction, the main limitations of currently proposed approaches were identified. These limitations were related to the difficulties associated to the trade-off between scalability and performance in link prediction techniques. Two scalable link prediction techniques were proposed that follow different approaches to exploit local network features.

On the other hand, different applications of link prediction techniques were addressed. We proposed a novel algorithm for generic prioritization, such as disease-gene prioritization, which achieved better results than other state-of-the-art techniques due to its capacity for integrating heterogeneous data sources. We also developed a novel algorithm for word sense disambiguation of semantic relations between concepts, based on link prediction and without the requirement of annotated data. We showed how our algorithm achieved better accuracy than other state-of-the-art techniques in different evaluation tasks and how relations extracted using our approach could improve the performance of state-of-the-art general-purpose word sense disambiguation techniques. In addition, since node role influences how links are formed in complex networks, we developed a novel distance metric based on the concept of automorphic equivalence with application to node role discovery.

Finally, we developed a software framework for network data mining. This framework, called NOESIS, contains efficient implementations of an extensive list of network-related algorithms, including a complete library of link prediction techniques.

Chapter I

PhD dissertation

The aim of this chapter is to describe the context of this doctoral dissertation. In the first section, we introduce and motivate the addressed problem. The second section is devoted to the introduction of the essential concepts and terminology used throughout this dissertation. In the third section, the objectives set for this dissertation are described. The results achieved in this thesis, and the resulting associated publications, are summarized in the fourth section. In the fifth section, concluding remarks are presented. Finally, the sixth section proposes future lines of work that arise from the research described in this dissertation.

1 Introduction

Leonhard Euler laid the foundations of graph theory when working in the problem of the *Seven Bridges of Königsberg* in 1736. Despite the prominence of this area of Mathematics, most work was done over the centuries from a theoretical perspective on the study of small graphs [1]. However, new graph-related problems and challenges have arisen as the amount of available data has increased dramatically over the last few decades [2].

Nowadays, many systems are composed of thousands or even millions of entities or agents with relationships or interactions between them. These systems exhibit complex dynamics and are studied in many different fields. These complex networks arise in very different domains, such as social networks [3], transport networks [4], or protein-protein interaction networks [5], just to mention some examples. The ubiquity of these networks and the limitations of graph theory led to the creation of a whole new area of scientific research that tries to model and characterize these complex systems on the foundation of other well-established areas, including graph theory [6].

Network theory involves the resolution of very different problems in the context of these complex networks, many of which have important applications in real world. For example, different topological metrics have been proposed in network theory for the quantification of interesting features and behaviors present in complex networks. Metrics related to network robustness can be used to build more resilient infrastructures, such as communication networks or power grids [7]. Network theory also involves the study of the role of nodes in complex networks [8]. For example, in social networks, some nodes represent people that could be considered as heavily influential for an specific group, while other nodes are best described as bridges between these groups. The task of identifying these roles is known as role discovery and has many interesting applications, including entity resolution [9] and anomaly detection [10]. Another widely-studied problem in network theory is community detection [11], which is the task of identifying groups or communities in complex networks. These communities, which summarize the structure of the network, are exploited in many interesting applications such as targeted marketing, which identifies different segments of population with similar interests in social networks [12]. These are just some examples of problems that are studied by network theory, but there are many more, such as network visualization techniques that generate pleasant visual representations of networks [13] or network formation models, which are theoretical models used to generate random networks that exhibit similar features to those observed in real-world networks [14].

The present dissertation focuses on the link prediction problem [15], which is the problem of inferring missing links that have not been observed or predicting links that will appear in the future in a network. There is a wide range of applications that motivate the study of this problem. For example, the performance of collaborative filtering recommender systems, which compute recommendations for users based on the preferences of similar users, can be improved by exploiting link prediction techniques to find more similar users [16]. Similarly, the Quality of Service (QoS) in mobile ad hoc networks can be improved by reducing the ratio of data packet loss using link prediction techniques to estimate most robust communication routes [17]. Another interesting application is the detection of anomalous messages in communication networks, such as email networks, by identifying very unlikely links using link prediction techniques [18]. In the context of bioinformatics, the onset of future diseases in patients can be predicted using link prediction techniques in comorbidity networks representing co-occurrence of diseases [19]. Closely related to the previous problem, link prediction techniques can be used to identify new candidate genes that may be associated to complex diseases using protein-protein interaction networks and disease similarity networks [20].

The link prediction problem requires understanding and modelling the dynamics that drive the formation of links in networks, which vary across networks from different domains. The link prediction problem is a supervised classification problem where the instances are pairs of nodes in the network with two possible labels, one representing link presence and other representing link absence. Predicting interactions in complex networks involves facing a series of challenges. To begin with, most real-world networks exhibit highly dynamic and stochastic behaviors that can be difficult to capture by machine learning models. Furthermore, the number of possible pairs of nodes, and therefore of instances in the problem, grows quadratically with respect to the number of nodes in the network. This situation implies the need for highly scalable techniques, since complex networks can be composed of thousands or even millions of nodes. In addition, the link prediction problem is a highly-imbalanced classification problem, because the number of unconnected pairs of nodes in a complex network is usually orders of magnitude greater than the number of connected pairs. This must be taken into account given that some machine learning techniques show difficulties when dealing with imbalanced datasets. Finally, a major concern in the link prediction problem is the presence of uncertainty and noise, as the absence of a link in the network does not imply the actual absence of a relationship or interaction between the entities represented by the pair of nodes that the link would connect. The absence of reliable negative instances transform the link prediction problem in a positive and unlabeled learning problem, usually known as PU learning.

Fortunately, despite these difficulties, the topology of the network and the features of nodes and links can be exploited to predict these missing or future links with reasonable accuracy in many cases [21]. Different network formation models have been proposed in the literature to capture the link formation dynamics observed in real-world networks. For example, the preferential attachment mechanism was proposed by Albert-László Barabási and Réka Albert to explain why, in some networks, there are nodes with a very high number of links in comparison to the rest of the nodes in the network [14]. In these networks, known as scale-free networks, the more links a node has, the more likely it is to form new links with other nodes. One of the most prominent examples of this type of networks is the Internet network [22].

However, the preferential attachment model has severe limitations to explain most of the links that are observed in real-world networks. Empirical observations also suggest that nodes tend to form links with similar nodes, a phenomenon known as homophily. In the context of link prediction in complex networks, the definition of similarity is problem-dependent: what may work well in some networks could perform badly in other networks. On the one hand, since homophily has been highly observed in social networks, where people tend to form links with persons with similar "sociodemographic, behavioural, and intrapersonal characteristics" [23], similarity could be defined in terms of node features. On the other hand, similarity could also be defined in terms of the topology of the network, such as triadic closure, where two nodes are more likely to be connected if they are close in the network or if they are connected to the same nodes [24].

The present dissertation addresses different topics related to the link prediction problem. An exhaustive study of the existing link prediction approaches has been performed. After identifying the limitations of existing techniques, we propose two efficient link prediction techniques based on different approaches. First, we introduce an adaptive approach based on the novel observation of the existence of a correlation between topological properties of the network and the definition of node similarity based on local network features exhibiting best performance. Second, we consider a probabilistic approach that aggregates evidence from the topology of the network to estimate the likelihood of link existence.

A contribution of this thesis will be the development of several new applications for link prediction techniques. A novel technique for candidate disease-gene prioritization using a link-prediction-based approach through propagation of information on heterogeneous networks will be proposed. The presented technique outperforms other state-of-the-art methods in different evaluation tasks. We will also tackle the problem of disambiguating semantic relations using a novel approach based on ideas gathered from the link prediction problem. Our experimentation will show how our method outperforms other state-of-the-art alternatives, and how our proposal can be applied to improve the performance of state-of-the-art word sense disambiguation techniques.

Side contributions will also arise as a result of our main line of study. On the one hand, major contributions will be performed to NOESIS, a high-performance network data mining framework, including the implementation of an extensive collection of link prediction techniques. On the other hand, we will propose a novel distance metric, based on the concept of automorphic node equivalence, which can be used in the role discovery task with potential applications to the link prediction problem.

2 Preliminaries

This section introduces all the major foundational concepts required to understand this dissertation. In its first subsection, basic concepts and terminology from graph theory and network science are introduced. Its second subsection is devoted to the formal definition of the link prediction problem and the strategies used for the evaluation of the performance of link prediction techniques.

2.1 Basic concepts

Graph theory is the branch of mathematics that studies graphs, which are mathematical structures used to represent relations between objects. A graph $G = (V, E)$ is a structure composed of a set V of objects, which are called vertices. Pairs of vertices are connected together by a set E of connections $e_{x,y}$, where $x, y \in V$, which are called edges when they represent undirected relations and arcs when they represent directed relations. In the multidisciplinary field of network theory, graphs with attributes are the key data structure used to represent and study complex networks. In this field, it is a convention to denote the vertices as nodes and the arcs and edges as links. For the sake of homogeneity, the network theory convention will be used for the rest of the dissertation.

A node is said to be a neighbor of or adjacent to other node if both are connected by a link. The neighborhood of a node x is the set Γ_x comprised of its neighbor nodes. The cardinality of this set is the degree of the node; in other words, the degree of a node is defined as the number of links connected to the node. In directed networks, we can make a distinction between in-degree, out-degree, and total-degree, defined as the number of incoming links, the number of outgoing links, and the sum of the number of incoming and outgoing links, respectively.

An important node-related measure is the clustering coefficient, which measures the tendency of a node to cluster with other nodes, forming triangles. The clustering coefficient of a node x is a value ranging between 0 and 1. It can be computed as

$$C_x = \frac{|\{e_{y,z} : \forall y, z \in \Gamma_x, e_{y,z} \in E\}|}{|\Gamma_x|(|\Gamma_x| - 1)}.$$

A path is a sequence of distinct consecutive links connecting a pair of nodes in the network where the same node can only be visited once. The length of a path is defined as the number of links that the path contains. The shortest path between two nodes is defined as the path between the two nodes with smaller length. The diameter of a network is equal to the length of the longest shortest path between two nodes in the network.

A network can be also represented as a square $|V| \times |V|$ matrix, known as the adjacency matrix. In unweighed networks, the elements of the adjacency matrix A are defined as

$$A_{i,j} = \begin{cases} 1 & \text{if } e_{i,j} \in E \\ 0 & \text{otherwise.} \end{cases}$$

In weighted networks, where a weight $w_{i,j}$ is associated to each link, the elements of the adjacency matrix are defined as

$$A_{i,j} = \begin{cases} w_{i,j} & \text{if } e_{i,j} \in E \\ 0 & \text{otherwise.} \end{cases}$$

It should be noted that the adjacency matrix must always satisfy $A_{i,j} = A_{j,i}$ for undirected networks.

A random walk in a network is a stochastic process simulating a single walker randomly moving through adjacent nodes in the network. Random walks can be used to iteratively estimate the probability distribution of reaching a node $v_j \in V$ starting from a node $v_i \in V$. This probability distribution P can be iteratively computed as

$$\vec{P}(t) = M^T \vec{P}(t-1),$$

where M is the transition probability matrix defined as $M_{i,j} = A_{i,j} / \sum_k A_{i,k}$ and $\vec{P}(0)$ is a vector representing the probability mass function of the starting position of the walker. This expression is iteratively applied until convergence, when the absolute value of the sum of the differences between two consecutive iterations is smaller than a predefined value indicating what can be considered a negligible difference.

2.2 The link prediction problem

The link prediction problem is defined as, given a complex network represented as a graph $G = (V, E)$, and assuming that some links are missing or will appear in the future, predicting or estimating the likelihood of the existence of a link between each pair of unconnected nodes in V. Link prediction techniques compute a probability or score for each pair of unconnected nodes $x, y \in V$ (i.e., $e_{x,y} \notin E$) reflecting the likelihood that there is a link between both nodes.

The link prediction problem is a supervised classification problem, which aims to predict the actual class of each pair of nodes. Two different classes are considered in this problem: link existence and link absence. Therefore, the most common validation methodology for link prediction techniques is the use of a training set, a test set, and an optional validation set. Although other approaches can be followed, the most common approach is splitting the set of links E to create these sets. Links in the training set must be used by link prediction techniques to try to predict links in the test set. The validation set is used to tune the hyperparameters, if any, of the link prediction technique.

A popular approach in the evaluation of classification tasks is cross-validation, a validation technique comprising rounds of evaluation over complementary subsets of the data. In k-fold cross-validation, the original set of links E is split into k disjoin subsets. For each evaluation round, $k-1$ subsets are combined and used as the training set while the remaining subset is used as test set. This process is repeated k times so that each subset acts as test set once.

Two methodologies can be used to measure the performance of link prediction techniques. The first one is the approach commonly used in traditional classification, where the classifier must output the predicted class and the evaluation is performed by comparing the predicted class against the actual class. Since most link prediction techniques usually output a score or probability proportional to the likelihood of the existence of a link, a second approach is based on ranking all pairs of nodes according to these scores and returning the top t pairs of nodes as positive instances, where t is equal to the number of links in the test set. The rest of pairs of nodes are returned as negative instances, predicting that they are unconnected.

		Actual class	
		Positive (link)	**Negative (no link)**
Predicted class	**Positive (link)**	True positive (TP)	False positive (FP)
	Negative (no link)	False negative (FN)	True negative (TN)

Table 1: Confusion matrix for a link predictor.

As in any supervised classification problem, a confusion matrix can be defined for a link predictor. The confusion matrix describes the performance of the classifier by counting the number of instances according to their actual and predicted class, as shown in Table 1. Different evaluation metrics can be defined using the values encoded in the confusion matrix. A metric typically used in the evaluation of classification problems is accuracy, defined as the fraction of instances that are correctly classified. Accuracy is computed as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

However, when evaluating link prediction, we must take into consideration that we are dealing with an imbalanced classification problem, since most real-world networks are sparse. This means that the number of non-existing links is usually orders of magnitude larger than the number of existing links. A trivial link prediction technique assigning to every pair of nodes the class corresponding to link non-existence would achieve a very high accuracy in most cases. In order to perform a proper evaluation, we must rely on other measures like precision and recall. Precision and recall are defined as

$$Precision = \frac{TP}{TP + FP} \qquad\qquad Recall = \frac{TP}{TP + FN}.$$

Since trying to maximize both metrics is a multi-objective problem, both scores are typically combined into one score, called F1 score, by computing their harmonic mean. When working with the ranking-based evaluation approach, precision can be used as a standalone evaluation metric because the number of positive instances that link prediction models will return is fixed beforehand to be equal to the size of the test set.

A powerful tool for evaluating classifiers, widely used to evaluate the performance of link prediction techniques, is the receiver operating characteristic (ROC) curve [25], which is the result of plotting the true positive rate, also known as sensitivity, as a function of the false positive rate, one minus specificity. The closer the curve is to the upper-left corner of the plot the better is the classifier. The curve of a random classifier matches with the diagonal from the bottom-left corner to the upper-right corner. The AUC, which stands for the area under the ROC curve, is a value ranging from zero to one that is equal to the probability of the classifier ranking a positive example better than a negative example. Therefore, higher AUC values indicate better classification performance, where an AUC of 1 represents a perfect classifier and a value of 0.5 represents a completely random classifier. In the context of link prediction, the AUC value can be asymptotically approximated by sampling random pairs of instances. These pairs of instances must consist of a pair of connected nodes considered the positive instance and a pair of unconnected nodes considered the negative instance. The AUC value can be approximated as

$$AUC \approx \frac{n_1 + 0.5n_2}{n},$$

where n is the number of pairs of samples considered, n_1 is the number of properly-ranked sample pairs, for which the positive instance is ranked higher than the negative instance; and n_2 is the number of pairs where both instances were equally ranked.

3 Objectives

The main objectives of this dissertation revolve around the link prediction problem in complex networks, both in the development of new link prediction techniques and the study of novel applications related to this problem. In particular, the present dissertation is organized around the following objectives:

- **Analysis of existing link prediction techniques.** We have studied the state of the art of link prediction techniques. Our study was motivated by the new techniques that had been developed since the publication of the latest relevant reviews of the field [15, 21]. Our review provides a classification of the different link prediction techniques according to the approach that they follow. We have also studied their computational complexity and carried out some experiments to evaluate their predictive strength.
- **Adaptive link prediction.** Similarity-based link prediction techniques are commonly used due to their high scalability. However, these techniques do not learn from the network and their performance strongly varies from network to network. One of our main goals was to study how these techniques are related and what properties of the network influence their performance. As a result of our study, we have proposed an adaptive link prediction technique.
- **Local probabilistic link prediction.** Probabilistic techniques can be used to model the uncertainty present in the link prediction problem, where link absence does not guarantee absence of actual relationship. However, these techniques present a high computational complexity in the phase required for estimating the parameters of the model. To solve this limitation, we have proposed a probabilistic model based on aggregating evidence from local features.
- With respect to the applications of link prediction, we address several problems listed below:

  - **Prioritization using heterogeneous data.** Gene prioritization refers to a family of computational techniques for inferring disease genes. We propose a link prediction-based approach, which integrates heterogeneous networks.
  - **Disambiguation of semantic relations.** We propose a novel knowledge-based approach for the disambiguation of semantic relations using redundancy of some knowledge bases.
  - **Automorphic distance metric for node role discovery.** We propose a novel distance metric capturing the concept of automorphic distance, which measures how far two nodes are from playing the same topological role in the network. We also show how generating vector representations based on the application of multidimensional scaling on automorphic distance leads to improved embeddings that capture role information better than previously-proposed techniques.

- **Development of a framework for network data mining.** Other goal of this thesis is the development of new features for the NOESIS open-source framework for network data mining, specially those features related to link prediction. In addition, we also develop a Python API for NOESIS.

4 Discussion of results

In this Section, we summarize the proposals and the results we have obtained within this dissertation.

4.1 Link prediction

This Subsection is devoted to the analysis of existing link prediction techniques, the proposal of adaptive link prediction, and the study of local probabilistic link prediction.

4.1.1 Link prediction: The state of the art

The challenging task of link prediction has attracted a lot of attention from the scientific community as result of the many applications that this problem has in the real world. Progress in this field has led to the development of a large number of approaches modelling different connectivity patterns. There have already been extensive surveys in the past for the link prediction problem [15, 26, 21, 27]. However, a new review of the state of the art of link prediction was desirable as new techniques have been developed since the publication of these surveys.

The link prediction problem has been tackled following very different approaches. One of our major contributions is the proposal of a two-level taxonomy to classify the wide spectrum of existing link prediction techniques (see Figure 1). In the first level of our proposed taxonomy, techniques have been categorized as similarity-based methods, probabilistic and statistical methods, algorithmic methods, and preprocessing methods.

Similarity-based methods rely on the assumption that nodes tend to form connections with similar nodes. These techniques take the form of a function that assigns a similarity

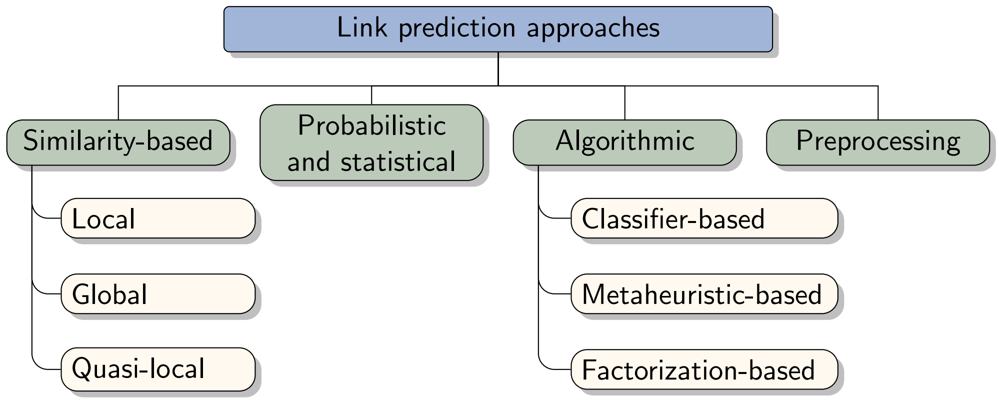


Figure 1: Proposed taxonomy for link prediction techniques.

score, where more similarity implies more probability of link existence, for a pair of given nodes based on the topology of the network. Different techniques capture different definitions of similarity. In the second level of the proposed taxonomy, similarity-based techniques are classified according to the range of topological information that they take into account: local approaches, which only use direct neighborhood information, global approaches, which consider the whole topology of the network for estimating the similarity between any pair of nodes, and quasi-local approaches, which are an intermediate point between local and global approaches. The most basic local link prediction technique, that surprisingly works very well in many cases, is counting the number of shared neighbors between nodes [28]. However, assuming that all shared neighbors contribute to the same extent is a naïve approach. The Adamic-Adar index [29] and the Resource Allocation index [30] penalize the contribution of each shared neighbor proportionally to its degree. Other techniques, such as the Jaccard index [31] or the Hub Promoted index [32], penalize node similarity proportionally to the amount of non-shared neighbors. Otherwise, global similarity-based techniques usually consider existing paths between nodes. The most basic approach would be the negated length of the shortest path between a pair nodes. However, this approach obtains poor results in practice as result of not taking into account indirect paths [33]. The Katz index, which can be computed in closed form, solves this limitation by summing the influence of all possible paths between the considered nodes [34]. Other family of global techniques is based on estimating the probability of reaching the target node starting from the source node by means of random walks. Random Walks with Restart [35] and Flow Propagation [36] are examples of this type of techniques. Finally, quasi-local approaches are usually restricted versions of global techniques, such as the Local Path Index [37], highly inspired by the Katz index, or Local Random Walks [38], based on length-restricted random walks.

The category of probabilistic and statistical methods is comprised by techniques that assume an a priori known network structure with unknown parameters that are estimated using statistical analysis and probability theory. Once the parameters that best fit the network have been estimated, the obtained model can be used to predict the likelihood of existence of each non-observed link. For example, the hierarchical structure model assumes a hierarchical organization of the network, where hub nodes act as bridges between communities of highly connected nodes [39]. A different approach, known as the stochastic block model, assumes that the network is organized in communities or blocks of densely connected nodes [40]. The parameters of this model, which are the probabilities of link between nodes of each possible pair of blocks, are estimated maximizing the likelihood of the model given a network. Other examples of techniques in this group are the cycle formation model [41], based on estimating the probability of existence of cycles of certain length, and the local co-occurrence model [42], based on the existence of relevant nodes in the paths between other nodes.

Since the link prediction problem is a supervised classification task, an evident approach is the use of machine learning techniques, such as classifiers or metaheuristics. This kind of learning and optimization techniques are categorized in our taxonomy as algorithmic methods. There are many studies on the application of most popular algorithms for classification to the link prediction problem [43]. The problem of class imbalance and their poor scalability limits their applicability in many situations. Recently, metaheuristic optimization algorithms like evolutionary algorithms have been used to estimate the influence of different possible factors driving the dynamics of link formation in networks [44]. Matrix factorization techniques, which have been widely used in recommender systems, are also included in this category. These techniques are based on the factorization of the adjacency matrix of the network in order to learn latent features explaining the existence or absence of links in the network [45].

Finally, preprocessing methods comprise the last group in our taxonomy. These approaches aim to reduce the noise present in the network by removing weak or spurious links. For example, the low-rank approximation of the adjacency matrix can be used to remove less relevant links while maintaining the overall structure of the network [46]. Other techniques like unseen bigrams, based on the idea of substituting nodes by similar elements, and filtering, based on using other link prediction techniques to score the strength of links and removing weakest ones, are also included in this section [33].

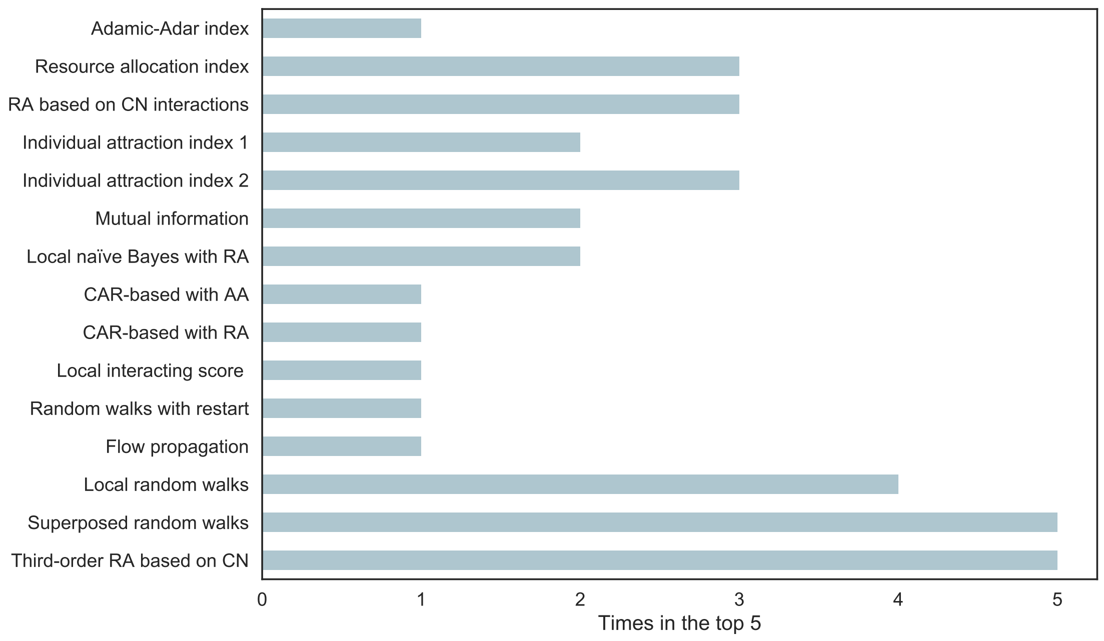


Figure 2: Number of times among the 5 best techniques in experiments with 40 techniques on 7 networks. Abbreviations stand for common neighbours (CN), Adamic-Adar index (AA), and Resource Allocation index (RA).

Another contribution that we made in our review of the state of the art of link prediction is the theoretical study of the computational complexity of similarity-based techniques. In addition, an exhaustive empirical validation was carried out by applying these link prediction techniques to complex networks from different domains in order to measure and compare their precision using fivefold cross-validation. The results obtained in our experimentation show that no technique is strictly better in terms of precision and that local and quasi-local techniques achieve very competitive results, sometimes even better, compared to global techniques. A summary of obtained results is shown in Figure 2, where the number of times each technique appears among the top five techniques with best precision is indicated, including only techniques appearing at least once among the best five.

The journal article associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. A survey of link prediction in complex networks. ACM Computing Surveys 49(4):69:1-69:33, 2017. DOI 10.1145/3012704.

4.1.2 Adaptive link prediction

Despite their simplicity, local similarity-based link prediction techniques remain as a relevant approach for the link prediction problem due to their reasonable precision and their high scalability, which is a critical requisite in many practical applications. Due to the large number of similarity-based techniques proposed in the literature and the difficulty to know which techniques will work better in practice, the current strategy is to manually evaluate different approaches in order to choose the definition of similarity that achieves the best performance in a particular context.

In order to overcome this practical limitation while maintaining the benefits in term of scalability of similarity-based techniques, we proposed an adaptive link prediction technique. We aimed to develop a novel approach that could adapt to the dynamics of different networks without requiring a learning phase. We hypothesized about the possibility of using topological characteristics of the network to estimate which technique would perform better for a given network.

To test our hypothesis, we derived an expression that generalizes different degree-penalization local similarity-based techniques under the equation

$$S(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} |\Gamma_z|^{-\alpha},$$

where x and y are the pair of nodes for which link likelihood is being estimated, Γ is the set of neighbors of a given node, and α is the adaptive parameter. It is straightforward to see how this expression is equivalent to the common neighbors technique when $\alpha = 0$, to the resource allocation index when $\alpha = 1$, and closely approximates the Adamic-Adar index when $\alpha \approx 0.37$.

One of our main contributions in this work is the study of how the parameter α of our proposed generalization relates to structural features of the network. For our experimentation, we gathered a set of fifteen networks, from very different domains, exhibiting very heterogeneous features. We evaluated the precision of our proposal on these networks for values of α between -1 and 2 in steps of size 0.1. The performance curves exhibited a pronounced convex shape, showing how the proper estimation of the α parameter has a critical impact in the achieved accuracy.

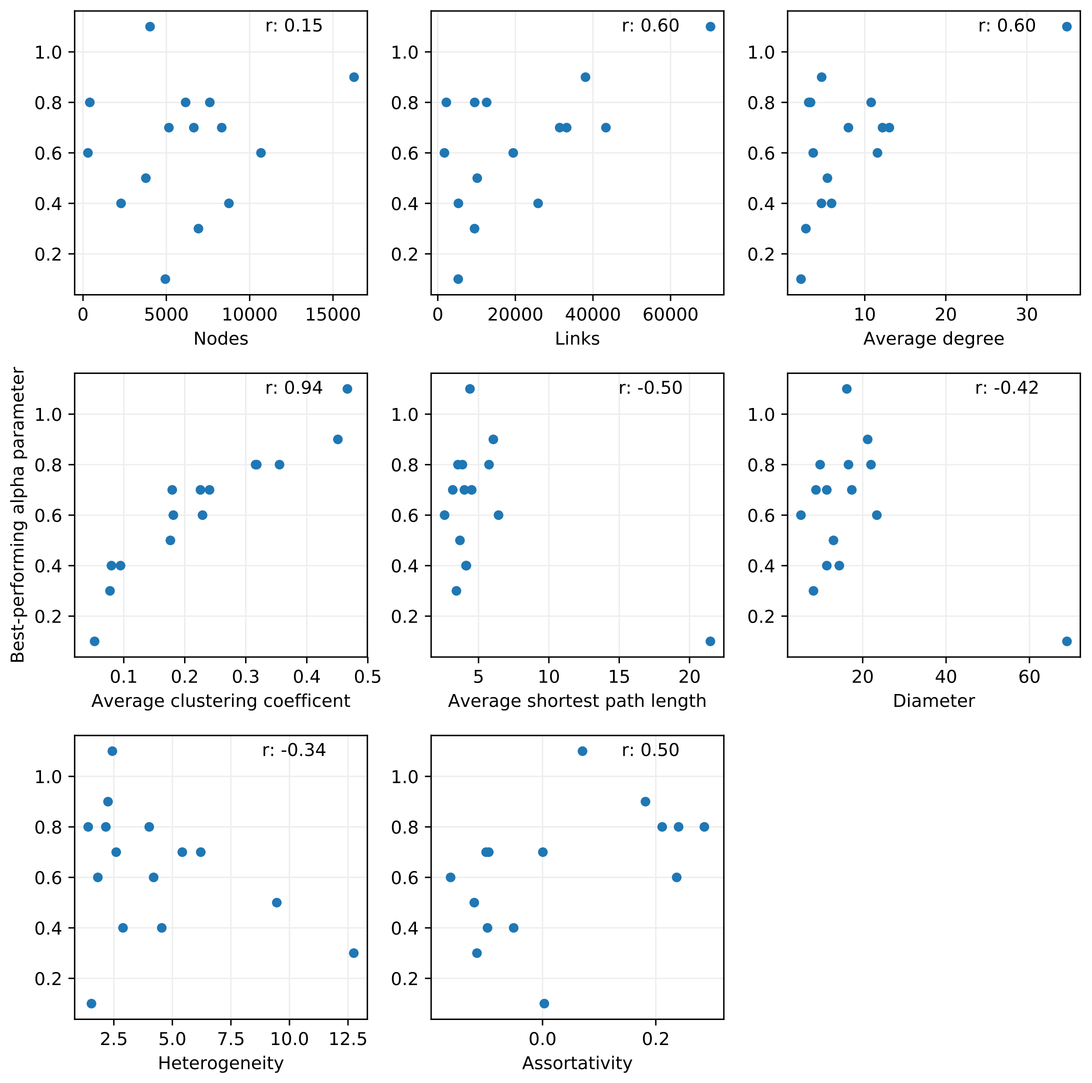


Figure 3: Relationship between the best-performing alpha value and different topological properties for a set of 15 networks. The r value is the Pearson correlation coefficient.

Our next step was to study if the best performing value for α was related to some topological property of the network. In order to carry out this test, we computed the Pearson correlation coefficient between the value of α in our expression that achieved the best results and different global topological features of the network, including the number of nodes in the network, the number of links in the network, the average degree, the average clustering coefficient, the average shortest path length, the diameter, the heterogeneity, and the assortativity. Our experimentation showed that most of these properties did not show a statistically significant correlation with the optimal value of α (see Figure 3). However, average clustering coefficient showed a very high and statistically significant correlation, suggesting that this feature could be used to accurately predict an approximation of the optimal value for α, and therefore could be used to estimate the most adequate definition of similarity, based on shared neighbors penalized by their degree, between nodes in the network.

In the light of these results, we introduced the average clustering coefficient in our proposed generalization as

$$S(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} |\Gamma_z|^{-\beta C},$$

where C is the average clustering coefficient of the network and β is a constant that had to be estimated. Linear regression without intercept on the relation between the average clustering coefficient and the best performing value for α in the previously mentioned networks was used to estimate $\beta \approx 2.5$ as the optimal value for this constant. It should be clarified that β remains as a constant regardless of the network where this similarity-based technique is applied, where the average clustering coefficient of the network is the variable element from one network to another.

Finally, in order to evaluate the precision of our approach, we performed a validation over a set of seven test networks and measured the precision and AUC achieved compared to different similarity-based link prediction techniques. The obtained results showed that our approach consistently achieved the best results in most cases. The Friedman test confirmed that this improvement was statistically significant.

The journal article associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. Adaptive degree penalization for link prediction. Journal of Computational Science 13:1-9, 2016. DOI 10.1016/j.jocs.2015.12.003.

4.1.3 Probabilistic local link prediction

After our review of the state of the art of the link prediction problem, we realized that most probabilistic approaches had severe limitations in their applicability in practice due to the high computational complexity of estimating the parameters of their models. However, scalable probabilistic approaches based on local features have been proposed recently. These approaches exhibit remarkable precision as they can satisfactorily use probabilistic models to capture the uncertainty inherently present in the network, where the absence of a link between a pair of nodes may be the result of the non-existence or the non-observation of a link that could actually exist.

In the light of these results, we developed a probabilistic approach based on the accumulation and aggregation of evidence of link existence. If the event corresponding to the existence of a link between nodes x and y is denoted as L_{xy} and the set of shared neighbors between these nodes is denoted as $\Gamma_{x\cap y}$, according to Bayes' theorem we could write

$$P(\overline{L_{xy}}|\Gamma_{x\cap y}) = \frac{P(\Gamma_{x\cap y}|\overline{L_{xy}})P(\overline{L_{xy}})}{P(\Gamma_{x\cap y})}.$$

Conditional independence between evidences from shared neighbors was assumed, so this expression was rewritten as

$$P(\overline{L_{xy}}|\Gamma_{x\cap y}) = \prod_{z \in \Gamma_{x\cap y}} \frac{P(z|\overline{L_{xy}})}{P(z)} P(\overline{L_{xy}}).$$

Despite the independence assumption is technically wrong in most cases, it has been empirically proven that good results can be obtained as many problems are satisfactorily tackled using approaches that make this assumption, such as the naïve Bayes classifier [47].

Since according to Bayes' theorem, the term $P(z|\overline{L_{xy}})/P(z)$ is equal to the term $P(\overline{L_{xy}}|z)/P(\overline{L_{xy}})$ and since $P(L_{xy}) = 1 - P(\overline{L_{xy}})$, this expression can be rewritten as

$$P(L_{xy}|\Gamma_{x\cap y}) = 1 - \prod_{z\in\Gamma_{x\cap y}} \frac{1-P(L_{xy}|z)}{1-P(L_{xy})}(1-P(L_{xy})),$$

where $P(L_{xy}|z)$ is the probability of link between x and y given the shared neighbor z. The definition of this probability is flexible, but we propose a simple approach for estimating this value as the probability of the existence of a link in the network between a pair of nodes given a shared neighbor of the same degree than z, which can be computed as

$$P(L_{xy}|z) = \frac{1}{|N_{|\Gamma_z|}|} \sum_{k\in N_{|\Gamma_z|}} \frac{\sum_{i\neq j,i,j\in\Gamma_k} P(L_{ij})}{|\Gamma_k|(|\Gamma_k|-1)},$$

where $N_{|\Gamma_z|}$ is the set of nodes with the same degree than z.

In order to assess the performance of our probabilistic approach, we measured the precision of our proposal in ten complex networks from different domains and compared the achieved results with those obtained by popular similarity-based techniques and a local naïve Bayes approach [48]. The experimentation showed that our approach achieved the best results more consistently than the other approaches, also obtaining the best results in average.

The book chapter associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. Probabilistic local link prediction in complex networks. In Proceedings of the 11th International Conference on Scalable Uncertainty Management. Lecture Notes in Computer Science, 10564:391-396, 2017. DOI 10.1007/978-3-319-67582-4.

4.2 Applications

This Subsection is devoted to the applications of link prediction studied within this dissertation, including prioritization using heterogeneous data, the disambiguation of semantic relations, and an automorphic distance metric for node role discovery.

4.2.1 Prioritization using heterogeneous data

Disease-gene prioritization is the task of identifying candidate genes that may be related to specific complex diseases [49]. A typical prioritization technique, given a disease as input, would evaluate the relevance of each gene for that disease based on training data and would return a ranking of genes sorted according to their inferred relevance to the disease. Most of these approaches rely on the fact that associated or interacting biological entities are more likely to share a common function and to be involved in the same processes [50]. The application of this principle allows to tackle this problem as a link prediction problem.

After the study of the state of the art of prioritization techniques, we found that most of them were highly tailored to specific prioritization tasks, disease-gene prioritization mainly. In addition, most of these techniques were designed to exploit very specific types of data, without providing mechanisms for the integration of data of different type. In order to overcome these limitations, we proposed a novel approach called ProphNet.

ProphNet was designed to handle heterogeneous data by abstracting from the biological or clinical domain and, therefore, working over abstract complex networks. Our model works over a set of networks D, each one representing the interactions or similarities between specific elements (e.g., protein-protein interactions or disease phenotype similarities), and a set of bipartite networks R connecting networks from D such that a path should exists between every pair of networks in D, forming a global network.

The proposed approach can handle different prioritization tasks by allowing the user to define the query entity or query set of entities, which are the entities with respect to the user wants to obtain candidate associated entities, and the target network, which is the network from set D representing the entities that the user wants to rank according to their association to the query. For example, for candidate gene prioritization, the studied disease would be the query entity and the protein-protein interaction network would be the target network.

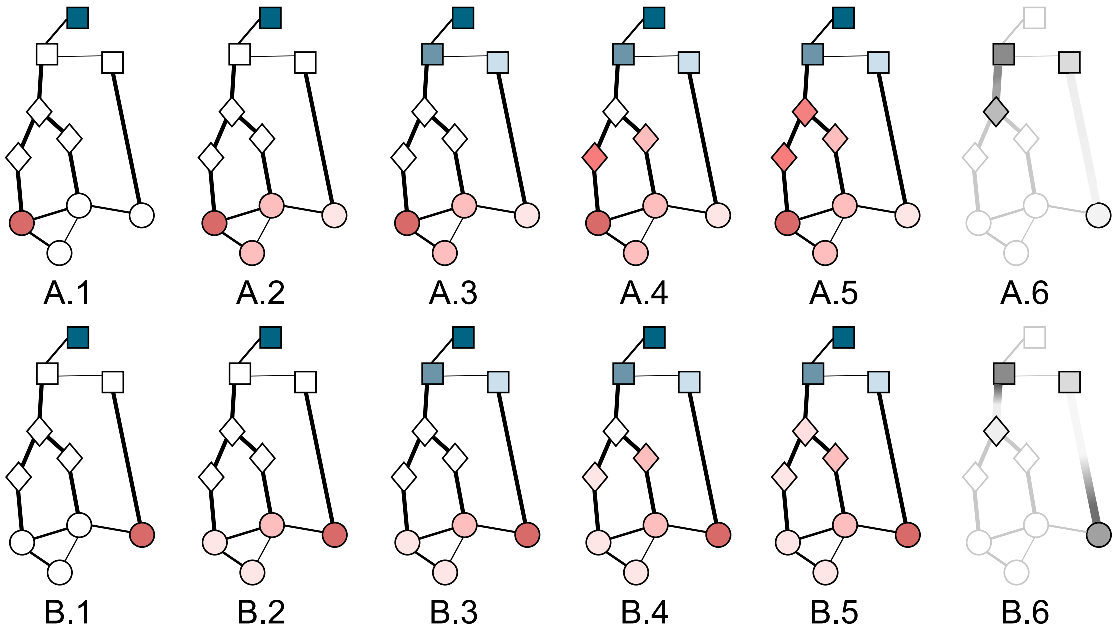


Figure 4: Toy example of two different ProphNet runs: in example A, the query node and the target node are highly correlated, in contrast to example B. Information is propagated from query nodes (colored in red) from query network (containing circled nodes) to target network (containing squared nodes) through an additional network (containing rhomboid nodes).

Our technique carries out the prioritization by performing a propagation of information from the nodes representing the query entities to the nodes in the target network, as shown in Figure 4. This process is performed by alternating two propagation steps for each possible path composed of networks from D and R connecting the query network, which is the network containing the query set of entities, and the target network, which is the network containing the nodes that are going to be ranked according to their relevance to the query set. The first step, which must be applied to make information flow through each network from D, involves an inner propagation of information using the Flow Propagation algorithm [36]. The second step, which is applied to make information flow between networks from D through networks from R, is computed as the average of flow propagated to the neighbors of a node.

This process of information propagation is repeated until the propagated flow reaches each network adjacent to the target network. The relevance to the query set of each entity in the target network is finally computed by measuring the correlation between the scores result of the propagated flow and the scores that are obtained as result of performing a propagation from the evaluated entity in the target network. Since this second propagation is a computationally expensive process, because it must be performed for each node in the target network, we suggest the precalculation of these propagations.

In order to evaluate the precision of our approach in different prioritization tasks, we applied ProphNet over a global network built using different datasets to build the subnetworks. First, a disease phenotype network built by using text mining techniques in the *Online Mendelian Inheritance in Man* (OMIM, [51]) database. The network contained 5080 disease phenotypes with 19729 weighted links connecting most similar phenotypes. Similarity between phenotypes was estimated by counting the occurrences of each term from specific sections of the *Medical Subject Headings Vocabulary* (MeSH). Second, a gene network, represented as a protein-protein interaction network, built with data from the *Human Protein Reference Database* (HPRD, [52]). The network contains 8919 proteins and 32331 interactions. Third, a network of protein domains with 48778 relationships between 5490 domains extracted from DOMINE [53] and InterDom [54]. Fourth, a network of 1393 connections between phenotypes and genes based on data directly extracted from the phenotype-gene relationship field from OMIM. Fifth, a network of relationships between domains and genes extracted from pFam [55]. Sixth, a network of relationships between domains and phenotypes extracted from Pfam and the UniProt database [56].

Two evaluation tests were performed. The first one consisted on a leave-one-out cross-validation gene-disease prioritization task. Furthermore, a validation test using new gene-disease associations added to OMIM between 2007 and 2010 was carried out. We compared the ROC curves and the obtained AUC with those obtained by rcNet [57], a state-of-the-art gene-disease prioritization technique. The second evaluation consisted on a leave-one-out cross-validation for domain-disease prioritization. The achieved results were compared with those obtained by DomainRBF [58], a state-of-the-art domain-disease prioritization technique.

The results obtained in our experiments show that our proposed approach performs substantially better than rcNet and DomainRBF, obtaining statistically significant superior performance scores in the different evaluation tasks. We also performed a robustness test to evaluate how the performance of our proposed technique is affected by hyperparameter selection. Obtained results suggested that ProphNet performance remained pretty consistent within a range of reasonable hyperparameter values.

Finally, we studied some interesting cases applying our approach for disease-gene prioritization. We applied ProphNet to compute the list of candidate genes for Alzheimer, Diabetes Mellitus Type 2, and Breast Cancer. We showed how recent scientific literature related the top ranked genes to these diseases despite these relations were not explicitly present in our training data.

The journal article associated to this part of the dissertation is:

V. Martínez, C. Cano, A. Blanco. ProphNet: A generic prioritization method through propagation of information. BMC Bioinformatics 15(S-1):S5, 2014. DOI 10.1186/1471-2105-15-S1-S5.

4.2.2 Disambiguation of semantic relations

Word-sense disambiguation (WSD) is the problem in natural language processing (NLP) that consists of identifying which is the actual sense or meaning of a word in a text [59]. This task remains as an open problem because the accuracy of these techniques is still far from human performance. A successful approach to WSD is the use of knowledge-based techniques exploiting structured information, such as ontologies, to perform disambiguation of words in texts. However, the acquisition of this knowledge is limited because the manual annotation of these relations by humans is a high time and resource consuming task, leading to the well-known knowledge acquisition bottleneck problem [60].

We proposed an approach to automatically disambiguate semantic relations between concepts based on ideas borrowed from the link prediction problem. Our approach was motivated by the hypothesis that the overlapping between ambiguous relations could be exploited to disambiguate them, as shown in Figure 5. Relations involving similar or even shared senses can mutually reinforce each other. We also hypothesized that these disambiguated relations could enhance the performance of general-purpose knowledge-based WSD techniques.

Our proposed approach relies on a taxonomy of concepts, which is used to propagate evidence between similar concepts. We used WordNet [61], a general lexical database of English, as our taxonomy of concepts as it provides groups of synonymous words, called synsets, and hypernymy/hyponymy relations that make up the taxonomy. Given an ambiguous relation, evidence is propagated across the taxonomy for each possible sense of the two involved words or concepts. Evidence is represented as a tuple containing a unique identifier for the ambiguous relation, a binary label encoding if the term is the source or the target in the relation, and the probability value of the evidence. The propagation process requires propagating this evidence by annotating the same tuple, with updated probability, in adjacent concepts in the taxonomy. Two parameters are introduced. First, α, which is called the upward propagation factor, indicates the probability of a parent node having a property observed in a child node. Second, the parameter β, called the downward propagation factor, indicating the probability of a child node having a property observed in a parent node.

Storing all these evidences in the taxonomy is an infeasible procedure in practice because it would require a number of annotations equal to the number of concepts in the taxonomy for each evidence. Instead, we propose an alternative approach to solve the same problem but only requiring the annotation of evidence in nodes in the path

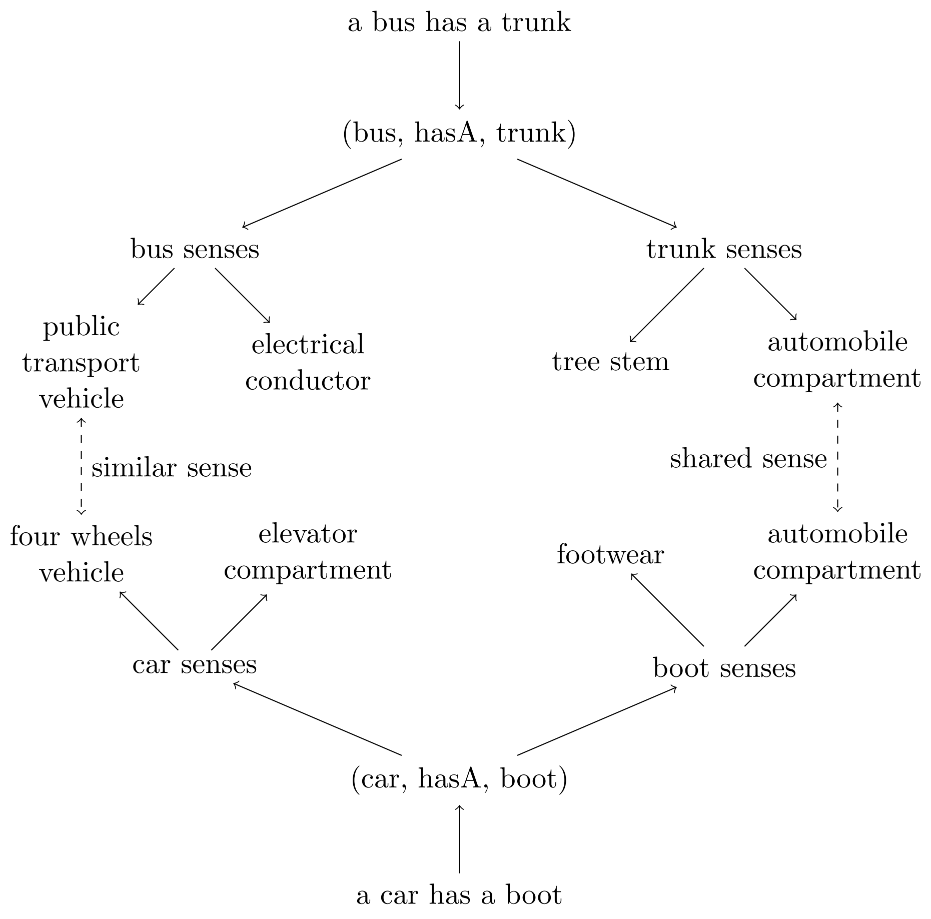


Figure 5: Example of two semantic relations that mutually reinforce each other for specific sense assignments due to similar sense sharing.

from the initially annotated concept to the root of the taxonomy. This new approach implies that the number of annotations becomes proportional only to the depth of the taxonomy.

Our proposed approach works as follows. First, given the collection of ambiguous semantic relations, the evidence is annotated for each of the possible meanings of the concepts involved. Then, these annotations are propagated across nodes in the path from the original nodes containing the initial evidence to the root of the taxonomy. The likelihood of existence of a relation between two concepts is computed by gathering the evidence stored in the nodes associated to these concepts and the evidence stored in nodes in the path from these concepts to the root node. These evidences are aggregated to compute the probability of existence of relation using the evidence aggregation framework that we proposed in our work. Finally, the disambiguation is performed by computing the probability of relation existence for all possible pairs of sense assignments and choosing the most likely one as their correct senses.

We carried out different evaluation tasks in order to measure the accuracy of our approach. Due to the novelty of our proposal, we could not find techniques previously proposed applicable with our problem settings. However, we repurposed different reasonable approaches. First, a random approach, which serves as baseline for our experiments. Second, an approach based on choosing the most frequent sense of words as the right sense. Frequency information was extracted from WordNet, which estimated it from a large corpus of text. Third, an approach choosing the pair of senses with lower shortest path length in the taxonomy. Finally, a state-of-the-art technique for learning semantic vector representations of word senses, known as NASARI [62]. For NASARI, the predicted pair of senses was the one with lowest cosine distance.

For the first evaluation task we created ambiguous relations from different WordNet [61] semantic relations and applied the disambiguation techniques previously described. The obtained results in this experiment showed how our approach outperformed the other approaches by a large margin. We also used this data to show how our approach is robust for variations in the hyperparameters. In the second evaluation task, we applied the compared techniques to disambiguate a popular ambiguous semantic network, known as ConceptNet [63], relying only on the WordNet taxonomy and ambiguous concepts stored in ConceptNet. NASARI and our approach obtained comparable results above the rest of techniques in this experiment. In the last experiment, we evaluated how the performance of general-purpose state-of-the-art word sense disambiguation techniques improves by including relations disambiguated by these specific disambiguation techniques. This evaluation task showed how including the relations obtained using our approach led to the best word sense disambiguation performance.

The article associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. Disambiguation of semantic relations using evidence aggregation according to a sense inventory. *Submitted to IEEE Transactions on Knowledge and Data Engineering (ISSN: 1041-4347).*

4.2.3 An automorphic distance metric for node role discovery

Node role discovery is the problem of dividing the nodes of a network into groups of nodes exhibiting similar connectivity patterns [8]. This is a problem of interest in the context of the link prediction problem because the role of nodes can be an important element driving the dynamics of link formation in real-world networks. Despite different definitions of role exist, it can be defined in terms of automorphic equivalence, where two nodes fall in the same class if the graph would remain the same if their labels were swapped. However, the definition of role in terms of automorphic equivalence is too restrictive and will rarely be satisfied in complex networks. In order to overcome this limitation, we proposed a metric capturing the concept of automorphic distance between a pair of nodes. This proposed metric captures, in terms of distance, how close or how far of being automorphically equivalent two nodes are.

Our distance metric is built on top of the Weisfeiler-Lehman algorithm for graph isomorphism testing [64]. In summary, the Weisfeiler-Lehman algorithm is an iterative procedure consisting of, starting from a labelling of the nodes corresponding to their degree, performing alternative steps where the labels of the neighbors are gathered, sorted, and hashed, producing new labels for the next iteration. This procedure is repeated until the sets of nodes that would be obtained by grouping nodes according to their labels in one iteration are equivalent to the set of nodes that would be obtained in the next iteration. The labels obtained after convergence are known as canonical labels and, if two graphs share the same canonical labels, they are isomorphic.

The proposed automorphic distance metric for nodes consists in measuring the distance between their canonical labels obtained using the Weisfeiler-Lehman algorithm. The distance between the labels of a pair of nodes can be used as a measure of how close are these two nodes of playing the same topological role in the network. We proposed building these distances iteratively. The Weisfeiler-Lehman algorithm begins by assigning a label to each node, such that two nodes only have the same label if and only if both nodes have the same degree. We defined the distance between labels in this first assignation as the absolute difference between the degree of the nodes associated to each label.

Given these initial distances, the distances for the labels in the subsequent iterations are calculated as the sum of the distance of the neighbor labels from previous iteration. In order to compute this summation, labels are paired to most similar ones by solving the assignment problem using the Hungarian algorithm [65].

We proved that our proposal is a valid distance metric by showing how it satisfies the required conditions, which are non-negativity, identity of indiscernibles, symmetry, and triangle inequality.

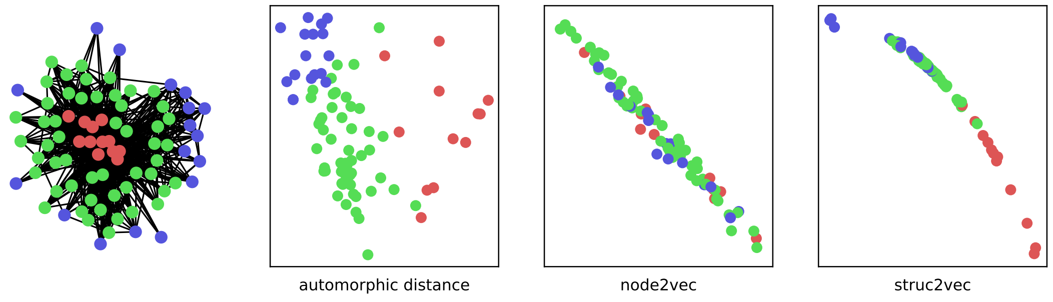


Figure 6: Node embeddings for a world trade network computed using different techniques: the automorphic distance proposed in this thesis, node2vec, and struc2vec.

Different experiments were performed in order to evaluate the performance of our distance metric for the node role discovery problem. We used the obtained distances between nodes to compute node embeddings using multidimensional scaling (MDS, [66]). We plotted the embeddings for the nodes in the Zachary's karate club network, representing the friendship network in a karate club after a dispute between the two leaders of the club. Plots showed how our approach can linearly separate the different node classes better than other approaches. We also tested our approach in a world trade network, as shown in Figure 6. Our results show how our method separates the different categories of countries better than alternative approaches.

The technical report associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. An automorphic distance metric and its application to node embedding for role mining. ArXiv e-prints 1712.06979, December 2017.

4.3 A network data mining framework

One of the research lines of this thesis was the development of software tools for network data mining, specially link prediction techniques, under the NOESIS project.

NOESIS[1], which stands for Network-Oriented Exploration, Simulation, and Induction System, is an open source framework for network data mining containing a large number of network analysis algorithms, including metrics for evaluating network structural properties, community detection techniques, and network layouts, among others. This framework, oriented towards high efficiency and scalability by using parallelization, is the result of the work of different developers. The main contribution in the context of this thesis is the implementation of a large number of link prediction techniques in the framework and the development of an API that allows to use NOESIS from Python[2].

NOESIS provides a friendly interface for a large number of network analysis algorithms. For example, the following Java code snippet, where *boilerplate code* has been omitted, shows how to create a network using the Barabási–Albert model and perform link prediction using preferential attachment:

```
Network network = new BarabasiAlbertNetwork(100, 10);
LinkPredictionScore method = new PreferentialAttachmentScore(network);
Matrix result = method.call();
```

The same example implemented using the Python API is shown in the following code fragment:

```
ns = Noesis()
network = ns.create_network_from_model('BarabasiAlbert', 100, 10)
predictor = ns.create_link_predictor('PreferentialAttachment')
result = predictor.compute(network)
ns.end()
```

In order to divulge our work, we wrote a paper explaining the NOESIS framework design and the collection of techniques implemented. The whole NOESIS ecosystem is implemented on top of the hardware abstraction layer, which provides routines for high performance parallel computation while abstracting the underlying computational complexity with popular paradigms like the map-reduce pattern [67]. On top of this layer, the reflective kernel and the data access layer were implemented. In the one hand, the kernel layer provides the main NOESIS base models and tasks as data structures and algorithms, respectively. This layer also implements the corresponding meta-objects, which can be dynamically manipulated at runtime. On the other hand, the data access layer provides an unified interface for accessing heterogeneous external data sources. This part of the system provides the I/O interfaces to read and write data in different formats, such as several standard network storage file formats. The NOESIS API relies on top of these layers, providing the interfaces that developers must use to exploit the different features of the framework. Additionally, an application generator module provides a graphical user interface for using NOESIS algorithms without the requirement of programming skills.

Since in our manuscript we introduced the different implemented techniques, our paper also serves as a general review of the network analysis field. For example, we

[1]`http://noesis.ikor.org`
[2]`https://github.com/fvictor/noesis-python`

described some of the most relevant theoretical network formation models, commonly used to study how real-world networks are formed and their underlying dynamics, including random models like Watts-Strogatz model or Barabási-Albert model. We also summarized several metrics for network structural properties, including centrality, reachability, or betweenness, among other groups of metrics. Different network visualization techniques computing pleasant layouts for network nodes are described in our work, including force-based, hierarchical, and regular layouts. An important part of our work is the categorization of the implemented community detection techniques. These techniques are categorized in our work as: hierarchical, which build a hierarchy of communities using agglomerative or divisive approaches; modularity-based, which compute partitions based on maximizing their modularity; partitional, which perform a partition of the network and relocate nodes iteratively; spectral, which work over the Laplacian representation of the network; and overlapping, which are able to compute overlapping communities where nodes may belong to different clusters. Other contribution of our work is the computational complexity analysis of the described community detection techniques. Finally, we also describe the implemented link prediction and scoring techniques and provide their computational complexity, inspired by the categorization that we proposed in our survey about link prediction.

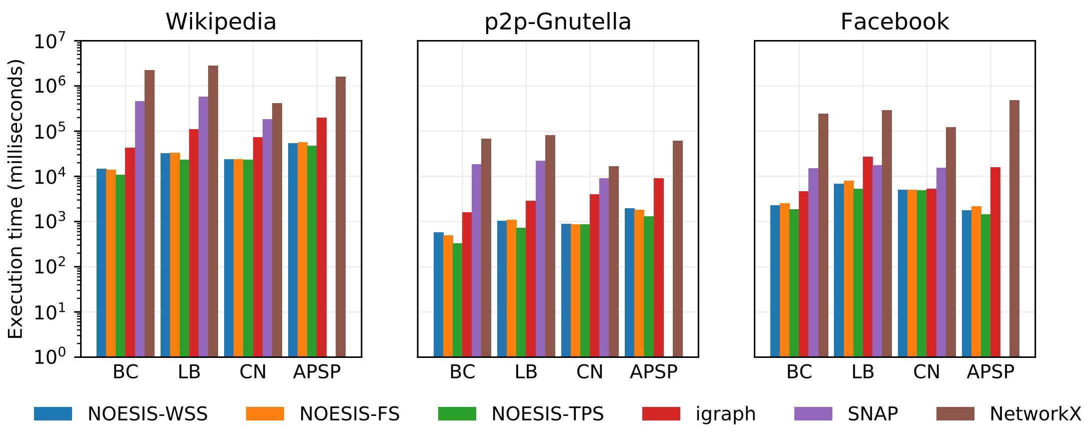


Figure 7: Execution times in milliseconds, represented on a logarithmic scale, of different network analysis frameworks for four different tasks applied to three complex networks.

Finally, we carried out different experiments in order to compare the performance of NOESIS with the performance of three popular network analysis frameworks: igraph [68], SNAP [69], and NetworkX [70]. We considered three different task schedulers for NOESIS: work stealing scheduler (WSS), future scheduler (FS), and thread pool scheduler (TPS). Four different tasks were considered: betweenness centrality (BC), link betweenness (LB), closeness (CN), and all shortest paths from every node using Dijkstra's algorithm (APSP). The APSP could not be carried out using SNAP due to lack of support. The execution times considering three different large complex networks are shown in Figure 7. A logarithmic scale was used because of the difference in orders of magnitude in performance of our proposal in contrast with the compared approaches.

The article associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. The NOESIS network-oriented exploration, simulation, and induction system. ArXiv e-prints 1611.04810, June 2017. *Submitted to Complexity (ISSN: 1099-0526).*

5 Concluding remarks

Different research lines were followed in this thesis with a common theme: link prediction in complex networks. The main lines of research in this thesis were the proposal of novel generic link prediction approaches, the development of software for network data mining, and the analysis of novel applications for link prediction techniques.

In the first line of research, our goal was the exhaustive study and understanding of the state of the art in link prediction problem, including novel techniques and approaches. This study led to the publication of a survey paper in ACM Computing Surveys, the proposal of a taxonomy to categorize link prediction techniques, and a complete empirical evaluation of these techniques using different complex network from diverse real-world domains. As a result of the acquired knowledge, we proposed two novel link prediction techniques. The first was based on the proposal of a generalization of degree penalization similarity-based link prediction techniques and the observation that the optimal parameters for this penalization are strongly correlated to measurable network topological features. This approach consistently outperformed the reference link prediction techniques in the evaluated prediction tasks without an increase in the needed computational resources. The success of this approach suggested the importance of considering adaptive techniques instead of relying on fixed definitions of similarity. The second proposed technique was a local probabilistic technique where different evidences are aggregated by assuming their conditional independence. After testing this novel approach using a simple feature as node degree, results showed how our technique achieved superior performance compared to the reference techniques while maintaining the high scalability of local link prediction techniques.

The second line of research comprised an important fraction of the contributions made in this thesis. In this line, we focused on developing novel applications of link prediction techniques to solve problems from very different domains. In the context of bioinformatics, we developed a novel prioritization technique through the integration of heterogeneous data, which outperformed two state-of-the-art approaches for disease-gene prioritization and disease-domain prioritization. In the context of natural language processing, we tackled the problem of disambiguating semantic relations between concepts as a link prediction problem. Our probabilistic approach gathers redundant evidence to disambiguate the semantic meaning of words in relations extracted from knowledge bases without using labelled data. Finally, we developed a novel approach for node role discovery based on the Weisfeiler-Lehman test for graph isomorphism, with applications to different network-related problems, including link prediction.

The third line of research of this thesis was dedicated to the development of software for high performance link prediction and general complex network analysis. We worked in the NOESIS framework to implement a collection of state-of-the-art link prediction techniques and develop an API that allowed to use NOESIS functionality from Python. We also worked in a technical report in order to introduce NOESIS to the scientific community, which also serves as a general introduction to the field of network analysis as result of describing the most relevant network mining-related algorithms.

6 Future work

Some promising research lines arise from the conclusions drawn from this thesis. Different ideas introduced in this dissertation can be used as the basis for the development of new link prediction techniques and new applications in different fields. Some interesting lines of future work are proposed below:

- **Novel adaptive and scalable link prediction techniques:** Our proposed link prediction techniques, both the adaptive and the probabilistic, aim to maintain a low computational complexity to preserve their high scalability while obtaining improved performance as a result of adapting to the network under study. Most current learning-based link prediction techniques consider the whole network in the learning process, limiting their applicability in real-world applications where networks can be comprised of millions of nodes. We plan to develop new link prediction techniques that can adapt to the network based on local or efficiently computable features.

- **Novel applications of prioritization techniques.** In this thesis, we proposed a technique that can exploit heterogeneous data to carry out different types of prioritization, with applications in the context of bioinformatics. Our results showed how the integration of data can improve the performance of these techniques. However, our experimentation is still limited as we only considered three networks in our seminal study. The study of the integration of more biological and medical data is a prominent line of future research to consider.

- **New techniques for the disambiguation of semantic relations.** Our work shows how redundancy in natural language as a result of synonymy, hyponymy, and hypernymy can be used to disambiguate semantic relations using approaches inspired by link prediction techniques, and how these disambiguated relations can be used to improve the results obtained by state-of-the-art word disambiguation techniques. An important point to improve in our proposal is its scalability, specially in the context of big data. We intend to work in a variant of our algorithm using distributed representations, which should lead to more scalable alternatives to address this problem in massive datasets.

- **Further development of the NOESIS framework:** NOESIS is currently in a mature state of development. Our results have shown how NOESIS not only includes a larger collection of network analysis techniques than popular frameworks, but it is also more efficient than well-established alternatives. A future line of work is to maintain the NOESIS framework updated to the ever evolving field of network data mining.

Bibliography

[1] Douglas B. West. *Introduction to Graph Theory.* Prentice Hall, 2nd edition, September 2000.

[2] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107, 2014.

[3] Stephen P. Borgatti, Ajay Mehra, Daniel J. Brass, and Giuseppe Labianca. Network analysis in the social sciences. *Science*, 323(5916):892–895, 2009.

[4] Roger Guimera, Stefano Mossa, Adrian Turtschi, and L.A. Nunes Amaral. The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles. *Proceedings of the National Academy of Sciences*, 102(22):7794–7799, 2005.

[5] Jean-François Rual, Kavitha Venkatesan, Tong Hao, Tomoko Hirozane-Kishikawa, Amélie Dricot, Ning Li, Gabriel F. Berriz, Francis D. Gibbons, Matija Dreze, Nono Ayivi-Guedehoussou, et al. Towards a proteome-scale map of the human protein–protein interaction network. *Nature*, 437(7062):1173, 2005.

[6] Mark E. J. Newman. The structure and function of complex networks. *Society for Industrial and Applied Mathematics Review*, 45(2):167–256, 2003.

[7] Giuliano Andrea Pagani and Marco Aiello. The power grid as a complex network: A survey. *Physica A: Statistical Mechanics and its Applications*, 392(11):2688–2700, 2013.

[8] Ryan A. Rossi and Nesreen K. Ahmed. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1112–1131, 2015.

[9] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1):5, 2007.

[10] Ryan A. Rossi, Brian Gallagher, Jennifer Neville, and Keith Henderson. Modeling dynamic behavior in large evolving graphs. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 667–676. ACM, 2013.

[11] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[12] Lei Tang, Huan Liu, Jianping Zhang, and Zohreh Nazeri. Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 677–685. ACM, 2008.

[13] Roberto Tamassia. *Handbook of graph drawing and visualization.* CRC press, 2013.

[14] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[15] David Liben-Nowell and Jon M. Kleinberg. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.

[16] Ilham Esslimani, Armelle Brun, and Anne Boyer. Densifying a behavioral recommender system by social networks link prediction methods. *Social Network Analysis and Mining*, 1(3):159–172, 2011.

[17] Anita Yadav, Yatindra Nath Singh, and R. R. Singh. Improving routing performance in AODV with link prediction in mobile adhoc networks. *Wireless Personal Communications*, 83(1):603–618, 2015.

[18] Zan Huang and Daniel Dajun Zeng. A link prediction approach to anomalous email detection. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Taipei, Taiwan, October 8-11, 2006*, pages 1131–1136, 2006.

[19] Francesco Folino and Clara Pizzuti. Link prediction approaches for disease networks. In *Information Technology in Bio- and Medical Informatics - Third International Conference, ITBAM 2012, Vienna, Austria, September 4-5, 2012. Proceedings*, pages 99–108, 2012.

[20] U. Martin Singh-Blom, Nagarajan Natarajan, Ambuj Tewari, John O Woods, Inderjit S. Dhillon, and Edward M. Marcotte. Prediction and validation of gene-disease associations using methods inspired by social network analyses. *PloS one*, 8(5):e58977, 2013.

[21] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011.

[22] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: The topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1-4):69–77, 2000.

[23] Miller McPherson, Lynn Smith-Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[24] Mark E.J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

[25] James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

[26] Mohammad Al Hasan and Mohammed J. Zaki. A survey of link prediction in social networks. In *Social Network Data Analytics*, pages 243–275. 2011.

[27] Peng Wang, Baowen Xu, Yurong Wu, and Xiaoyu Zhou. Link prediction in social networks: The state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[28] Mark E.J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, 2001.

[29] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[30] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

[31] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[32] Erzsébet Ravasz, Anna Lisa Somera, Dale A. Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, 2002.

[33] David Liben-Nowell. *An algorithmic approach to social networks*. PhD thesis, Massachusetts Institute of Technology, 2005.

[34] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[35] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 613–622, 2006.

[36] Oron Vanunu and Roded Sharan. A propagation–based algorithm for inferring gene–disease assocations. In *German Conference on Bioinformatics*, pages 54–52, 2008.

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

[38] Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *Europhysics Letters*, 89(5):58007, 2010.

[39] Aaron Clauset, Cristopher Moore, and Mark E.J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98, 2008.

[40] Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.

[41] Zan Huang. Link prediction based on graph topology: The predictive value of generalized clustering coefficient. In *Workshop on Link Analysis: Dynamics and Static of Large Networks*, 2006.

[42] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 322–331. IEEE, 2007.

[43] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *Workshop on link analysis, counter-terrorism and security*, 2006.

[44] Catherine A. Bliss, Morgan R. Frank, Christopher M. Danforth, and Peter Sheridan Dodds. An evolutionary algorithm approach to link prediction in dynamic social networks. *Journal of Computational Science*, 5(5):750–764, 2014.

[45] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2011.

[46] Jérôme Kunegis and Andreas Lommatzsch. Learning spectral graph transformations for link prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 561–568. ACM, 2009.

[47] Harry Zhang. The optimality of Naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, pages 562–567, 2004.

[48] Zhen Liu, Qian-Ming Zhang, Linyuan Lü, and Tao Zhou. Link prediction in complex networks: A local Naïve Bayes model. *Europhysics Letters*, 96(4):48007, 2011.

[49] Stein Aerts, Diether Lambrechts, Sunit Maity, Peter Van Loo, Bert Coessens, Frederik De Smet, Leon-Charles Tranchevent, Bart De Moor, Peter Marynen, Bassem Hassan, et al. Gene prioritization through genomic data fusion. *Nature Biotechnology*, 24(5):537, 2006.

[50] Stephen Oliver. Proteomics: Guilt-by-association goes global. *Nature*, 403(6770):601, 2000.

[51] Marc A. Van Driel, Jorn Bruggeman, Gert Vriend, Han G. Brunner, and Jack A.M. Leunissen. A text-mining analysis of the human phenome. *European Journal of Human Genetics*, 14(5):535, 2006.

[52] Suraj Peri, J. Daniel Navarro, Ramars Amanchy, Troels Z. Kristiansen, Chandra Kiran Jonnalagadda, Vineeth Surendranath, Vidya Niranjan, Babylakshmi Muthusamy, T.K.B. Gandhi, Mads Gronborg, et al. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Research*, 13(10):2363–2371, 2003.

[53] Balaji Raghavachari, Asba Tasneem, Teresa M. Przytycka, and Raja Jothi. Domine: A database of protein domain interactions. *Nucleic Acids Research*, 36(S1):D656–D661, 2007.

[54] See-Kiong Ng, Zhuo Zhang, Soon-Heng Tan, and Kui Lin. InterDom: a database of putative interacting protein domains for validating predicted protein interactions and complexes. *Nucleic Acids Research*, 31(1):251–254, 2003.

[55] Robert D. Finn, Jaina Mistry, Benjamin Schuster-Böckler, Sam Griffiths-Jones, Volker Hollich, Timo Lassmann, Simon Moxon, Mhairi Marshall, Ajay Khanna, Richard Durbin, et al. Pfam: Clans, web tools and services. *Nucleic Acids Research*, 34(S1):D247–D251, 2006.

[56] Eric Jain, Amos Bairoch, Severine Duvaud, Isabelle Phan, Nicole Redaschi, Baris E. Suzek, Maria J. Martin, Peter McGarvey, and Elisabeth Gasteiger. Infrastructure for the life sciences: design and implementation of the uniprot website. *BMC Bioinformatics*, 10(1):136, 2009.

[57] TaeHyun Hwang, Wei Zhang, Maoqiang Xie, Jinfeng Liu, and Rui Kuang. Inferring disease and gene set associations with rank coherence in networks. *Bioinformatics*, 27(19):2692–2699, 2011.

[58] Wangshu Zhang, Yong Chen, Fengzhu Sun, and Rui Jiang. DomainRBF: a Bayesian regression approach to the prioritization of candidate domains for complex diseases. *BMC Systems Biology*, 5:55, 2011.

[59] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10, 2009.

[60] Douglas B. Lenat, Mayank Prakash, and Mary Shepherd. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine*, 6(4):65, 1985.

[61] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[62] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. NASARI: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64, 2016.

[63] Hugo Liu and Push Singh. ConceptNet—a practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):211–226, 2004.

[64] Boris Weisfeiler and A.A. Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

[65] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics*, 2(1-2):83–97, 1955.

[66] Ingwer Borg and Patrick J.F. Groenen. *Modern Multidimensional Scaling: Theory and applications*. Springer Science & Business Media, 2005.

[67] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[68] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006.

[69] Jure Leskovec and Rok Sosič. SNAP: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology*, 8(1):1, 2016.

[70] Daniel A. Schult. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference*. Citeseer, 2008.

Chapter II

Publications: published, accepted, and submitted papers

This chapter collects the scientific publications result of this dissertation. These publications are organized into three categories: publications related to the study of the link prediction problem, publications about applications of link prediction, and publications associated to the software developed during this thesis.

1 Link prediction

This Section contains the publications related to the formal study of the link prediction problem, including a survey of link prediction techniques, a journal paper proposing an adaptive link prediction technique, and a book chapter about the probabilistic link prediction approach.

1.1 Link prediction: The state of the art

The journal paper associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. A survey of link prediction in complex networks. ACM Computing Surveys 49(4):69:1-69:33, 2017. DOI 10.1145/3012704.

- Status: **Published**
- ISSN: 0360-0300
- Impact Factor (JCR 2016): 6.748
- Subject Category:
  - Computer Science, Theory & Methods. Ranking 2/104.

A Survey of Link Prediction in Complex Networks

Víctor Martínez   Fernando Berzal   Juan-Carlos Cubero

Abstract

Networks have become increasingly important to model complex systems comprised of interacting elements. Network data mining has a large number of applications in many disciplines including protein-protein interaction networks, social networks, transportation networks, and telecommunication networks. Different empirical studies have shown that it is possible to predict new relationships between elements attending to the topology of the network and the properties of its elements. The problem of predicting new relationships in networks is called link prediction. Link prediction aims to infer the behavior of the network link formation process by predicting missed or future relationships based on currently observed connections. It has become an attractive area of study since it allows us to predict how networks will evolve. In this survey we will review the general-purpose techniques at the heart of the link prediction problem, which can be complemented by domain-specific heuristic methods in practice.

1 Introduction

A link is a connection between two nodes in a network. This simple concept can be used to represent extremely complex systems where a large number of elements interact among them. The proliferation of data that can be represented as networks has created new opportunities but also new challenges in the field of data mining. A large number of problems related to network mining are currently being studied, including community detection [1], structural network analysis [2], or network visualization [3]. One of the most interesting network-related problems is link prediction, which consists of inferring the existence of new relationships or still unknown interactions between pairs of entities based on their properties and the currently observed links [4]. The approaches and techniques designed to solve this problem enable the extraction of implicit information present in the network, and the identification of spurious links, as well as modeling and evaluating network evolution mechanisms.

Link prediction methods have been successfully applied to biological networks in order to predict previously unknown interactions between proteins [5], significantly reducing the costs of empirical approaches [6]. They have also been used to model highly dynamic systems, such as email or telephone call networks [7]. Link prediction techniques are widely present in our daily lives, suggesting people we may know but we are not still connected to in our social networks [4] or products we could be interested in electronic commerce [8].

Networks have been extensively studied since the proposition of the first basic models to identify the laws that drive network formation and lead to their structural features [9]. Some techniques that could be considered as link prediction methods were then proposed. However, it was not until specific link-prediction-oriented seminal works [10], which performed a comprehensive analysis of the problem, when this field came under the spotlight due to its applicability and usefulness in a great variety of contexts.

Link prediction is grounded in the empirical evidence that two entities are more likely to interact if they are similar. Similarity in networks must be understood as an abstract concept and could vary between networks. Understanding the domain that the network represents is a crucial step to define the similarity between two nodes. In most domains, it has been observed that nodes tend to form highly connected communities [11]. This has led to the common definition of similarity as the amount of relevant direct or indirect paths between nodes.

One of the main difficulties in link prediction is achieving a good balance between the amount of information considered to perform the prediction and the algorithm complexity of the techniques needed to collect that information. Since actual networks are usually formed by hundreds of thousands or even millions of nodes, the techniques used to perform link prediction must be highly efficient. However, considering only local information could lead to poor predictions, especially in very sparse networks.

Different reviews about this topic have been previously published and have influenced this work [4, 12, 13, 14]. However, new approaches have been developed since their publication and a new review of the state of the art is desirable. A large number of domain-specific link prediction techniques have been proposed. However, these techniques are left out of this review since most of them are tuned variations of the basic methods that will be described below. In this work, we focus on link prediction techniques in undirected networks using derived topological features. These techniques are more versatile than attribute-based methods since topology-based techniques are not domain-specific.

We make several contributions in this survey. First, we perform a detailed and thorough study of the state-of-the-art of link prediction approaches and methods using a unified notation. This study includes the computational complexity analysis of most important techniques, which is not always provided by their original authors. Second, we propose a taxonomy to classify link prediction techniques attending to the methodology that they employ and the amount of information they use. Finally, we perform an empirical study of the techniques by applying the most important methods to a set of networks with different properties and evaluating their results.

2 The link prediction problem

The link prediction problem in undirected networks is defined as follows. Given a snapshot of an undirected network in time t where each node represents an entity or actor and each link represents an interaction or relationship between the pair of entities connected through the link, the link prediction problem can be formally defined as inferring the subset of missing links (existing but non-observed links) in the current snapshot or that will be formed in time $t + \Delta$.

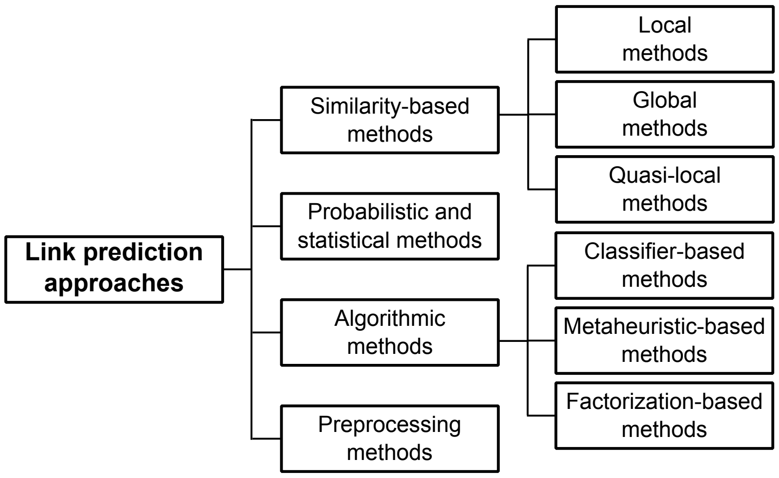


Figure 1: Our proposed taxonomy for link prediction techniques.

Most existing link prediction techniques consider it a ranking problem where pairs of unconnected nodes are given a score proportional to the likelihood of existence of a link between them. A threshold is usually established: all pairs with a score above the threshold are considered as positive instances and all pairs below the threshold are viewed as negative instances. This threshold can be specified by the user, application dependent or automatically determined. As far as we know, automatic threshold selection in link prediction remains an unexplored problem. The link prediction problem can be viewed as a binary classification problem for links in the network where two classes are considered: positive or existence of link and negative or absence of link.

A large number of link prediction techniques have been proposed in the specialized literature. These techniques differ in different aspects, including the evolution rules that they model, their computational complexity, or the type or amount of information they consider. We propose a custom extended version (see Figure 1) of the taxonomy presented by [13]. These taxonomies classify the previously proposed methods attending to the approach they follow and the amount of information that they take into account. Each method is described in detail below.

2.1 Applications of Link Prediction

Link prediction techniques have found a large number of applications in very different fields. Any domain where entities interact in a structured way can potentially benefit from link prediction. Some interesting or widely-used applications of link prediction are described below.

Link prediction techniques are used to improve similar users selection in recommender systems that follow a collaborative approach, leading to better recommendation results [15]. A similar application is related to social networks, which have become extremely popular in the modern society. The users of these systems expect to have simple and effective mechanisms to find their acquaintances among the massive amount of users registered. Most social networks are using link prediction techniques to automatically suggest acquaintances with a high degree of accuracy.

In the field of biology, link prediction techniques are being applied to find possible interactions between pairs of proteins in protein-protein interaction network (PPI network) [16]. In vitro experiments to determine which proteins interact are expensive in money and time, so studied targets are carefully selected when there is a prior evidence, which could be obtained computationally.

Other application is found in collaboration prediction in scientific co-authorship networks. Collaboration data is easily accessible, since some journal indexing sites make public their collections. Link prediction methods has become a tool to better understand how some research fields will evolve by predicting which authors or groups could potentially collaborate in the future [17].

Entity resolution, also known as record linkage or deduplication, consists of finding duplicated references or records in a dataset. Traditionally, entity resolution only relied in attribute similarity between entries. However, recently, some authors have shown that considering context information in network-structured domains using link prediction techniques to take into consideration the similarity between the instances can lead to improvements in entity resolution [18].

Social network analysis has been widely used to analyse the structure of criminal and terrorist networks in order to fight against organized crime [19]. For example, [20] has shown that the topology of some criminal networks does not change if a significant fraction of links are reinserted using link prediction techniques. These results suggest that link prediction can reveal actual links in criminal networks, allowing us to anticipate certain criminal actions.

Finally, networks can be used to analyze how tendencies spread across the society. Network analysis can be used to improve marketing studies. Some authors have shown how link prediction can be used in viral marketing in order to achieve better marketing plans [21].

2.2 Terminology and Notation

A graph or network G is an ordered pair $G = (V, E)$ where V is a set of optionally-labeled vertices or nodes and E is a set of links between pairs of elements from the set V. A link between two nodes x and y is noted as $e_{x,y}$. The number of nodes in the network, also known as the size of the network, is denoted as $|V|$. The number of links is denoted as $|E|$. We can distinguish between directed links (noted as arcs), which connect a source node to a destination node, and undirected links (noted as edges), when there is no concept of source and destination. A directed graph is composed only of arcs. Similarly, an undirected graph is composed only of edges. Finally, a mixed graph can contain both types of links (arcs and edges).

The set of nodes connected through an edge to a node $x \in V$ is called the neighborhood of x and is denoted as Γ_x. In undirected graphs, the degree of a node x is defined as the number of edges connected to the node and will be denoted as $|\Gamma_x|$. In directed graphs, the degree of a node is the sum of the out-degree and the in-degree, which are the count of outgoing arcs and incoming arcs, respectively. The average degree of a network is denoted as $\langle\Gamma\rangle$ and is equal to the average degree of all its nodes.

Let a loop be an edge or arc connecting a node to itself. A simple graph is defined as a graph without loops and with no more than one edge or arc between each pair of vertices. The techniques reviewed in this survey assume there are no loops in the network.

A path is a sequence of links that connect a sequence of nodes in the graph. In directed networks, path steps are restricted to move from the source node to the destination node of the same arc. The path length is the number of links in the path. The shortest path between two vertices is the path with the smaller length between those vertices. Multiple shortest paths for a pair of vertices can coexist. A graph is called connected if there is a path between each pair of nodes $x, y \in V$. If the graph is not connected, it is composed of components. A component is a connected subgraph. A connected graph has only one component. If one of the components has a significantly larger number of nodes compared to the other components, it is usually called the main or giant component.

3 Similarity-based Methods

Similarity-based methods assume that nodes tend to form links with other similar nodes. These methods stem from the hypothesis that two nodes are similar if they are connected to similar nodes or are near in the network according to a given distance function. These approaches define a function $s(x, y)$ that assigns a score known as similarity for every pair of nodes x and y. This measure is computed for each interesting pair of nodes, usually those with non-observed links between them. Pairs of nodes are ranked in decreasing order based on their similarity scores, so links at the top of the ranking are supposed to be more likely to be present in the set of missing links.

The definition of similarity is not a trivial task, since it has a heuristic component. The similarity function can vary between networks even from the same domain. As a non-surprising result, a large number of similarity-based methods with different definitions of similarity have been proposed. It has been empirically shown that the similarity between nodes can be defined in terms of network topological properties.

As an additional contribution of this survey, the algorithmic complexity is also computed for every similarity-based method using the big O notation. Three variables have been considered to measure problem size: v as the number of nodes, e as the number of edges, and k as the maximum degree of a node. Some simple optimizations will be taken into account in our algorithmic complexity analysis.

For example, the intersection between two sets of size n and m, respectively, can be computed with complexity $O(n + m)$ using hash tables. Insertion and lookup time complexities are $O(1)$ using hashing. A set is iterated to fill a hash table. The second set is iterated to check whether its members already are in the hash table. Collisions can be totally avoided since each node has an unique assigned number. The set union operation has the same time complexity.

Matrix operations are also considered. Matrix addition and subtraction are $O(n^2)$. However, in sparse networks they can be computed with complexity $O(nt)$ where $t << n$. The inversion of a matrix of size $n \times n$ is of cubic time complexity $O(n^3)$ for the naive algorithm. Some algorithms have been proposed to improve this efficiency but they are always worse than quadratic. The multiplication of two matrices of size $n \times m$ and $m \times p$ respectively has a complexity $O(mnp)$ in dense matrices. However, in sparse matrices, this efficiency can be reduced to $O(mtp)$ where $t << n$. It should be noted that the adjacency matrices of real-world networks are typically sparse.

3.1 Local Approaches

Local similarity-based approaches use node neighborhood-related structural information to compute the similarity of each node with other nodes in the network. These approaches are faster than non-local techniques and highly parallelizable. In addition, they allow us to handle efficiently the link prediction problem in very dynamic and changing networks such as online social networks. Their main drawback is that using only local information restricts the set of nodes similarity can be computed for to distance-two nodes (neighbors of neighbors). This can be a big drawback as many links are formed at distances greater than two in many real-world networks, specially in non small-world networks [4]. However, these methods have shown a very competitive prediction accuracy against more complex techniques. It should be noted that, since these methods are limited to two-hop nodes, their time complexity is $O(vk^2 f(k))$ where $f(k)$ is the complexity of computing the similarity between a pair of nodes and their spatial complexity is $O(vk^2)$.

3.1.1 Common neighbors (CN)

Common neighbors is the simplest local technique. The similarity between two nodes is defined as the number of shared neighbors between both nodes [4]. It makes sense to assume that, if two individuals share many acquaintances, they are more likely to meet than two individuals without common contacts. Different studies have confirmed this hypothesis by observing a correlation between the number of shared neighbors between pairs of nodes and the probability of being linked [22]. This method defines the similarity function as

$$s(x, y) = |\Gamma_x \cap \Gamma_y|$$

Despite its simplicity, this measure performs surprisingly well on most real-world networks and beats very complex approaches. This method is the basis for other approaches presented below. Using this method to compute similarity for all possible pairs results in a local link prediction technique with $O(vk^2(k+k))) = O(vk^3)$ time complexity.

3.1.2 The Adamic-Adar index (AA)

This similarity measure, initially proposed by Lada Adamic and Eytan Adar, was intended to measure the similarity between two entities based on their shared features [23]. However, each feature weight is logarithmically penalized by its appearance frequency. If we take neighbors as features, it can be written as

$$s(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{\log |\Gamma_z|}$$

This equation is a variation of the common neighbors similarity function where each shared neighbor is penalized by its degree. This intuitively makes sense in a large number of real-world networks. For example, in social networks, the amount of resources or time that a node can spend on each of its neighbors decreases as its degree increases, also decreasing its influence on them. Its computational complexity is, again, $O(vk^3)$.

3.1.3 The resource allocation index (RA)

This index is motivated by the resource allocation process that takes place in complex networks [24]. It models the transmission of units of resources between two unconnected nodes x and y through neighborhood nodes. Each neighborhood node gets a unit of resource from x and equally distributes it to its neighbors. The amount of resources obtained by node y can be considered as the similarity between both nodes. This similarity function is formulated as

$$s(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{|\Gamma_z|}$$

It should be noted that this measure is strongly related to the common neighbors and the Adamic-Adar index [25]. The resource allocation index has shown to be the local measure that achieves better results in a large number of networks. Some recent works have led to the conclusion that the performance of these metrics increases by making the penalization for high degree nodes more pronounced in scale-free networks [26]. This method also has a $O(vk^3)$ time complexity.

3.1.4 Resource allocation based on common neighbor interactions (RA-CNI)

Resource allocation based on common neighbor interactions is motivated by the resource allocation process where each node sends a unit of resource to its neighbors [27]. However, this method also takes into consideration the return of resources in the opposite direction. It is defined as

$$s(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{|\Gamma_z|} + \sum_{e_{i,j} \in E, |\Gamma_i| < |\Gamma_j|, i \in \Gamma_x, j \in \Gamma_y} \left(\frac{1}{|\Gamma_i|} - \frac{1}{|\Gamma_j|} \right)$$

Experimental results obtained by its original authors show that this method achieves a better precision than the original local resource allocation score in many real-world networks. The computational complexity of this method is, however, $O(vk^4)$ in the worst case, worse than the aforementioned methods.

3.1.5 The preferential attachment index (PA)

This index is a direct result of the well-known Barabási-Albert complex network formation model [28, 29]. Many real network node degrees follow a power law distribution resulting in scale-free networks that could not be explained by previous network formation models. Albert-László Barabási and Réka Albert built a theoretical model based on the observation that the probability of link formation between two nodes increases as the degree of these nodes does. This formation model leads to the concept of "the rich get richer", which generates the power law degree distribution observed in scale-free networks. The similarity between two nodes, according to the Barabási-Albert model, can be estimated as

$$s(x,y) = |\Gamma_x||\Gamma_y|$$

This measure can be also applied in non-local contexts, since it does not rely on shared neighbors. However, its prediction accuracy is usually poor when applied as a global measure. The computational complexity of this method is $O(vk^2)$, faster than the methods based on shared neighbors.

3.1.6 The Jaccard index (JA)

This widely-used coefficient in information retrieval systems was proposed by Paul Jaccard (1868-1944) to compare the similarity and diversity of sample sets [30]. It measures the ratio of shared neighbors in the complete set of neighbors for two nodes. This similarity function is defined as

$$s(x,y) = \frac{|\Gamma_x \cap \Gamma_y|}{|\Gamma_x \cup \Gamma_y|}$$

It can be easily seen that this method is yet another variation of the common neighbors method where there is a penalization for each non-shared neighbor. The algorithmical time complexity of this method is $O(vk^2(2k + 2k)) = O(vk^3)$.

3.1.7 The Salton index (SA)

This index is also known as the cosine similarity [31]. This measure is closely related to the Jaccard index and some works have shown that, in most practical situations, the Salton index yields a value that is approximately twice the Jaccard index [32]. This similarity function is defined as

$$s(x, y) = \frac{|\Gamma_x \cap \Gamma_y|}{\sqrt{|\Gamma_x||\Gamma_y|}}$$

The computational time complexity of this method is, again, $O(vk^3)$.

3.1.8 The Sørensen index (SO)

This index was developed by the botanist Thorvald Sørensen in 1948 to compare the similarity between different ecological community data samples [33]. Despite its similarity with the Jaccard index, it is less sensitive to outliers [34]. Sørensen similarity is defined as

$$s(x, y) = \frac{2|\Gamma_x \cap \Gamma_y|}{|\Gamma_x| + |\Gamma_y|}$$

The algorithmic time complexity of this method for all possible distance-two pairs is $O(vk^3)$.

3.1.9 The hub promoted index (HPI)

This index was proposed as a result of studying modularity in metabolic networks [35]. These networks show a hierarchical structure with small highly internally-connected modules that are also highly isolated from each other. The main goal of this similarity measure is to avoid link formation between hub nodes and promote link formation between low degree nodes and hubs. This index defines similarity as

$$s(x, y) = \frac{|\Gamma_x \cap \Gamma_y|}{\min(|\Gamma_x|, |\Gamma_y|)}$$

The computation complexity of this method is, once more, $O(vk^3)$.

3.1.10 The hub depressed index (HDI)

This index is based on the hub promoted index but has an opposite goal [35]. The hub depressed index promotes link formation between hubs and between low degree nodes, but not between hubs and low degree nodes. This similarity function can be defined as

$$s(x,y) = \frac{|\Gamma_x \cap \Gamma_y|}{\max(|\Gamma_x|, |\Gamma_y|)}$$

This method has the same computational complexity that the hub promoted index, which is $O(vk^3)$.

3.1.11 The local Leicht-Holme-Newman index (LLHN)

This index is defined as the ratio of actual paths of length two between two nodes and a value proportional to the expected number of paths of length two between them [36]. Its own authors proclaim that this index is a more sensitive measure of structural equivalence than others like the Salton index or the Jaccard index. The similarity function define by this index can be computed as

$$s(x,y) = \frac{|\Gamma_x \cap \Gamma_y|}{|\Gamma_x||\Gamma_y|}$$

As almost all local methods, this one has a time complexity of $O(vk^3)$.

3.1.12 The individual attraction index (IA)

This index is similar to the resource allocation index but also takes into account how connected are the shared neighbors [37]. It makes sense to assume that two nodes with the same number of shared neighbors are more likely to be connected if their neighborhood are also highly-connected among them. These links are called inner links. This similarity is computed as

$$s(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{|e_{z,\Gamma_x \cap \Gamma_y}| + 2}{|\Gamma_z|}$$

where $|e_{z,\Gamma_x \cap \Gamma_y}|$ is the number of links between node z and nodes from the set $\Gamma_x \cap \Gamma_y$. [37] also proposed an alternative version with a lower computational complexity, which they called the simple individual attraction index:

$$s(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{|e_{\Gamma_x \cap \Gamma_y}| + 2}{|\Gamma_z||\Gamma_x \cap \Gamma_y|}$$

where $|e_{\Gamma_x \cap \Gamma_y}|$ is the number of edges in the network that connect common neighbors of nodes x and y. The computational complexity of the more complex version, which we will call IA1, is $O(vk^4)$, whereas the computation complexity of the more efficient version, which we will call IA2, is $O(vk^3)$.

3.1.13 Mutual information (MI)

This method treats the problem from the perspective of information theory by computing the conditional self-information of the existence of a link given the set of common neighbors [38]. The similarity among two nodes is computed as

$$s(x,y) = -I(e_{x,y}|\Gamma_x \cap \Gamma_y) = -I(e_{x,y}) + \sum_{z \in \Gamma_x \cap \Gamma_y} I(e_{x,y}; z)$$

where $I(e_{x,y})$ is the self-information of x and y being connected, which is computed as

$$I(e_{x,y}) = -\log_2 \left(1 - \prod_{i=1}^{|\Gamma_y|} \frac{|E| - |\Gamma_x| - i + 1}{|E| - i + 1} \right)$$

and $I(e_{x,y}; z)$ is the mutual information of the existence of a link between x and y and the set of shared neighbors of these nodes, which is computed as

$$I(e_{x,y}; z) = \frac{1}{|\Gamma_z|(|\Gamma_z| - 1)} \sum_{u,v \in \Gamma_z : u \neq v} (I(e_{u,v}) - I(e_{u,v}|z))$$

Finally, the conditional self-information of x and y being connected is computed as

$$I(e_{x,y}|z) = -\log_2 \frac{|\{e_{x,y} : x, y \in \Gamma_z, e_{x,y} \in E\}|}{\frac{1}{2}|\Gamma_z|(|\Gamma_z| - 1)}$$

The complexity of evaluating a single link is $O(k^4)$, as shown by its authors. Therefore, applying it to nodes at distance two leads to a computational complexity $O(nk^6)$ when used as a local link prediction technique.

3.1.14 Local Naïve Bayes (LNB)

This method assumes that each shared neighbor has a different role or degree of influence, which can be estimated using probability theory [39]. This method estimates the similarity of two nodes as

$$s(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} f(z) \log(oR_z)$$

where o is a constant for the network, which is computed as

$$o = \frac{p_{unconnected}}{p_{connected}} = \frac{\frac{1}{2}|V|(|V| - 1)}{|E|} - 1$$

R_z is the role or influence of the node, which is computed as

$$R_z = \frac{2|\{e_{x,y} : x, y \in \Gamma_z, e_{x,y} \in E\}| + 1}{2|\{e_{x,y} : x, y \in \Gamma_z, e_{x,y} \notin E\}| + 1}$$

and $f(z)$ is a function that measures the influence of the node. The authors suggest $f(z) = 1$ from common neighbors, $f(z) = \frac{1}{\log|\Gamma_z|}$ from the Adamic-Adar index, or $f(z) = \frac{1}{|\Gamma_z|}$ from the resource allocation method. The computational complexity of this method is, therefore, $O(vO(f(z)) + vk^3)$.

3.1.15 CAR-based indices (CAR)

CAR-based methods are proposed under the hypothesis that two nodes are more likely to be linked if their common neighbors are members of a strongly inner-linked cohort, named a local-community (LC) [40]. This assumption allows us to give more importance to nodes interlinked with other neighbors. A CAR-based version of common neighbors is defined as

$$s(x, y) = \sum_{z \in \Gamma_x \cap \Gamma_y} 1 + \frac{|\Gamma_x \cap \Gamma_y \cap \Gamma_z|}{2}$$

In a similar way, a CAR-based variation of the resource allocation can be computed as

$$s(x, y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{|\Gamma_x \cap \Gamma_y \cap \Gamma_z|}{|\Gamma_z|}$$

The computational complexity of this approach relies on the underlying technique. Both examples shown above have a computational complexity $O(vk^4)$.

3.1.16 Functional similarity weight (FSW)

This method is derived from the Sørensen index considering that the probability of a node x interacting with a node y is independent of the probability of the node y interacting with the node x in directed networks [41]. However, this score can be also applied to undirected networks as

$$s(x, y) = \left(\frac{2|\Gamma_x \cap \Gamma_y|}{|\Gamma_x - \Gamma_y| + 2|\Gamma_x \cap \Gamma_y| + \lambda}\right)^2$$

where λ is computed as $\lambda = \max(0, \langle\Gamma\rangle - (|\Gamma_x - \Gamma_y| + |\Gamma_x \cap \Gamma_y|))$. This parameter is included to penalize the similarity between nodes when any of them has a small degree. The computational complexity of this method is $O(vk^3)$.

3.1.17 Local interacting score (LIT)

This score is an iterative variation of the functional similarity weight [42]. Initially, weights are assigned as $s^{x,y}(0) = 1$ for connected pairs of nodes and $s^{x,y}(0) = 0$ for the rest of pairs. Then, weights are iteratively updated as

$$s^{x,y}(t) = \frac{\sum_{u \in \Gamma_x \cap \Gamma_y} s^{z,x}(t-1) + \sum_{v \in \Gamma_x \cap \Gamma_y} s^{z,y}(t-1)}{\sum_{u \in \Gamma_x} s^{z,x}(t-1) + \sum_{v \in \Gamma_y} s^{z,y}(t-1) + \lambda(x) + \lambda(y)}$$

where $\lambda(x)$ is computed as

$$\lambda(x) = \max\left(0, \sum_{u \in V} \sum_{v \in \Gamma_u} s^{u,x}(t)/|V| - \sum_{z \in \Gamma_x \cap \Gamma_y} s^{z,x}(t-1)\right).$$

$\lambda(x)$ plays the role of the λ penalization in functional similarity weight. The computational complexity of each iteration is $O(vk^3)$. When the process is limited to l iterations, the final computational complexity is $O(lvk^3)$.

3.2 Global Approaches

Global similarity-based indices use the whole network topological information to score each link. These methods are not limited to measuring similarity between distance-two nodes. However, their computational complexity can make them unfeasible for large networks and their parallelization can be very complex, specially in distributed environments where the complete topology of the network may not be known by every computational agent. Despite they exhibit very diverse time complexities, their spatial complexity is $O(v^2)$, since they have to store a score for every pair of nodes.

3.2.1 Negated shortest path (NSP)

Negated shortest path [10] is a basic graph similarity measure that requires to compute the shortest path between a pair of nodes. Shortest paths can be efficiently computed with Dijkstra's algorithm, which has a $O(e \log v)$ complexity using an adjacency list representation of the network and a heap data structure for its priority queue. Given the shortest path between a pair of nodes x and y, their similarity can be computed as

$$s(x, y) = -|\mathit{shortest\ path}_{x,y}|$$

Since shortest paths must be computed for each node in the network, the overall time complexity of this method is $O(ev \log v)$. Negated path prediction accuracy is poor even when compared to most local methods. Other methods described below, based on multiple paths, obtain significantly better results. This fact illustrates the importance of considering indirect paths in link prediction techniques.

3.2.2 The Katz index (KI)

This index sums the influence of all possible paths between two pairs of nodes, incrementally penalizing paths by their length [43]. This index is defined as

$$s(x,y) = \sum_{l=1}^{\infty} \beta^l |paths^l_{x,y}| = \sum_{l=1}^{\infty} \beta^l (A^l)_{x,y}$$

where $paths^l_{x,y}$ is the set of paths of length l between nodes x and y, and A is the adjacency matrix of the network. It should be noted that the l-th power of the matrix has each of its entries equal to the count of paths of length l between the corresponding pair of nodes. The parameter β is a damping factor where $0 < \beta < 1$. Giving a larger value to this parameter increases the influence of longer paths. If 1 is added to each element of the diagonal of the resulting similarity matrix S, this expression can be written in matrix terms as $S = \beta AS + I$. The similarity between all pairs of nodes can be directly computed using the closed-form by rearranging for S in the previous expression and subtracting the previously added 1s to the elements in the diagonal:

$$S = (I - \beta A)^{-1} - I$$

where I is the identity matrix. The similarity for each pair of nodes x and y is $s(x,y) = S_{x,y}$, where $S_{x,y}$ is the (x,y)-element of the matrix S. The Katz index has a great predictive power but the high algorithmic complexity required to compute the inverse of a matrix limits its applicability to small networks. The time complexity of this method is $O(vk + v^3 + v)$ where the $O(vk)$ term is due to matrix subtraction and scalar multiplication, $O(v^3)$ is due to matrix inversion, and $O(v)$ comes from the subtraction of the diagonal elements in the identity matrix. Therefore, the time complexity of this method is $O(v^3)$.

3.2.3 The global Leicht-Holme-Newman index (GLHN)

The global version of the Leicht-Holme-Newman index is based on the same fundamentals that the Katz index [36]. This index assigns a similarity proportional to the number of paths between nodes. Similarity between all pairs of nodes is defined as

$$S = I + \sum_{l=1}^{\infty} \phi^l A^l$$

where the identity matrix term indicates maximal self-similarity. Parameter ϕ is similar to the damping factor used in the Katz index. The number of actual paths of length l between each pair of nodes can be replaced by their expected value, which is computed as

$$Expected\left((A^l)_{x,y}\right) = \frac{|\Gamma_x||\Gamma_y|}{2|E|}\lambda_1^{l-1}$$

where λ_1 is the largest or dominant eigenvalue of the network adjacency matrix A. Replacing A^l by $Expected(A^l)$, this similarity can be finally expressed as

$$S = 2|E|\lambda_1 D^{-1} \left(I - \frac{\beta}{\lambda_1} A \right)^{-1} D^{-1}$$

where D is a diagonal matrix with each element set to $D_{i,i} = |\Gamma_i|$ and β is a free parameter related to ϕ. If the constant factor is removed, this expression can be iteratively computed avoiding the matrix inversions as

$$S(t) = \frac{\beta}{\lambda_1} AS(t-1) + I$$

starting with $S(0)$ having all its elements set to zero. This iterative process is performed until convergence. Iterative methods require a threshold parameter ϵ with a value close to zero. When the absolute difference between two iterations is below this threshold, the process is considered to have converged.

In order to compute the required dominant eigenvalue, different approaches can be followed. One of the most efficient is power iteration, or the Von Mises iteration method. This technique performs a series of iterative steps that converge to the largest eigenvector. It can be expressed as

$$b(t) = \frac{Ab(t-1)}{\|Ab(t-1)\|}$$

where A is the adjacency matrix of the network and b is a vector of length $|V|$, which can be initialized with random values. This process is repeated until convergence. Finally, it has been shown that the dominant eigenvalue is equal to the norm of this vector so $\lambda_1 = \|b(c)\|$ where c is the step of convergence.

The similarity between two nodes is defined as $s(x, y) = S(c)_{x,y}$ where c is the time step when convergence is reached. The computational complexity of this method is analyzed as follows, considering c as the number of iterations required. The largest eigenvalue can be computed in $O(cvk)$ time using the power iteration method. The complexity of the iterative process to obtain $S(t)$ is $O(cv^2k)$. In summary, the complexity of the Global Leicht-Holme-Newman method is $O(cv^2k + cvk) = O(cv^2k)$.

3.2.4 Random walks (RW)

Given a graph and a starting node, let us suppose that we randomly select a neighbor of this node and move to it; then, we repeat this process for each reached node. This Markov chain of randomly-selected nodes is known as a random walk on the graph [44]. Random walks were introduced by the mathematician Karl Pearson and have been applied to describe lots of stochastic processes in many fields such as economics, physics, or biology [45]. If we define $\overrightarrow{p^x}$ as the probability vector of reaching any node starting a random walk from node x, the probability of reaching each node can be iteratively approximated by

$$\overrightarrow{p^x}(t) = M^T \overrightarrow{p^x}(t-1)$$

where M is the transition probability matrix defined by the adjacency matrix A normalized by rows, with $M_{i,j} = A_{i,j}/\sum_k A_{i,k}$, and $\overrightarrow{p^x}(0)$ has all its elements set to 0 except $\overrightarrow{p^x_x}(0)$, which is set to 1. The transition probability matrix must satisfy some properties to ensure convergence. These constraints are satisfied for undirected simple networks. The above equation is iteratively applied until a stop condition is met such as $\sum_{i \in V}(\overrightarrow{p^x_i}(t) - \overrightarrow{p^x_i}(t-1))^2 < \epsilon$, where ϵ is a threshold value close to zero. Finally, the similarity for each pair of nodes x and y is defined as $s(x,y) = \overrightarrow{p^x_y}$. Assuming that the network is sparse and c is the number of iterations until convergence, the time complexity of this method is $O(cv^2k)$, since a sparse multiplication between the adjacency matrix and the probability vector ($O(vk)$) is repeated for each node in each iteration. If we also take into account the matrix normalization, the final time complexity is $O(cv^2k + vk) = O(cv^2k)$.

3.2.5 Random walks with restart (RWR)

Let us consider a model based on the definition of random walk where we pick a node and move following a random walk with probability α or we return to the starting node with probability $(1-\alpha)$. This model is known as a random walk with restart [46]. It is almost equivalent to the rooted version of the popular Google's PageRank algorithm [47]. This problem can be defined as an optimization problem:

$$\min_{\overrightarrow{p^x}} \ \alpha \sum_{i,j \in V} M^T_{i,j}(\overrightarrow{p^x_i} - \overrightarrow{p^x_j})^2 + (1-\alpha)\sum_{i \in V}(\overrightarrow{p^x_i} - \overrightarrow{s^x_i})^2$$

where M is the transition probability matrix computed for random walks and $\overrightarrow{s^x}$ is the seed vector of length $|V|$ with all its elements set to 0 except for $\overrightarrow{s^x_x} = 1$. The closed-form solution of this equation is

$$\overrightarrow{p^x} = (1-\alpha)(I - \alpha M^T)^{-1}\overrightarrow{s^x}.$$

Although this computation may be unfeasible for large networks, it can be iteratively approximated by the following iterative equation

$$\overrightarrow{p^x}(t) = \alpha M^T \overrightarrow{p^x}(t-1) + (1-\alpha)s^x$$

where $\overrightarrow{p^x}(0)$ has all its elements initially set to zero. This expression is iteratively applied, as in the random walk method. Since this measure is not symmetric, the final similarity for each pair of nodes is defined as $s(x,y) = \overrightarrow{p^x_y} + \overrightarrow{p^y_x}$. Like the random walk method, if we assume that the network is sparse, its complexity is $O(vc(vk+k)+vk) \approx O(cv^2k)$ where the $O(vc(vk+k))$ term is the time complexity of random walk including vector addition and the $O(vk)$ term is the computational complexity of the normalization process.

3.2.6 Flow propagation (FP)

Despite the fact that the random walk with restart method converges to a vector of probabilities, its iterative definition corresponds to a propagation process. Alternative normalizations can be used for the adjacency matrix. For example, [48] proposed to apply RWR replacing the normalized adjacency matrix with the normalized Laplacian matrix, which can be computed as

$$M = D^l A D^r$$

where D^l and D^r are diagonal matrices whose elements are respectively defined as $D^l_{i,i} = 1/\sqrt{\sum_j A_{i,j}}$ and $D^r_{i,i} = 1/\sqrt{\sum_j A_{j,i}}$. The computational complexity of this method is the same than the random walk with restart method complexity, since the transition matrix can be computed by the multiplication of each adjacency matrix entry by a scalar value.

3.2.7 Maximal entropy random walk (MERW)

Nodes tend to be linked to central nodes in structured networks. The maximal entropy random walk method incorporates the centrality of nodes in order to model this behaviour [49]. This method aims to maximize the entropy rate of a walk μ that is defined as

$$\mu = \lim_{l \leftarrow \infty} \frac{-\sum_{path^l_{x,y} \in paths^l} p(path^l_{x,y}) \ln p(path^l_{x,y})}{l}$$

where $p(path^l_{x,y}) = M_{x,h} M_{h,i} ... M_{i,j} M_{j,y}$. In order to maximize the entropy, each element of the transition matrix is computed as

$$M_{i,j} = \frac{A_{i,j}}{\lambda} \frac{\psi_j}{\psi_i}$$

where λ is the largest eigenvalue of the adjacency matrix and ψ is the normalized eigenvector with respect to λ satisfying $\sum_{x \in V} \psi^2_x = 1$. Following this theory, [50] proposes entropy maximizations of other global techniques described in this survey. The computational complexity of the normalization process is $O(cvk+vk)$ since computing the largest eigenvalue and the associated vector is $O(cvk)$ using the power iteration method as we have described. The final computational complexity of this method is $O(cvk+2vk+vc(vk+k)) = O(cv^2k)$.

3.2.8 SimRank (SR)

SimRank is a method that computes how soon two random walkers starting from nodes x and y are expected to meet [51]. This method is recommended for directed or mixed networks. However, as the number of undirected edges increases, it becomes impractical. It is recursively defined as

$$s(x,y) = \beta \frac{\sum_{i \in \Gamma_x} \sum_{j \in \Gamma_y} s(i,j)}{|\Gamma_x||\Gamma_y|}$$

where $s(z,z) = 1$ and $0 < \beta < 1$ is a damping factor. The high algorithmic complexity of this method makes it necessary to apply optimization techniques for its computation. Its original authors suggest pruning recursive branches beyond a radius l. The authors of SimRank suggest a near-linear complexity with a large constant factor in directed networks. However, this recursive expansion process has a complexity $O(k^{2l})$. Since two nested summations must be performed, the complexity for each pair of nodes is $O(k^{2l+2})$. This score must be computed for each pair of nodes so the final algorithmic time complexity is $O(v^2 k^{2l+2})$. It should be noted that, if pruning is applied with a radius lower than the network diameter, this method can be considered as a quasi-local method (quasi-local methods are described below in Section 3.3).

3.2.9 Pseudoinverse of the Laplacian matrix (PLM)

The Laplacian matrix provides an alternative representation of a graph, and it is widely-used in spectral graph theory [52]. It can be defined as $L = D - A$ where D is a diagonal matrix of size $|V|$ with each element $D_{i,i} = |\Gamma_i|$ and A is the adjacency matrix of the graph. The Moore-Penrose pseudoinverse of the Laplacian Matrix is noted as L^+. This matrix is a kernel and can be considered as a similarity matrix [53]. The Moore-Penrose pseudoinverse can be computed in many different ways. One of the most popular approaches, implemented in most important mathematical packages such as MATLAB, GNU Octave, or NumPy, is based on singular value decomposition (SVD). Detailed steps to compute the SVD are included in the description of the low-rank approximation preprocessing method (see section 6). Given the Laplacian matrix decomposition $L = U \Sigma V^T$, the Moore-Penrose pseudoinverse is computed as $L^+ = V \Sigma^+ U^T$ where Σ^+ is the matrix obtained from replacing each non-zero element of Σ by its inverse. Similarity can be computed using an inner-product-based measure such as the cosine distance:

$$s(x,y) = \frac{L^+_{x,y}}{\sqrt{L^+_{x,x} L^+_{y,y}}}$$

The complexity of the Moore-Penrose pseudoinverse of the Laplacian matrix computation is dominated by the cost of computing the SVD, which is $O(v^3)$. Once this matrix has been computed, the complexity of computing the similarity for each pair of nodes is $O(v^2)$. The overall complexity of this global link prediction method is, therefore, $O(v^3 + v^2) = O(v^3)$.

3.2.10 Average commute time (ACT)

The average commute time is defined as the average number of steps that a random walker starting from node x takes to reach a node y for the first time and go back to x [44]. If the number of steps needed to reach node y starting from node x in a random walk is denoted by $m(x, y)$ (also known as hitting time), the ACT value $c(x, y)$ between both nodes can be defined as

$$n(x, y) = m(x, y) + m(y, x)$$

The average commute time can also be computed in terms of the pseudoinverse of the Laplacian matrix L:

$$n(x, y) = |E|(L^+_{x,x} + L^+_{y,y} - 2L^+_{x,y})$$

where $\sqrt{n(x, y)}$ defines a distance measure called the Euclidean commute time distance (ECTD) between nodes x and y [53]. Finally, the similarity between two nodes can be computed as the inverse of the squared ECDT between both nodes, ignoring the constant $|E|$:

$$s(x, y) = \frac{1}{L^+_{x,x} + L^+_{y,y} - 2L^+_{x,y}}$$

The complexity of this method is the same we obtained for the pseudoinverse of the Laplacian matrix method, which was $O(v^3 + v^2) = O(v^3)$.

3.2.11 Random forest kernel index (RFK)

In graph theory, a spanning tree of a graph G is defined as a connected undirected subgraph with no cycles that includes all the vertices and some or all the edges of G. The matrix-tree theorem [54] states that the number of spanning trees in G is equal to any cofactor of an entry of its Laplacian representation. A cofactor is the determinant of the matrix obtained by removing the row and column of a given element. A rooted forest is defined as the union of disjoint rooted spanning trees. It can be proved that the cofactor of $(I + L)_{x,y}$ is equal to the number of spanning rooted forests in which x and y are contained in the same x-rooted spanning tree. The inverse of this number can be considered as a measure of accessibility between x and y. Therefore, a similarity measure can be defined as

$$S = (I + L)^{-1}$$

Given this similarity matrix, the similarity between a pair of nodes is $s(x, y) = S_{x,y}$. The algorithmic time complexity of this method is $O(v^3 + vk + v) = O(v^3)$ if we take into account the Laplacian matrix computation, $O(vk)$, the addition of the diagonal elements of the identity matrix, $O(v)$, and the matrix inversion, $O(v^3)$.

3.2.12 The Blondel index (BI)

The Blondel index was initially proposed to measure similarity for a pair of vertices in different graphs [55]. However, it can be adapted to work in a single graph. It is iteratively computed as

$$S(t) = \frac{AS(t-1)A^T + A^T S(t-1)A}{\|AS(t-1)A^T + A^T S(t-1)A\|_F}$$

where $S(0) = I$ and $\|M\|_F$ is the Frobenius matrix norm. The Frobenius norm for a matrix is computed as

$$\|M_{m\times n}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(M_{i,j})^2}$$

This measure is iteratively computed as in random-walk-based methods. The final similarity between a pair of nodes is defined as $s(x, y) = S_{x,y}(c)$ where c is the odd time step when convergence is reached. It is important to remark that this method reaches convergence only in odd iterations. The difference between the similarity matrices obtained in odd iterations must be computed in order to test for convergence.

The complexity of this method is $O(cv^2k + cv^2) = O(cv^2k)$, which can be derived from sparse matrix multiplications, $O(v^2k)$, matrix addition, matrix division by a scalar, and the Frobenius norm, $O(v^2)$.

3.3 Quasi-local Approaches

Quasi-local methods have recently emerged to strike a balance between local and global measures. Quasi-local approaches are almost as efficient to compute as local methods but also consider additional topological information, as global methods do. They do not take into account the similarity between any arbitrary pair of nodes in the network but they are neither limited to neighbors of neighbors. Some quasi-local methods have access to the whole network but their algorithmic time complexity is still below the time complexity of global methods. Their spatial complexity is $O(vk^{2+s})$, where s depends on specific parameters that set the number of iterations or the length of the paths considered.

3.3.1 The local path index (LPI)

This index is strongly based on Katz index but it only considers a finite number of path lengths [56]. The similarity matrix can be computed as

$$S = \sum_{i=2}^{l} \beta^{i-2} A^i$$

where $l > 2$ is the maximal path length and β a damping factor. It should be noted that, when $l = 2$, it would be equivalent to the common neighbors method. Similarity for each pair of nodes x and y is defined as $s(x, y) = S_{x,y}$. This measure is typically used with $l = 3$ due to its algorithmic complexity. When the damping factor is set to a low value, this measure obtains very similar results to the Katz index but avoids the matrix inversion computation. If the power of each adjacency matrix from the previous summation term is reused, the complexity of this method $O(lv^2k + lv^2) = O(lv^2k)$.

3.3.2 Local random walks (LRW)

Local random walks exploit the concept of random walks but limit the number of iterations to a fixed a priori small number l [44]. This method does not focus on the stationary state when convergence is reached like other random walk-based approaches. It is defined as

$$s^{x,y}(t) = \frac{|\Gamma_x|}{2|E|} \overrightarrow{p_y^x}(t) + \frac{|\Gamma_y|}{2|E|} \overrightarrow{p_x^y}(t)$$

where $\overrightarrow{p_y^x}(t)$ is the probability vector obtained by the random walk at iteration t. This method only requires us to compute a limited number of random walk steps for all the nodes in the network, so its computational complexity is $O(lv^2k)$, where usually $l < i$, being i the number of steps the random walk method would need to converge.

3.3.3 Superposed random walks (SRW)

Random walk-based methods are too sensitive to the topology of the network in distant zones. The superposed random walk method, which is based on the local random walk method, has been proposed to counteract this issue by continuously releasing the walker at the starting node [44]. This behaviour can be obtained by superposing each walker contribution as

$$s^{x,y}(t) = \sum_{i=1}^{t} \left(\frac{|\Gamma_x|}{2|E|} \overrightarrow{p_y^x}(i) + \frac{|\Gamma_y|}{2|E|} \overrightarrow{p_x^y}(i) \right)$$

The final similarity is computed as $s(x, y) = s^{x,y}(l)$. The computational complexity of this method is $O(lv^2k + lvk) = O(lv^2k)$.

3.3.4 Third-order resource allocation based on common neighbor interactions (ORA-CNI)

This measure is an extension of resource allocation based on common neighbor interactions that also takes into consideration distance three paths [27]. It redefines resource allocation for nodes at distance three. It is computed as

$$s(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{|\Gamma_z|} + \sum_{\substack{e_{i,j} \in E, \\ |\Gamma_i| < |\Gamma_j|, i \in \Gamma_x, j \in \Gamma_y}} \left(\frac{1}{|\Gamma_i|} - \frac{1}{|\Gamma_j|} \right) + \beta \sum_{[x,p,q,y] \in paths^3_{x,y}} \frac{1}{|\Gamma_p||\Gamma_q|}$$

where β is a damping factor to adjust the influence of the 3-hop resource allocation term. This method starts from the resource allocation based on common neighbors interactions complexity, which, it must also compute for distance-three nodes. Adding the third term to reach 3-hop nodes increases its computational complexity to $O(vk^3(k^2+k^3)) = O(vk^6)$.

3.3.5 FriendLink (FL)

FriendLink is a quasi-local measure based on the path count between nodes of interest, like the local path index [57]. However, this method uses a normalization and other path length penalization mechanisms. The similarity between two nodes x and y is computed as

$$s(x,y) = \sum_{i=2}^{l} \frac{1}{i-1} \frac{(A^i)_{x,y}}{\prod_{j=2}^{i}(|V|-j)}$$

where $\frac{1}{i-1}$ is an attenuation factor similar to the Katz or the local path index damping factors; and the count of paths of length i between x and y is normalized by the count of paths of length i between x and y that would exist in a complete version of the network. The algorithmic complexity of this method is $O(lv^2k)$.

3.3.6 PropFlow predictor (PFP)

PropFlow is a similarity-based method that computes the probability that a restricted random walk starting from x ends at y in l steps or fewer [58]. Given a source node x and maximal path length l, this algorithm returns an array S_x where each element $S_{x,y}$ is the score assigned between the pair of nodes x and y. New links can be predicted, like in other similarity-based methods, by setting the similarity score to $s(x,y) = S_{x,y}$.

The pseudocode of the algorithm can be found in Algorithm 1. This method is similar to the random walk with restart or rooted PageRank algorithms, but it employs the localized method of propagation and, therefore, is more insensitive to noise far from the source node x. It is also more efficient to compute, since it employs a breadth-first search limited to l

steps. The analysis of this method, given that the maximum size of set $OldSearch$ is k^l and the process must be repeated $|V|$ times, results in a computational complexity of $O(vlk^l)$.

```
Input: Network $G = (V, E)$, node $x$ and max
       path length $l$.
Output: Score $S_{x,y}$ for all $n \leq l$-degree neighbors
        of $y$ from $x$.
$Found = \{x\}$;
$NewSearch = \{x\}$;
$S_{x,x} = 1$;
for each $z$ in $V - \{x\}$ do
    $S_{x,z} = 0$;
end
for $CurrentDegree$ from 0 to $l$ do
    $OldSearch = NewSearch$;
    $NewSearch = \emptyset$;
    for each $i$ in $OldSearch$ do
        for each $j$ in $\Gamma_i$ do
            $S_{x,j} \leftarrow S_{x,j} + \frac{S_{x,i}}{|\Gamma_i|}$;
            if $j$ is not in $Found$ then
                $Found = Found \cup \{j\}$;
                $NewSearch = NewSearch \cup \{j\}$;
            end
        end
    end
end
```

ALGORITHM 1: PropFlow predictor (unweighted version).

3.4 Summary

Our survey has shown that very different approaches can be used to perform link prediction. A large number of methods using diverse approaches have been described in this chapter. These methods have been catalogued using the taxonomy shown in Figure 1. This taxonomy has two levels, which allow us to categorize each method based on its main features.

Similarity-based methods are the most studied category in link prediction and they conform the core of this survey. These methods are usually applied in massive networks so their time complexity is a fundamental feature. A summary of the similarity-based techniques that have been studied in this survey and their time complexity is shown in Table 1, grouping methods by their taxonomic classification.

As it has been seen in this chapter, most link prediction techniques are heuristics based on some coherent assumptions. The heuristic nature of the link prediction problem allows a rich variety of approaches that might work better or worse depending on the particular context, since network formation is a complex process and some fundamental factors can vary among networks. This implies that it is impossible to devise a method that works better than all other methods for all networks.

Type	Name	Time complexity	Reference
Local	CN	$O(vk^3)$	[4]
	AA	$O(vk^3)$	[23]
	RA	$O(vk^3)$	[24]
	RA-CNI	$O(vk^4)$	[27]
	PA	$O(vk^2)$	[28]
	JA	$O(vk^3)$	[30]
	SA	$O(vk^3)$	[31]
	SO	$O(vk^3)$	[33]
	HPI	$O(vk^3)$	[35]
	HDI	$O(vk^3)$	[35]
	LLHN	$O(vk^3)$	[36]
	IA1	$O(vk^4)$	[37]
	IA2	$O(vk^3)$	[37]
	MI	$O(nk^6)$	[38]
	LNB	$O(O(f(z)) + vk^3)$	[39]
	CAR	$O(vk^4)$	[40]
	FSW	$O(vk^3)$	[41]
	LIT	$O(lvk^3)$	[42]
Global	NSP	$O(ev \log v)$	[10]
	KI	$O(v^3)$	[43]
	GLHN	$O(cv^2k)$	[36]
	RW	$O(cv^2k)$	[45]
	RWR	$O(cv^2k)$	[46]
	FP	$O(cv^2k)$	[48]
	MERW	$O(cv^2k)$	[50]
	SR	$O(v^2k^{2l+2})$	[51]
	PLM	$O(v^3)$	[53]
	ACT	$O(v^3)$	[53]
	RFK	$O(v^3)$	[54]
	BI	$O(cv^2k)$	[55]
Quasi-local	LPI	$O(lv^2k)$	[56]
	LRW	$O(lv^2k)$	[44]
	SRW	$O(lv^2k)$	[44]
	ORA-CNI	$O(vk^6)$	[27]
	FL	$O(lv^2k)$	[57]
	PFP	$O(vlk^l)$	[58]

Table 1: Computational complexity and references for similarity-based link prediction methods. Columns (from left to right): type of similarity-based link prediction technique, name of the technique, computational complexity of the technique, and original reference to the technique.

3.5 Experimentation

In practice, link prediction strongly relies on similarity-based techniques, since most of them are efficient enough to be applied to massive networks. Furthermore, similarity-based scores are usually used as features when more complex approaches are applied using other algorithmic techniques. We have performed an extensive comparison of these techniques in order to study how they behave in different kinds of networks. As far as we know, the most complete existing comparison was [13]. We have implemented and applied to different networks all the similarity-based techniques described above. Other methods described in the following sections have not been considered due to their computational complexity and their need to tune different parameters, which would lead to unrepresentative results. In the following section, we describe the set of networks used in our experiments, how they were preprocessed, and how their structural properties were measured. In addition, results obtained are presented and discussed.

3.5.1 Datasets

Seven networks with different backgrounds and different topological properties were collected: a protein-protein interaction network of budding yeast (YST, [59]), a neural network of Caenorhabditis elegans (CEL, [60]), a network describing face-to-face contacts of people during the exhibition Infectious: Stay Away in 2009 at the Science Gallery in Dublin (INF, [61]), a frequent copurchasing network of books about US politics published in 2004 during the presidential election campaign and sold by Amazon (BCK, [62]), a network of friendships among users of the hamsterster.com website (HMT, [63]), a North American Transportation Atlas Data (NORTAD) flights from 1997 network (USA, [64]), and a coauthorship network of scientists working on network theory and experiments (NSC, [65]). Each network was preprocessed to make their links undirected, isolated nodes were deleted, and duplicated links and self-loops were removed. Some important topological properties of these networks were computed and included in the supplementary material.

3.5.2 Evaluation and results

To evaluate each technique, we conducted a 5-fold cross-validation as described in the supplement. We report the precision (see Table 2) and the AUC (see Table 3) for each possible method and network combination. Methods with parameters were evaluated with different reasonable hand-picked values. Local interacting score has been tested with 2, 4, and 6 iterations and, as suggested by its authors, its best results were obtained with only 2 iterations. Katz index (β), global Leicht-Holme-Newman index (ϕ), SimRank and local path index (β) with l fixed to 3 in both cases, and third-order RA-CNI (β) were tested with values 0.1, 0.01 and 0.001. The best results for Katz, SimRank and local path were obtained with $\beta = 0.001$. The best results for the Leicht-Holme-Newman index were obtained with $\phi = 0.1$. ORA-CNI was set to $\beta = 0.001$. All random walk-based methods with an α parameter were tested for 0.5, 0.7 and 0.9, finally selecting 0.5 in all cases since the results were very stable for all the tested parameter values. Finally, local random walk, superposed random walk and PropFlow predictors were tested with parameters t and l set to 3, 5, and 7. The best average precision was obtained choosing 3 in all cases.

The number of times that each method appears in the top five for each network is

Method	YST	CEL	INF	BCK	HMT	USA	NSC
CN	0.0876	0.1119	0.3484	0.2101	0.2453	0.4091	0.3561
AA	0.1080	0.1532	0.4080	0.2608	0.3267	0.4558	0.6217
RA	0.0876	0.1448	0.4163	0.2563	0.3959	0.5160	**0.6652**
RA-CNI	0.0951	0.1490	0.4242	0.2630	**0.4406**	0.4965	0.6306
PA	0.0310	0.1051	0.0848	0.1820	0.0787	0.3819	0.1932
JA	0.0030	0.0373	0.4104	0.1297	0.2472	0.1081	0.4884
SA	0.0026	0.0340	0.4188	0.1395	0.2409	0.0866	0.4997
SO	0.0030	0.0373	0.4104	0.1297	0.2472	0.1081	0.4884
HPI	0.0000	0.0000	0.2764	0.1463	0.0000	0.0000	0.0005
HDP	0.0104	0.0364	0.4017	0.1053	0.2461	0.0810	0.4208
LLHN	0.0002	0.0023	0.1271	0.0897	0.0817	0.0104	0.2334
IA1	0.1079	0.1569	0.4195	0.2619	0.4093	0.4638	0.6170
IA2	0.0855	0.1493	**0.4268**	0.2427	0.4199	0.4927	0.6596
MI	0.1228	0.1676	0.4040	0.2313	0.3324	0.4365	0.5125
LNB-CN	0.1189	0.1569	0.3964	0.2494	0.2996	0.4440	0.5414
LNB-AA	0.1190	0.1555	0.4036	0.2534	0.3574	0.4685	0.6362
LNB-RA	0.0954	0.1462	0.4105	0.2540	0.4019	0.5169	0.6610
CAR-CN	0.1073	0.1240	0.3942	0.2029	0.2816	0.4228	0.3914
CAR-AA	0.1241	0.1480	0.4047	0.2222	0.3191	0.4322	0.5074
CAR-RA	0.1245	0.1560	0.4148	0.2517	0.3873	0.4487	0.5074
FSW	0.0504	0.0792	0.3637	0.1349	0.2266	0.2158	0.4849
LIT	0.1049	0.1220	0.4253	0.1814	0.4097	0.3984	0.5022
NSP	0.0000	0.0001	0.0001	0.0007	0.0000	0.0001	0.0003
KI	0.1186	0.1513	0.3924	0.2471	0.2595	0.4332	0.4300
GLHN	0.0876	0.1119	0.3484	0.2101	0.2453	0.4091	0.3561
RW	0.0891	0.1276	0.2555	0.1927	0.2639	0.1599	0.4282
RWR	0.1175	**0.1983**	0.4090	0.2268	0.3751	0.3316	0.5292
FP	0.1013	0.1643	0.3508	0.2449	0.2845	0.4059	0.4303
MERW	0.0129	0.0666	0.2575	0.1565	0.2618	0.0927	0.5036
SR	0.0009	0.0033	0.1309	0.1066	0.0877	0.0127	0.2918
PLM	0.0176	0.1066	0.1703	0.2494	0.0017	0.3782	0.0149
ACT	0.0284	0.0889	0.1826	0.2199	0.0807	0.3791	0.0543
RFK	0.0260	0.0847	0.2329	0.1746	0.2381	0.0814	0.4712
BI	0.0710	0.1359	0.1483	0.2221	0.0988	0.3923	0.1940
LPI	0.1069	0.1438	0.3852	0.2404	0.1723	0.4200	0.4194
LRW	**0.1857**	0.1839	0.3819	0.2177	0.4232	0.4972	0.4927
SRW	0.1380	0.1657	0.4123	0.2540	0.4265	**0.5230**	0.6580
ORA-CNI	0.0993	0.1490	0.4242	**0.2630**	0.4405	0.4967	0.6601
FL	0.1069	0.1443	0.3848	0.2358	0.2328	0.4191	0.4194
PFP	0.0405	0.1331	0.1834	0.2110	0.2456	0.1449	0.4776

Table 2: Precision results. Average precision of the five iterations of the cross-validation performed for each method and network pair.

Method	**YST**	**CEL**	**INF**	**BCK**	**HMT**	**USA**	**NSC**
CN	0.6850	0.8274	0.9264	0.8691	0.9523	0.9278	0.9056
AA	0.6855	0.8450	0.9300	0.8782	0.9553	0.9391	0.9061
RA	0.6854	0.8485	0.9305	0.8801	0.9561	0.9439	0.9061
RA-CNI	0.6854	0.8495	0.9307	0.8803	0.9564	0.9433	0.9061
PA	0.6846	0.8091	0.8991	0.8505	0.9386	0.9177	0.9043
JA	0.6837	0.7766	0.9285	0.8558	0.9492	0.8926	0.9059
SA	0.6837	0.7831	0.9285	0.8621	0.9501	0.8995	0.9059
SO	0.6837	0.7766	0.9285	0.8558	0.9492	0.8926	0.9059
HPI	0.6834	0.7933	0.9255	0.8671	0.9484	0.8666	0.9058
HDP	0.6836	0.7680	0.9277	0.8475	0.9483	0.8878	0.9057
LLHN	0.6828	0.7276	0.9195	0.8314	0.9411	0.7821	0.9056
IA1	0.6854	0.8490	0.9305	0.8791	0.9562	0.9418	0.9061
IA2	0.6853	0.8483	0.9306	0.8795	0.9562	0.9434	0.9061
MI	0.6527	0.8325	0.9083	0.8426	0.9379	0.9089	0.7765
LNB-CN	0.6858	0.8411	0.9263	0.8717	0.9541	0.9337	0.9057
LNB-AA	0.6858	0.8445	0.9285	0.8765	0.9557	0.9403	0.9059
LNB-RA	0.6857	0.8451	0.9295	0.8772	0.9561	0.9440	0.9059
CAR-CN	0.6850	0.8272	0.9264	0.8682	0.9525	0.9269	0.9056
CAR-AA	0.5792	0.7130	0.8254	0.7224	0.8832	0.8971	0.7554
CAR-RA	0.5792	0.7140	0.8255	0.7230	0.8838	0.9001	0.7554
FSW	0.6839	0.7822	0.9256	0.8535	0.9473	0.8949	0.9058
LIT	0.6730	0.8313	0.9206	0.8550	0.9501	0.9164	0.8424
NSP	0.7887	0.7443	0.9121	0.8456	0.9423	0.8135	0.9142
KI	0.8044	0.8507	0.9528	0.8946	0.9630	0.9180	0.9147
GLHN	0.6850	0.8274	0.9264	0.8691	0.9523	0.9278	0.9056
RW	0.7811	0.8354	0.9494	0.8851	0.9551	0.8796	0.9151
RWR	0.8029	**0.8967**	**0.9637**	**0.9183**	0.9681	0.9362	0.8892
FP	0.8113	0.8873	0.9599	0.9018	0.9674	0.9382	0.9154
MERW	0.7806	0.8519	0.9454	0.8906	0.9575	0.8809	0.9151
SR	0.6828	0.7310	0.9200	0.8334	0.9414	0.7856	0.9056
PLM	0.7725	0.8480	0.9439	0.8908	0.6435	0.9407	0.5263
ACT	0.7659	0.7370	0.7969	0.7456	0.8793	0.8749	0.5654
RFK	0.7964	0.8614	0.9549	0.8984	0.9593	0.9106	0.9150
BI	0.7784	0.7159	0.7730	0.8166	0.8800	0.8674	0.9081
LPI	0.8025	0.8345	0.9501	0.8882	0.9591	0.9136	0.9137
LRW	0.8164	0.8874	0.9514	0.8928	0.9647	0.9291	0.8536
SRW	**0.8210**	0.8936	0.9608	0.9139	**0.9726**	**0.9463**	0.9164
ORA-CNI	0.8135	0.8539	0.9515	0.8977	0.9682	0.9386	0.9164
FL	0.8025	0.8342	0.9500	0.8882	0.9619	0.9112	0.9137
PFP	0.8159	0.8645	0.9568	0.9081	0.9636	0.8899	**0.9171**

Table 3: AUC results. Average AUC of the five iterations of the cross-validation performed for each method and network pair.

Method	**1st**	**2nd**	**3rd**	**Top 5**
AA				1
RA	1		1	3
RA-CNI	1	1		3
IA1			1	2
IA2	1			3
MI			1	2
LNB-RA		2		2
CAR-AA				1
CAR-RA			1	1
LIT		1		1
RWR	1			1
FP				1
LRW	1	1		4
SRW	1	1	1	5
ORA-CNI	1	1	2	5

Method	**1st**	**2nd**	**3rd**	**Top 5**
RA			1	1
RA-CNI				1
IA2				1
LNB-RA		1		1
RWR	3		1	4
FP			1	6
MERW				1
RFK				2
LRW		1	1	3
SRW	3	3	1	7
ORA-CNI		2		3
PFP	1		2	5

Table 4: Summary of methods appearing in the top five of the rank. Number of times that each method appears in the first, second, third position and in top five for all networks according to precision (left table) and AUC (right table). The rows of methods that never are in the top five of the rank are omitted.

summarized in the Table 4 for precision and AUC. Different conclusions can be extracted from our empirical results.

Global techniques obtain the worst results in average. Their poorer performance, and the fact that tuning their parameters strongly penalize indirect paths, suggest that global methods are worse because they tend to consider too much noise. It can be seen that global methods are less present in the top five. Only RWR, FP, MERW, and RFK reach the top five in a few occasions in networks with a low average clustering coefficient.

Local techniques work surprisingly well. Some local methods reach the top 5 a few times, including AA, CAR-based methods, and the mutual information technique; however, RA and its variation RA-CNI, LNB-RA method, and IA techniques are present in the top 5 a reasonable number of times when compared to other methods. These results suggest that most of the information that can be used to perform link prediction is of local nature.

Quasi-local techniques also obtain good results in average. ORA-CNI, PFP, LRW and SRW appear among the bests for almost all networks. The SRW technique is the method that has shown to obtain the bests results in average in our experiments.

Finally, the variety of methods in the top of the ranking shows that the performance of each technique strongly depends on the structural properties of the network. This highlights the importance of analyzing the properties of the network before choosing a particular link prediction technique. As we observe in our results, the quality of the results is related to the average clustering coefficient of the nodes with degree above one. This is reasonable since most link prediction techniques are variations of counting shared neighbors, and the count of shared neighbors increases as the clustering coefficient does. Other variable that seems to play an important role is the average degree. This makes sense since as we know the more neighbors of a node, the more information we have to predict new links for it. However, revealing which specific properties play such an important role in link prediction is still an unsolved problem that requires further work.

4 Probabilistic and Statistical Approaches

Many network formation models have been successfully described in terms of statistical and probabilistic concepts [66]. These studies have opened the door to link prediction techniques based on statistical analysis and probability theory. These approaches usually assume that the network has a known structure. They build a model that fits the structure and estimate model parameters using statistical methods. These parameters are used to compute the formation probability of each non-observed link. These probability values can be used to rank potential links as we did in similarity-based methods.

4.1 The hierarchical structure model

Some studies show that many real networks are hierarchically-organized, including metabolic networks, protein interaction networks, internet domains, and some social networks like actor networks [67]. In hierarchical networks, nodes with higher degree are expected to have a lower clustering coefficient than lower degree nodes. Hub nodes weakly-connect isolated communities of highly clustered nodes. This way, a hierarchical structure is formed.

The method proposed by [68] represents a hierarchically-structured network by a dendrogram with $|V|$ leaves and $|V|-1$ internal nodes. Each leaf represents a node from the network and each internal node represents a relationship among its descendant nodes in the dendrogram. Each internal node n has an associated probability p_n, which is equal to the probability of a link between nodes of both branches descending from it. Each network has multiple representations based on dendrograms depending on how internal nodes are set. Given a dendrogram representation D of the network, let e_n be the number of links in the network connecting nodes that have internal node n as their lowest common ancestor in D. The likelihood of dendrogram D together with the set of internal node probabilities can be estimated as

$$\mathcal{L}(D, \{p_n\}) = \prod_{n \in D} p_n^{e_n} (1 - p_n)^{l_n r_n - e_n}$$

where l_n and r_n are, respectively, the numbers of leaves in the left and the right subtrees with root n. If the dendrogram D is fixed, its likelihood can be maximized by a set of probabilities $\overline{p}_n$ computed as

```
**Input**: Network $G = (V, E)$, number $n$ of dendrograms to sample.
**Output**: Probability $P_{x,y}$ for all unconnected pairs of nodes.
$Samples = \emptyset$;
for $i$ from 1 to $n$ do
    Initialize the Markov chain with a random dendrogram;
    Run Monte Carlo algorithm until equilibrium is reached;
    Insert resulting dendrogram $D$ into $Samples$;
end
for each $e_{x,y}$ in $U_G - E$ do
    $avg\_prob = 0$;
    for each $sample$ in $Samples$ do
        $r \leftarrow$ lower common ancestor of $x$ and $y$ in $sample$;
        $avg\_prob \leftarrow avg\_prob + \frac{\overline{p}_r}{|Samples|}$ ;
    end
    $P_{x,y} = avg\_prob$;
end
```

ALGORITHM 2: Link prediction based on the hierarchical structure model.

$$\overline{p}_n = \frac{e_n}{l_n r_n}$$

$\overline{p}_n$ represents the ratio of the number of actual edges with respect to the number of potential ones. Based on this result, the e_n term can be removed from the likelihood formula to estimate the likelihood of a dendrogram at its maximum as

$$\mathcal{L}(D) = \prod_{n \in D} [\overline{p}_n^{\overline{p}_n}(1 - \overline{p}_n)^{1-\overline{p}_n}]^{l_n r_n}$$

Once the theoretical background is set, these results can be used to perform link prediction. A Markov chain Monte Carlo method [69] is used to sample a set of dendrograms with a probability proportional to their likelihood. The transitions between dendrograms consist of rearranging subtrees of the current dendrogram in other order. The complete procedure, which computes the probability of link formation between each pair of unconnected nodes, is described in Algorithm 2.

4.2 The stochastic block model

Most networks do not fit in a hierarchical schema. A more general approach is to consider that nodes are distributed in communities or blocks. The probability of link formation between two nodes directly depends on the block they belong to [70]. In this model, we are required to compute a partition $\mathcal{M}$ of the network where each node is assigned to one group or block $m \in \mathcal{M}$. Given a partition, the likelihood of the network structure can be estimated as

$$\mathcal{L}(G|\mathcal{M}) = \prod_{a,b \in \mathcal{M}} p_{a,b}^{l_{a,b}}(1 - p_{a,b})^{r_{a,b} - l_{a,b}}$$

where $l_{a,b}$ is the number of edges between nodes in groups a and b, $|\{e_{x,y} : x \in a, y \in b\}|$; and $r_{a,b}$ is the number of pairs between nodes of both groups, which are $|a||b|$ when $a \neq b$ and $|a|(|a|-1)$ when $a = b$. This likelihood is maximized for

$$\overline{p}_{a,b} = \frac{l_{a,b}}{r_{a,b}}$$

Applying the Bayes theorem, the probability of a link with maximum likelihood can be computed as

$$P_{x,y} = \frac{\sum_{\mathcal{M}\in\omega} \mathcal{L}(e_{x,y} \in E|\mathcal{M})\mathcal{L}(G|\mathcal{M})p(\mathcal{M})}{\sum_{\mathcal{M}'\in\omega} \mathcal{L}(G|\mathcal{M}')p(\mathcal{M}')}$$

where ω is the set of possible partitions. It should be noted that ω grows fast as the number of nodes in the network increases ($\omega \in O(2^{|V|})$), which makes this approach impractical for large networks. The Metropolis algorithm can be used to sample partitions, but this process is still computationally expensive.

4.3 The cycle formation model

The cycle formation model is based on the assumption that networks have the tendency to close cycles in their formation process [71]. This assumption matches with other techniques like the common neighbors method, which counts the number of cycles of length three what would be formed if the evaluated link existed. This method tries to capture longer cycles by extending the overall clustering coefficient to a generalized clustering coefficient. This generalized clustering coefficient is defined as

$$C(k) = \frac{\text{number of cycles of length } k}{\text{number of paths of length } k}$$

where k is the length of the cycles under analysis.

A cycle formation model of degree k, denoted as $CF(k)$ with $k > 0$, characterizes each order formation mechanisms, $g(1), ..., g(k)$, by a single coefficient, $c_1, ..., c_k$, which describes the conditional probability of k-order cycle formation. The expected clustering coefficient of degree k based on this model can be estimated as

$$f(c_1, ..., c_k) = \sum_i |G_i| Pr(G_i) Pr(e_{1,k+1} \in E|G_i)$$

where G_i is the set of possible connected graphs with i nodes for each order in this model. Finally, given the coefficients, the probability of the existence of a link is computed as

```
Input: Network $G = (V, E)$, model degree $k$.
Output: Probability $P_{x,y}$ for all unconnected pairs of nodes.
Compute Generalized Clustering Coefficients $C(2), ..., C(k)$;
$c_1$ = Connecting probability in random graph with same degree distribution that $G$;
$c_2 = \frac{(1-c_1)C(2)}{c_1 - 2c_1C(2) + C(2)}$;
for $i$ from 3 to $k$ do
    $c_i \leftarrow 0.5$;
end
for $i$ from 3 to $k$ do
    $c_i \leftarrow \arg\min_{c_i} |C(i) - f(c_1, ..., c_k)|$;
end
for each $e_{x,y}$ in $U_G - E$ do
    $P_{x,y} \leftarrow p_{x,y}(c_1, ..., c_k)$;
end
```

ALGORITHM 3: Link prediction based on the cycle formation model.

$$p_{x,y}(c_1, ..., c_k) = \frac{c_1 \prod_{i=2}^{k} c_i^{|paths_{x,y}^i|}}{c_1 \prod_{i=2}^{k} c_i^{|paths_{x,y}^i|} + (1-c_1) \prod_{i=2}^{k} (1-c_i)^{|paths_{x,y}^i|}}$$

The whole link prediction method based on this cycle formation model is reproduced in Algorithm 3.

4.4 The local co-occurrence model

The probabilistic methods presented above are prohibitive for large networks due to their high computational complexity. The local co-occurrence model proposes a scalable probabilistic method based on local topological features of the network [72].

A set $C_{x,y}$ composed of relevant nodes, called central neighborhood, is computed for each pair of nodes x and y. These sets can be obtained using different topological measures. The original paper proposes to compute these sets by obtaining all simple paths (without cycles) of length $1, ..., k$. The t nodes in most frequent paths are selected as the central neighborhood set of a pair of nodes. In addition, a collection of non-derivable itemsets (NDI) is efficiently computed for each of these pairs by a depth-first search [73]. Non-derivable itemsets are those itemsets whose occurrence statistics cannot be inferred from other itemset patterns. These itemsets provide non-redundant constraints that can be used to learn probabilistic models without losing information.

These sets are used to learn a Markov random field (MRF) undirected graph model. This model is iteratively built satisfying constrains associated to the NDI sets. Finally, the built model allows to compute the probability of existence of each link. The final link prediction method appears in Algorithm 4.

```
Input: Network $G = (V, E)$, central neighborhood set max size $t$, max
       path length $k$.
Output: Probability $P_{x,y}$ for all unconnected pairs of nodes.
for each $e_{x,y}$ in $U - E$ do
    $C_{x,y} = \emptyset$;
    for $l$ from 2 to $k$ do
        $p_i \leftarrow$ Compute and sort by length and frequency $paths^l_{x,y}$;
        for each $p$ in $p_i$ do
            if $|C_{x,y}| < t$ then
                Insert all nodes in $p$ into $C_{x,y}$;
            end
        end
    end
    NDI = Compute non-derivable itemsets from $C_{x,y}$;
    $R_{x,y} = \emptyset$;
    for each $ndi$ in $NDI$ do
        if $ndi$ in $C_{x,y}$ then
            Insert $ndi$ into $R_{x,y}$;
        end
    end
    $M$ = Initialize Markov Random Fields using $C_{x,y}$ and $R_{x,y}$;
    while not $M$ satisfies all constrains in $R_{x,y}$ do
        for each $r$ in $R_{x,y}$ do
            Update $M$ to force satisfying $r$;
        end
    end
    $P_{x,y}$ = Infer probability of $e_{x,y}$ from $M$;
end
```
ALGORITHM 4: Link prediction based on the local co-occurrence model.

5 Algorithmic Methods

All the approaches presented in the previous sections are based on computing a score for each non-observed link by defining a similarity or a probability function. However, link prediction can also benefit from other algorithmic approaches, including supervised learning and optimization techniques. These approaches have been less explored in the literature of link prediction but present interesting properties.

5.1 Classifier-based methods

The link prediction problem can be approached by classical supervised learning techniques. It can be seen as a classification problem with two classes: existence and absence of link. This is a very powerful technique since it can use any topological property and measure; or even any other link prediction measure as a feature. This approach, however, has to deal with the well-known class imbalance problem [74], since almost all real networks are sparse; that is, the number of absent links is extremely higher than the number of existent links.

Several classifier-based approaches have been proposed. Almost any type of classifier can be used. Some works have compared different classifiers including decision trees, support vector machines (SVMs), k-nearest neighbors, multilayer perceptrons, radial basis function networks, naive Bayes, and different ensembles of these classifiers [75]. Other authors have obtained good results using random forest classifiers [76]. Random forests are decision tree ensembles trained on the same training set but using different subsets of the available features.

Many classifier-based methods do not rank possible links like similarity-based or probabilistic methods. This property makes their comparison harder, since the number of predicted links in each class cannot be controlled in this case.

5.2 Metaheuristic-based methods

Link formation is a complex process with a large number of factors involved. All approaches are heuristic, in the sense that they try to outperform a random baseline predictor by making some assumptions about link formation in the studied network. Assuming that all links are formed by the same mechanism is an oversimplification of the problem.

Recently, [77] proposed an approach based on an evolutionary algorithm. Their method assumes that different link formation heuristics can coexist and cooperate in the same network. It uses an evolution strategy to optimize the influence of different base link predictors including local and global similarity-based indices and node similarity features in a Twitter reciprocal reply network. Each individual or candidate solution u in the population is a vector $w^{(u)}$ of as many real numbers as the number of heuristics being considered. Each of these values represents the weight or the influence of the heuristic in the network. Each candidate represents a similarity-based link predictor characterized by the similarity function

$$s(x,y) = \sum_{i=1}^{|w^{(u)}|} w_i^{(u)} s_i(x,y)$$

where $s_i(x,y)$ is the similarity function for the i-th heuristic. The fitness of each candidate is defined as the precision obtained by applying it on a second training subset and a second test subset created from sampled links from the original training set. A covariance matrix adaptation evolution strategy (CMA-ES) is applied to generate new candidates with better fitness by creating new populations of candidates based on combinations and mutations of the previous ones [78].

5.3 Factorization-based methods

Matrix factorization models have been widely used in recommender systems since they can extract latent features or use additional features to perform prediction [79]. For example, [80] has suggested a latent feature learning method for link prediction composed of a latent vector $\overrightarrow{l_i}$ for each node i, a scaling factor $F_{x,y}$ for each link, weights for node features W_n and a vector of weights for link features $\overrightarrow{w_l}$. Furthermore, each node i has a vector of features $\overrightarrow{a_i}$ and each link has a vector of features $\overrightarrow{b_{x,y}}$.

In this model, given the latent vectors, the scaling factor, and the weights, a prediction score is computed for each pair of nodes x and y as

$$s(x,y) = \frac{1}{1 + \exp(-\overrightarrow{l_x}^T F \overrightarrow{l_y} - \overrightarrow{a_x}^T W_n \overrightarrow{a_y} - \overrightarrow{w_l}^T \overrightarrow{b_{x,y}})}$$

Latent vectors, the scaling factor, and the weights are obtained in a training stage that optimizes the following function

$$\min_{l,F,W_n,\overrightarrow{w_l}} \sum_{e_{x,y} \in E} \ell(A_{x,y}, s(x,y)) + \Omega(l, F, W_n, \overrightarrow{w_l}))$$

where ℓ is a loss function and Ω is a regularizer term to prevent overfitting. These terms can be selected to customize the model [79]. This training stage is performed using Stochastic Gradient Descent (SGD) until convergence.

6 Preprocessing Methods

Preprocessing methods are also known as higher-level approaches or meta-approaches, since they are intended to be used in conjunction with other methods. The main goal of preprocessing approaches is to reduce the noise present in the networks as “weak” or “false” links, in order to improve the performance of the methods described above.

6.1 Low-rank approximation

This method simplifies the structure of the network to reduce its noise using the adjacency matrix A representation of the graph by solving the low-rank approximation problem [81]. This optimization problem tries to minimize a cost function that measures the fit between the original matrix and an approximation matrix of reduced rank. The rank of the approximated matrix is usually set to a relatively small number. This problem can be algorithmically solved in an efficient way using the singular value decomposition (SVD) of the original matrix, which is a factorization of the form

$$A = U \Sigma V^T$$

where U and V^T are unitary matrices and Σ is a diagonal matrix with no negative elements. There are different techniques to compute the SVD. The most basic approach relies on the fact that singular values are the square roots of the eigenvalues of AA^T. Indeed, given the expression of the decomposition, it can be stated that

$$AA^T = (U \Sigma V^T)(V \Sigma U^T) = U \Sigma^2 U^T$$

Since the columns of U are eigenvectors of A, it can be obtained using a technique to compute the eigenvectors of a matrix, and therefore, allowing to obtain the Σ matrix using simple linear algebra calculations. Unfortunately, this technique is not practical for large matrices due to lack of numerical accuracy. One of the most commonly used SVD algorithm, which shows a better accuracy, is given by [82].

Given the SVD of A, the low-rank matrix A' can be approximated by

$$A' = U_{1:|V|,1:k}\Sigma_{1:k,1:k}V^T_{1:k,1:|V|}$$

where k is the desired rank. The first eigenvectors explain most of the variance, so the low-rank approximated matrix maintains the overall structure of the graph but removing its less significant links.

6.2 Unseen bigrams

A bigram is a sequence of two adjacent elements in a string composed of tokens or words. The frequency distribution of bigrams has been extensively studied in many applications such as linguistics, speech recognition, or cryptography. Unseen bigrams are valid bigrams not observed in given string set. It has been observed that the same tokens in different bigrams with similar appearance distributions are likely to be interchangeable and to form unseen bigrams. For example, if we observe bigrams "a house", "the house", "a tree", "the tree", and "a car" then we can infer that "the car" is an unseen bigram. The idea of "substitution" presented by unseen bigrams can be adapted to link prediction in order to reduce noise by replacing a node by its most similar nodes [10]. For example, the common neighbors similarity can be rewritten as

$$S(x, y) = |S^\delta_x \cap \Gamma_y|$$

where S^δ_x is a set with the δ most similar nodes to x. The similarity method employed to obtain the set S^δ_x could also be the common neighbors similarity or any other similarity-based technique.

6.3 Filtering

Originally called clustering [10], we refer to it as filtering in order to avoid any ambiguity. Removing the weakest links (usually, those observed between nodes with a small number or no shared neighbors) could help improve the results obtained by link prediction methods due to the associated noise reduction. Most of the techniques presented in this survey assign a score $S(x, y)$ to non-observed links, but they can be applied to observed links in order to estimate their strength. Therefore, the filtering approach consists of applying any link prediction technique that assigns a score to each pair of nodes between connected nodes to weigh the observed link and remove the fraction γ of weakest links to obtain a cleaned-up network.

7 Link prediction in different kinds of networks

In this work, a large number of proposed techniques focusing only on undirected unweighed networks without attributes have been discussed. However, an increasing number of link prediction techniques for other types of networks and the consideration of non-topological additional information is also of interest in practice. The role of weak ties in weighted networks remains an open question [83]. Heterogeneous networks with different types of nodes and links require different approaches to those used in homogeneous networks [84]. Predicting not only a link but also its source and its destination in directed networks has recently started to be under study [85]. Link prediction in signed networks, where links can be positive or negative, is other active area of research [86]. Time-aware methods can be proposed for networks with links labelled with their time of formation [87]. Aligned networks are sets of different networks partially matched by anchor nodes and links, and sets of networks representing the same agent in different domains are available, so specific techniques to exploit this additional information are being developed [88]. Most of the presented methods in our survey can be adapted to the characteristics of these real-world usage scenarios.

8 Conclusions

In this survey, we have performed, as far as we know, the most comprehensive study about the link prediction methods that have been proposed in terms of the number of methods and networks employed. These methods have been classified according to a custom taxonomy based on their theoretical approach. The most important results that support specific link prediction techniques and network formation models have also been described.

A comprehensive experimentation has been performed for a large number of fundamental and state-of-the-art methods using a varied set of networks with different properties. It has been observed that new links can be better predicted using only local or quasi-local information in most networks. Considering indirect connections only adds noise and computational complexity to the link prediction problem.

Link prediction is a relatively young research area and many open challenges remain. Further studies are required in order to understand why some methods work better or worse than others depending on the network they are applied to. Studying which network structural properties lead to better performance for each technique is an open research problem. In addition, very few techniques adapt to the global structure of the network and no technique adapts to the local structure of networks. The main difficulty when dealing with complex networks in practice is their size, which limits the kinds of techniques that can be applied.

Link prediction remains an open research problem, given its importance in many applications. New techniques with better accuracy and performance trade-offs are expected to be proposed in the forthcoming future.

Acknowledgements

This work is partially supported by the Spanish Ministry of Economy and the European Regional Development Fund (FEDER), under grant TIN2012-36951, and partially by the Ministry of Education of Spain under the program "Ayudas para contratos predoctorales para la formación de doctores 2013" (grant BES-2013-064699).

References

[1] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[2] Gueorgi Kossinets and Duncan J Watts. Empirical analysis of an evolving social network. *Science*, 311(5757):88–90, 2006.

[3] Roberto Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013.

[4] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[5] Víctor Martínez, Carlos Cano, and Armando Blanco. Prophnet: A generic prioritization method through propagation of information. *BMC bioinformatics*, 15(Suppl 1):S5, 2014.

[6] Benno Schwikowski, Peter Uetz, and Stanley Fields. A network of protein–protein interactions in yeast. *Nature biotechnology*, 18(12):1257–1261, 2000.

[7] Joshua O'Madadhain, Jon Hutchins, and Padhraic Smyth. Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explorations Newsletter*, 7(2):23–30, 2005.

[8] Zan Huang, Xin Li, and Hsinchun Chen. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 141–142. ACM, 2005.

[9] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[10] David Liben-Nowell. *An algorithmic approach to social networks*. PhD thesis, Massachusetts Institute of Technology, 2005.

[11] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[12] Mohammad Al Hasan and Mohammed J Zaki. A survey of link prediction in social networks. In *Social network data analytics*, pages 243–275. Springer, 2011.

[13] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[14] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[15] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[16] Yanjun Qi, Ziv Bar-Joseph, and Judith Klein-Seetharaman. Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins: Structure, Function, and Bioinformatics*, 63(3):490–500, 2006.

[17] Milen Pavlov and Ryutaro Ichise. Finding experts by link prediction in co-authorship networks. *FEWS*, 290:42–55, 2007.

[18] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[19] Valdis E Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, 2002.

[20] Jennifer Xu and Hsinchun Chen. The topology of dark networks. *Communications of the ACM*, 51(10):58–65, 2008.

[21] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002.

[22] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, 2001.

[23] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[24] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

[25] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. Adaptive degree penalization for link prediction. *Journal of Computational Science*, 13:1–9, 2016.

[26] Srinivas Virinchi and Pabitra Mitra. Similarity measures for link prediction using power law degree distribution. In *Neural Information Processing*, pages 257–264. Springer, 2013.

[27] Jianpei Zhang, Yuan Zhang, Hailu Yang, and Jing Yang. A link prediction algorithm based on socialized semi-local information. *Journal of Computational Information Systems*, 10(10):4459–4466, 2014.

[28] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[29] Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet mathematics*, 1(2):226–251, 2004.

[30] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547579, 1901.

[31] Gerard Salton and Michael J McGill. *Introduction to modern information retrieval.* McGraw-Hill New York, 1983.

[32] Lieve Hamers, Yves Hemeryck, Guido Herweyers, Marc Janssen, Hans Keters, Ronald Rousseau, and André Vanhoutte. Similarity measures in scientometric research: the jaccard index versus salton's cosine formula. *Information Processing & Management*, 25(3):315–318, 1989.

[33] Thorvald Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. skr.*, 5:1–34, 1948.

[34] Bruce McCune, James B Grace, and Dean L Urban. *Analysis of ecological communities*, volume 28. MjM software design Gleneden Beach, Oregon, 2002.

[35] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *science*, 297(5586):1551–1555, 2002.

[36] EA Leicht, Petter Holme, and Mark EJ Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[37] Yuxiao Dong, Qing Ke, Bai Wang, and Bin Wu. Link prediction based on local information. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 382–386. IEEE, 2011.

[38] Fei Tan, Yongxiang Xia, and Boyao Zhu. Link prediction in complex networks: A mutual information perspective. *PloS one*, 9(9):e107056, 2014.

[39] Zhen Liu, Qian-Ming Zhang, Linyuan Lü, and Tao Zhou. Link prediction in complex networks: A local naïve bayes model. *EPL (Europhysics Letters)*, 96(4):48007, 2011.

[40] Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi. From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific Reports*, 3, 2013.

[41] Hon Nian Chua, Wing-Kin Sung, and Limsoon Wong. Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. *Bioinformatics*, 22(13):1623–1630, 2006.

[42] Guimei Liu, Jinyan Li, and Limsoon Wong. Assessing and predicting protein interactions using both local and global network topological metrics. *Genome Informatics*, 21:138–149, 2008.

[43] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[44] Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5):58007, 2010.

[45] Karl Pearson. The problem of the random walk. *Nature*, 72(1865):294, 1905.

[46] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 613–622, Washington, DC, USA, 2006. IEEE Computer Society.

[47] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[48] Oron Vanunu and Roded Sharan. A propagation-based algorithm for inferring gene-disease assocations. In *German Conference on Bioinformatics*, pages 54–52. Citeseer, 2008.

[49] Z Burda, J Duda, JM Luck, and B Waclaw. Localization of the maximal entropy random walk. *Physical review letters*, 102(16):160602, 2009.

[50] Rong-Hua Li, Jeffrey Xu Yu, and Jianquan Liu. Link prediction: the power of maximal entropy random walk. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1147–1156. ACM, 2011.

[51] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[52] Daniel Spielman. Spectral graph theory. *Lecture Notes, Yale University*, pages 740–0776, 2009.

[53] Francois Fouss, Alain Pirotte, J-M Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Knowledge and Data Engineering, IEEE Transactions on*, 19(3):355–369, 2007.

[54] Pavel Chebotarev and Elena Shamis. Matrix-forest theorems. *arXiv preprint math/0602575*, 2006.

[55] Vincent D Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM review*, 46(4):647–666, 2004.

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

[57] Alexis Papadimitriou, Panagiotis Symeonidis, and Yannis Manolopoulos. Fast and accurate link prediction in social networking systems. *Journal of Systems and Software*, 85(9):2119–2132, 2012.

[58] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252. ACM, 2010.

[59] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, et al. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic acids research*, 31(9):2443–2450, 2003.

[60] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.

[61] Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. What's in a crowd? analysis of face-to-face behavioral networks. *Journal of theoretical biology*, 271(1):166–180, 2011.

[62] Valdis Krebs. A network of books about recent us politics sold by the online bookseller amazon. com. *Unpublished http://www. orgnet. com*, 2008.

[63] Hamsterster full network dataset – KONECT, jun 2014.

[64] Vladimir Batagelj and Andrej Mrvar. Pajek datasets. *Web page http://vlado. fmf. uni-lj. si/pub/networks/data*, 2006.

[65] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[66] Anna Goldenberg, Alice X Zheng, Stephen E Fienberg, and Edoardo M Airoldi. A survey of statistical network models. *Foundations and Trends in Machine Learning*, 2(2):129–233, 2010.

[67] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003.

[68] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[69] Charles J Geyer. Practical markov chain monte carlo. *Statistical Science*, 7(4):473–483, 1992.

[70] Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.

[71] Zan Huang. Link prediction based on graph topology: The predictive value of the generalized clustering coefficient. In *Workshop on Link Analysis (KDD)*, 2006.

[72] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 322–331. IEEE, 2007.

[73] Toon Calders and Bart Goethals. Depth-first non-derivable itemset mining. In *SDM*, pages 250–261. SIAM, 2005.

[74] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[75] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM06: Workshop on Link Analysis, Counter-terrorism and Security*, 2006.

[76] William Cukierski, Benjamin Hamner, and Bo Yang. Graph-based features for supervised link prediction. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1237–1244. IEEE, 2011.

[77] Catherine A Bliss, Morgan R Frank, Christopher M Danforth, and Peter Sheridan Dodds. An evolutionary algorithm approach to link prediction in dynamic social networks. *Journal of Computational Science*, 5(5):750–764, 2014.

[78] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[79] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[80] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2011.

[81] Jérôme Kunegis and Andreas Lommatzsch. Learning spectral graph transformations for link prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 561–568. ACM, 2009.

[82] James Demmel and William Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific and Statistical Computing*, 11(5):873–912, 1990.

[83] Linyuan Lü and Tao Zhou. Link prediction in weighted networks: The role of weak ties. *EPL (Europhysics Letters)*, 89(1):18001, 2010.

[84] Yizhou Sun, Jiawei Han, Charu C Aggarwal, and Nitesh V Chawla. When will it happen?: relationship prediction in heterogeneous information networks. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 663–672. ACM, 2012.

[85] Yunpeng Zhao, Elizaveta Levina, and Ji Zhu. Link prediction for partially observed networks. *arXiv preprint arXiv:1301.7047*, 2013.

[86] Panagiotis Symeonidis and Eleftherios Tiakas. Transitive node similarity: predicting and recommending links in signed social networks. *World Wide Web*, 17(4):743–776, 2014.

[87] Tomasz Tylenda, Ralitsa Angelova, and Srikanta Bedathur. Towards time-aware link prediction in evolving social networks. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis*, page 9. ACM, 2009.

[88] Zhengdong Lu, Berkant Savas, Wei Tang, and Inderjit S Dhillon. Supervised link prediction using multiple sources. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 923–928. IEEE, 2010.

Supplementary material

Evaluation methodology

A systematic evaluation methodology is required in order to evaluate the quality of the predictions performed by a particular link prediction technique. Link prediction methods require a set of links to be used as a priori information to measure the similarity between unconnected nodes. Thus, a training set E^T and a validation or test set E^V are required in order to evaluate the performance of a method for a specific network. Following the classical supervised learning validation methodology, the training set is used by link prediction methods to infer the occurrence of links contained in the validation set.

If different snapshots of the network can be obtained, the observed links in time t can be used as training set, E^T, and new observed links at time $t + \Delta$ can be used as validation set, E^V. Both sets have to satisfy $E^T \cup E^V = E$ and $E^T \cap E^V = \emptyset$. A link belonging to the test set E^V is usually referred as a missing link. Let U_G be the graph of size $|V|$ containing the $\frac{|V||V-1|}{2}$ possible links that would exist if the network were complete. A non-observed link is a link in the set $U_G - E^T$. Finally, a non-existent link is a link from the set $U_G - E$.

Since, in most scenarios, only a single snapshot of the network can be accessed, a cross-validation process can be performed to build the training and the validation sets. Cross-validation is a validation technique that comprises different rounds of evaluation with complementary subsets. The most used approach in link prediction is k-fold cross-validation where the original set of links E is partitioned in k subsets of equal size. Only one of these sets is retained as validation set E^V while the rest of sets are joined and used as training set E^T. This process is repeated k times, using each subset as validation set only once.

Given the output set E^O of links predicted as existing and the validation set E^V, different performance measures can be used to evaluate the quality of the obtained results. In k-fold cross-validation, the measures obtained in each iteration are usually averaged to obtain a single global evaluation score.

There is a large collection of well-studied measures defined to evaluate the performance of supervised learning methods. As in any binary classification problem, we can define a confusion matrix or contingency table comparing the predicted class with the actual class. We have four different situations for a potential link $e_{x,y}$:

- True positive link (TP): If $e_{x,y} \in E^O$ and $e_{x,y} \in E^V$.
- False positive link (FP): If $e_{x,y} \in E^O$ and $e_{x,y} \notin E^V$.
- False negative link (FN): If $e_{x,y} \notin E^O$ and $e_{x,y} \in E^V$.
- True negative link (TN): If $e_{x,y} \notin E^O$ and $e_{x,y} \notin E^V$.

Given the situation for each link in $U - E^T$, different measures can be computed to evaluate the predictions performed by any link prediction technique. The most widely applied measures are described in the supplementary material.

Link prediction results can be assessed using different evaluation metrics, which are described below. One of the most commonly used metric is precision, which is defined as the fraction of true positive links among the set of links predicted as positive ones. It is computed as

$$precision = \frac{TP}{TP + FP}$$

Other common evaluation metric is the sensitivity, also known as recall or true positive rate, which is the ratio between the true positive links among the predicted links and the actual number of positive links. It can be calculated as

$$sensitivity = \frac{TP}{TP + FN} = \frac{TP}{P}$$

Sensitivity can be seen as the probability of an actually positive link to be predicted. Obtaining maximal sensitivity is trivial in some validation contexts by predicting all as positive elements, at the cost of a lower precision. When evaluating ranking-based techniques with a threshold selected to obtain $|E^O| = |E^V|$, this measure is redundant if the precision is given.

The specificity is other measure also known as the true negative rate. It measures the fraction of negative links which are correctly identified as such. It is calculated as

$$specificity = \frac{TN}{TN + FP} = \frac{TN}{N}$$

Given the precision, this measure is also redundant when the conditions described for sensitivity are satisfied.

On the one hand, precision and sensitivity are focused only on the true positive links. On the other hand, specificity focuses only on the true negative links. A different measure, called accuracy, takes into account both ratios. This measure is defined as the fraction of successfully predicted links. It can be written as

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

If $|E^O|$ is set so that $|E^O| = |E^V|$, this measure is also redundant given the precision. However, in actual environments, where $|E^V|$ is unknown, this measure is very informative.

Precision and sensitivity are basic measures used to evaluate supervised learning. It is important to achieve a good balance between both, which can be negatively correlated in some contexts such as information retrieval. The F-score, also known as F-measure, was proposed to combine precision and sensitivity in a single analytical score. Maximizing this score allows to maximize both scores maintaining an equilibrium controlled by a β parameter. The F-score considers precision and sensitivity by computing their weighted harmonic mean:

$$F_\beta = (1 + \beta^2) \frac{\text{precision} \times \text{sensitivity}}{(\beta^2 \times \text{precision}) + \text{sensitivity}}$$

where the β parameter defines the desired balance between precision and sensitivity. The F1-score is defined as F_β with $\beta = 1$, giving precision and sensitivity the same importance. The F2-score, with $\beta = 2$, emphasizes precision over sensitivity. On the other hand, the F0.5-score, with $\beta = 0.5$, emphasizes sensitivity over precision. Despite the fact that this measure does not consider the true negative rate, it has been widely used in information retrieval and machine learning.

Finally, a receiver operating characteristic curve, better known as a ROC curve, is a graphical plot that illustrates the performance of a binary classifier by plotting the sensitivity (true positive rate) as a function of 1-specificity (false positive rate) [1]. Each point in a ROC curve represents a sensitivity versus 1-specificity pair corresponding to a particular decision threshold.

The area under the ROC curve is also called AUC. This value ranges from 0 to 1 and summarizes the classifier performance. In the link prediction context, the AUC value is equal to the probability of the classifier ranking a random missing link (a link from the validation set E^V) better than a random non-existent link (a link from $U_G - E$). This value can be asymptotically approximated by a random sampling of pairs of missing and non-existent links as

$$AUC = \frac{n' + 0.5n''}{n}$$

where n is the number of sampled pairs of links, n' the number of pairs where the missing link was ranked better than the non-existent link and n'' the number of pairs where both links were ranked equally (for example, by obtaining the same score or probability of existence). As expected, the AUC value for a random classifier must approach to 0.5 as the number of sampled pairs is increased.

Network metrics

For each network used in our experiments, we computed different topological properties (see Table 1), including their number of nodes, their number of links, their average node degree, their average path length, and their diameter. We also include their clustering coefficient [3], which measures the tendency of a node to be linked with neighbors of its neighbors forming triangles. We computed the average clustering coefficient of the network. The clustering coefficient of each node, ranging from 0 to 1, is computed as

$$C_x = \frac{|\{e_{y,z} : y \in \Gamma_x, z \in \Gamma_x, e_{y,z} \in E\}|}{|\Gamma_x|(|\Gamma_x| - 1)}$$

Other measure we considered is heterogeneity, which is related to the node degree distribution of the network [4]. Heterogeneity measures the variance of node degrees in the network. A higher value for heterogeneity represents a larger number of high-degree nodes compared to the number of low-degree nodes. This value can be computed as

$$H = \frac{\langle \Gamma^2 \rangle}{\langle \Gamma \rangle^2} = \frac{\frac{1}{|V|}\sum_{x \in V} |\Gamma_x|^2}{(\frac{1}{|V|}\sum_{x \in V} |\Gamma_x|)^2}$$

Feature	**YST**	**CEL**	**INF**	**BCK**	**HMT**	**USA**	**NSC**
$\|V\|$	2284	297	410	105	2426	332	1461
$\|E\|$	6646	2148	2765	441	16630	2126	2742
$\langle k \rangle$	5.8196	14.4646	13.4878	8.4000	13.7098	12.8072	3.7536
C	0.1345	0.2924	0.4558	0.4875	0.5375	0.6252	0.6937
C*	0.2001	0.3079	0.4672	0.4875	0.6145	0.7494	0.8782
H	2.8479	1.8008	1.3876	1.4207	3.1011	3.4639	1.8486
APL	4.2942	2.4553	3.6309	3.0788	3.1473	2.7381	2.5937
D	11	5	9	7	10	6	17
r	−0.0991	−0.1632	0.2258	−0.1279	0.0474	−0.2079	0.4616

Table 1: The structural features of the networks used in our experiments. From top to bottom: number of nodes, number of links, average degree of each node, average clustering coefficient, average clustering coefficient of nodes with degree above one, heterogeneity, average shortest path length, diameter of the network and average degree assortativity.

Finally, the assortativity coefficient, or assortative mixing, was also taken into account. Assortativity measures the preference of nodes in a network to attach to other similar ones [2] according to a given similarity definition. We consider assortativity using node degrees. Degree assortativity is equivalent to the correlation coefficient among the degrees of every pair of connected nodes. The degree assortativity score is computed as

$$\begin{aligned} r &= corr_{e_{x,y}\in E}(|\Gamma_x|, |\Gamma_y|) \\ &= \frac{\frac{1}{|E|}\sum_{e_{x,y}\in E}|\Gamma_x||\Gamma_y| - [\frac{1}{|E|}\sum_{e_{x,y}\in E}(|\Gamma_x|+|\Gamma_y|)/2]^2}{\frac{1}{|E|}\sum_{e_{x,y}\in E}(|\Gamma_x|^2+|\Gamma_y|^2)/2 - [\frac{1}{|E|}\sum_{e_{x,y}\in E}(|\Gamma_x|+|\Gamma_y|)/2]^2} \end{aligned}$$

References

[1] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31:1–38, 2004.

[2] Mark EJ Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

[3] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.

[4] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

1.2 Adaptive link prediction

The journal paper associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. Adaptive degree penalization for link prediction. Journal of Computational Science 13:1-9, 2016. DOI 10.1016/j.jocs.2015.12.003.

- Status: **Published**
- ISSN: 1877-7503
- Impact Factor (JCR 2016): 1.748
- Subject Category:
  - Computer Science, Theory & Methods. Ranking 38/104.
  - Computer Science, Interdisciplinary Applications. Ranking 58/105.

Adaptive Degree Penalization for Link Prediction

Víctor Martínez Fernando Berzal Juan-Carlos Cubero

Abstract

Many systems of interest are best described using networks that represent binary relationships among their elements. Link prediction aims to infer the link formation process by predicting missed or future relationships based on currently-observed connections. Different techniques and measures have been proposed in the literature to solve this problem. Similarity-based local methods achieve high precision with a low computational complexity. However, determining which particular technique should be applied for each particular network remains an open question. In this paper, we exploit the existence of a relationship between the best-performing degree of penalization for shared neighbors and the network clustering coefficient. We propose an Adaptive Degree Penalization link prediction method, a novel link prediction technique that achieves better results than previously-proposed methods.

1 Introduction

The link prediction problem consists of inferring the formation of new relationships or the existence of still-unknown connections between pairs of entities in a network based on their properties and currently-observed links [1]. This problem has attracted a lot of attention, since a large number of systems in many different fields can be described using networks. Approaches and techniques to solve this problem allow us to extract implicit information present in the network, identify spurious links, or model and evaluate network evolution mechanisms. These problems are of great interest since they are closely related to other problems usually found in different disciplines. For example, link prediction has been used to predict previously unknown protein interactions in protein-protein interaction networks [2]. It has also been used to study and predict future author collaborations and tendencies in co-authorship networks [3]. In fact, link prediction is present in our daily lives when we get friendship suggestions in social networks [4] or recommendations of new products in e-commerce web sites [5].

The link prediction problem is formally defined as follows. Let G be an undirected graph $G = (V, E)$, where V is a set of optionally-labeled nodes and E is a set of edges (also referred to as links) between pairs of elements from set V. Given a snapshot of the network G at time t, the link prediction problem consists of inferring the subset of missing links in the current snapshot that will be formed at time $t + \Delta$.

Some notational conventions are important to properly describe the proposed solutions for the link prediction problem. An edge between nodes x and y is denoted as $e_{x,y}$. The number of nodes in the network is $|V|$. The number of edges is $|E|$. The set of nodes connected through an edge to a node x is called the neighborhood of x and is referred to as Γ_x. The degree of a node x in an undirected graph is defined as the number of edges connected to the node and will be denoted as $|\Gamma_x|$.

Many link prediction methods are based on the observation that nodes that share a higher number of neighbors are more likely to be connected [6]. Well-known link prediction techniques take into account the number of directly shared neighbors (local methods) or the number of chains of neighbors between two nodes (global methods) to estimate the probability of the existence of a potential link. However, these techniques always work the same way, regardless of the network they are applied to. Our work is motivated by the lack of further studies about how link prediction techniques are affected by network structural properties and how existing methods can be adapted to the structural properties of particular networks in order to obtain better results.

This paper is organized as follows. Related work is presented in Section 2. We propose and describe a generalized degree penalization similarity measure in Section 3. In Section 4, we analyze the relationship of the best-performing degree penalization with respect to the topological properties of the network. A novel link prediction technique called Adaptive Degree Penalization is presented in Section 5. Finally, the conclusions drawn from this study and some lines of future research are presented in Section 6.

2 Related work

Link prediction has been the subject of many studies [7]. A large number of techniques following different approaches have been proposed to deal with the link prediction problem [8]. In this work, we limit our scope to techniques that consider only network topology, albeit methods considering other attributes have also been proposed [9].

The first and most studied approach is based on the similarity between nodes [1, 8]. Similarity-based techniques assume that nodes are more likely to form links with similar nodes. A function that assigns a similarity score $s(x, y)$ to every pair of nodes in the network is defined. This similarity score can take into account different features, which can be topological properties or network-specific attributes. All possible pairs of nodes are ranked in decreasing order based on their similarity scores. Links at the top of the ranked list are supposed to be more likely to be present in the set of missing links.

Similarity-based methods can be categorized depending on the amount of information taken into consideration when computing the similarity function. For example, local similarity techniques consider only direct neighbor information. This family of techniques can achieve high precision in most networks and have a linear time complexity, which makes them suitable for large networks. On the other hand, global methods use the whole topology of the network to compute the similarity score for every possible link. This type of techniques has the advantage of being able to compute the similarity between each pair of nodes regardless of their distance within the network, instead of being limited to neighbor-sharing pairs of nodes. Their main drawbacks are their high computational complexity and their sensitivity to noise, which usually leads to lower precision than local methods. Finally, quasi-local techniques have been proposed to try to find an equilibrium between the amount of considered information and the computational complexity of the resulting methods. Most quasi-local techniques are either based on local ones with small variations to consider neighbors of neighbors or based on global ones with constraints on the lengths of the considered paths.

An alternative approach is to describe the network formation model in statistical terms. Statistical approaches build a parameterized model assuming the existence of a known structure in the network [10, 11, 12, 13]. The parameters of the model for a particular network are estimated using statistical methods. Finally, the adjusted model is used to compute the probability of the formation of each possible link. The main problem of this kind of techniques is that they suffer from a very high computational cost, which limits their applicability to networks of only hundreds or a few thousand nodes. In addition, they can only be applied to networks with a particular structure.

Other algorithmic approaches have also been proposed. Since link prediction techniques are inherently heuristic, some metaheuristic-based methods have been proposed in order to automatically adjust the influence of a set of local similarity-based techniques in an attempt to maximize precision [14]. The link prediction problem can be seen as a classification problem with two classes (existence and absence of links). This point of view allows the application of traditional machine learning techniques [15, 16]. These techniques can obtain better results than other approaches at the cost of a previous training stage, which is not always possible in many applications. Furthermore, they have the drawback that the predictive model they build is often hard to understand and analyze.

In this paper, we focus our attention on local similarity-based techniques, since these techniques are widely used due to their high scalability and the reasonable precision they obtain [17]. Computational complexity is really important in link prediction, since most real networks are huge, with hundreds of thousands or even millions of nodes. Even worse, there are usually time or resource constraints in many problems related to link prediction. In fact, most recommender systems use only local techniques.

The most basic local method is called Common Neighbors (CN). This technique assigns a score based just on the number of shared neighbors:

$$s^{CN}(x, y) = |\Gamma_x \cap \Gamma_y| \tag{1}$$

It makes sense to assume that, if two individuals share many acquaintances, they are more likely to meet than two individuals without common contacts. Different studies have confirmed this hypothesis by observing a correlation between the number of shared neighbors between pairs of nodes and their probability of being linked [6].

Lada Adamic and Eytan Adar proposed the Adamic-Adar Index (AA) to measure the similarity between two entities based on their shared features [18]. This measure was adapted to link prediction by considering shared neighbors as features:

$$s^{AA}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{\log |\Gamma_z|} \tag{2}$$

This equation is a variation of the common neighbors similarity function. Here, each shared neighbor is penalized by its degree. This intuitively makes sense in a large number of real-world networks. For example, in social networks, the amount of resources or time that a node can spend on each of its neighbors decreases as its degree increases, also decreasing its influence on them.

The Resource Allocation Index (RA) was motivated by the resource allocation process which takes place in complex distribution networks [17]. It models the transmission of resources between two unconnected nodes x and y through neighborhood nodes. Each neighborhood node gets a given amount of resources and distributes them evenly among its neighbors. The amount of resources obtained from node x by node y through their shared neighbors can be considered as a similarity measure between both nodes. The resource allocation index has shown to be the local measure with better results in a large number of networks [19]. It can be computed as

$$s^{RA}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{|\Gamma_z|} \tag{3}$$

Other local similarity-based techniques have been proposed, including the Preferential Attachment Index [20], the Jaccard Index [21], the Salton Index [22], the Sørensen Index [23], the Hub-Promoted and Hub-Depressed Indices [24], and the Leicht-Holme-Newman Index [25]. Most of these techniques are variations of the previously-described measures, but also consider other features such as the number of unshared neighbors. Different comparative studies have shown that these variations work better in very specific contexts, yet are worse on average [17].

3 Similarity based on adjustable degree penalization

The Common Neighbors method, the Adamic-Adar Index, and the Resource Allocation Index have been presented in the literature as three different link prediction techniques. It can be readily seen that these methods assume that the probability of existence of a link between two nodes is proportional to the number of shared neighbors between them, but penalize each one according to their degree, with a null penalization in the Common Neighbors case. From our point of view, CN, AA, and RA are just variations of the same technique, which considers shared neighbors and penalizes them by their degree using different penalization schemes.

In addition, these techniques limit the degree of penalization to fixed values without taking into account that the most suitable penalization degree could be different according to the peculiarities of the network under study.

A generalized expression for the existing degree penalization local link prediction techniques that allows us to specify the desired degree penalization level can be stated as

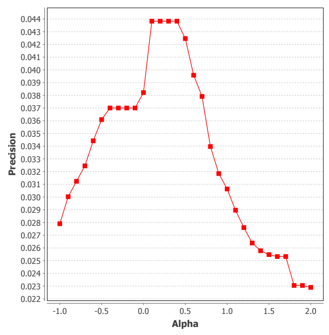
$$s^{GDP}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} |\Gamma_z|^{-\alpha} \tag{4}$$

where α is a free parameter to adjust the penalization to each particular network and $|\Gamma_z|$ is the degree of the shared neighbor z. It can be seen that this Generalized Degree Penalization (GDP) expression is equivalent to the Common Neighbors method when $\alpha = 0$, when no penalization is performed. Furthermore, this expression is equal to the Resource Allocation Index when $\alpha = 1$. Finally, the behavior of the Adamic-Adar Index can be closely approximated by setting α to a value between 0 and 1. For example, we obtained $\alpha \approx 0.37$ after fitting the function $1/x^\alpha$ to the function $1/\log x$ in the interval $[2, 20]$, which encompasses the expected degree values for most nodes in typical real-world networks.

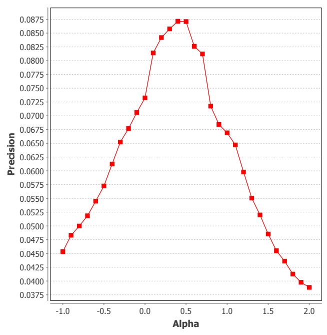
Our definition of local similarity offers two main benefits. First, it allows us to unify the analysis of three existing measures instead of having three different related methods. Second, existing measures do not obtain optimal results since the optimal value of the α parameter is not adjusted by existing techniques.

Which one of those three variations works better in practice depends upon the network they are applied to. However, no advances in determining which technique is better have been accomplished. A complete empirical evaluation is typically done on a case-by-case basis. The ideal degree penalization scheme varies among networks, since each of the aforementioned local techniques obtains better results than the others depending on the network under analysis. Since those techniques only rely on topological properties, the optimal α value should be determined by the network structure.

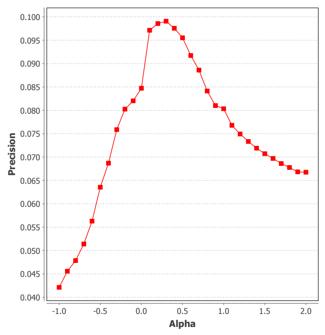
In order to understand how different values of α behave in different networks with different properties, we tested the α parameter for a reasonable range of values and plotted the precision and the AUC obtained for each value, as shown in Figure 1 and Figure 2, respectively. The best-performing degree penalization was estimated for a collection of networks, which we describe in the appendix, by applying the Generalized Degree Penalization technique to each network and varying α in a range from -1.0 to 2.0 in steps of size 0.1. We measured the obtained precision and AUC for each network by performing a 5-fold cross-validation experiment as described in the appendix.



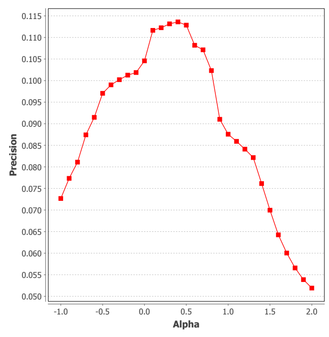
(a) UPG



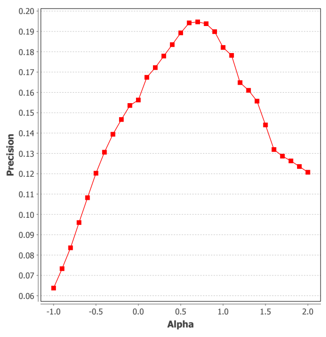
(b) HPD



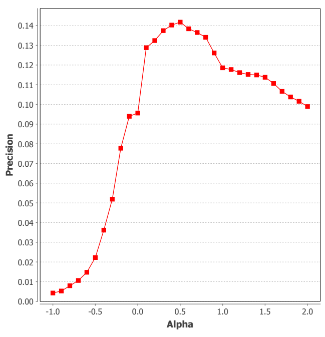
(c) ERD



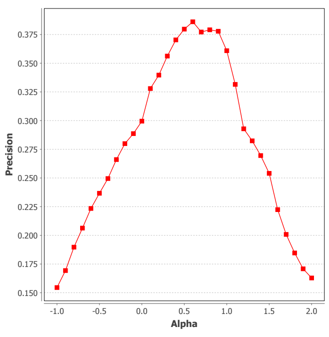
(d) YST



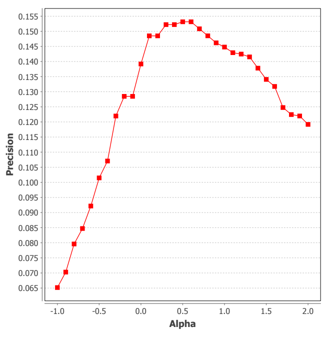
(e) ADV



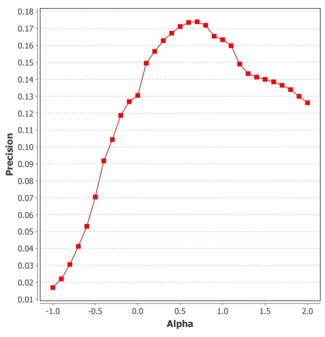
(f) KHN



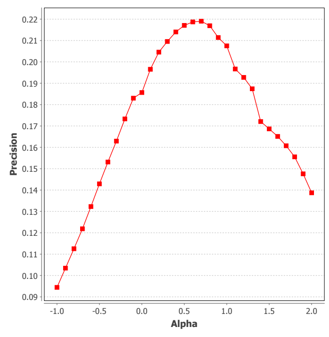
(g) PGP



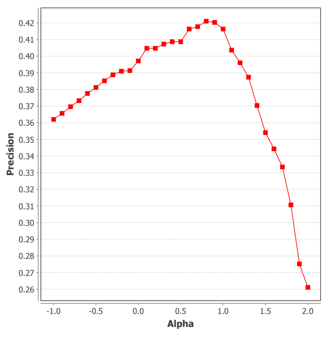
(h) CEG



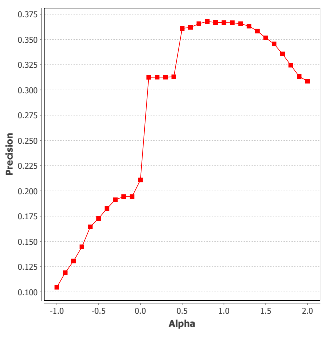
(i) LDG



(j) ZWL

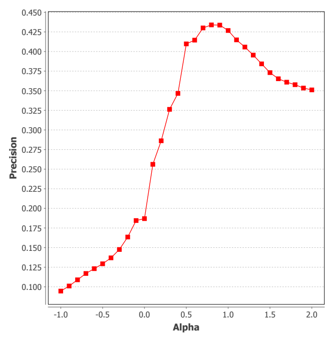


(k) INF

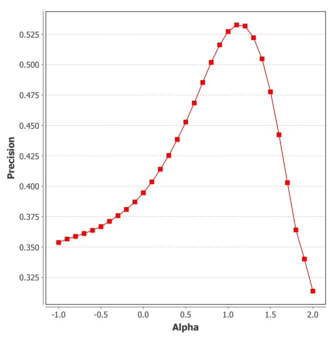


(l) HTC

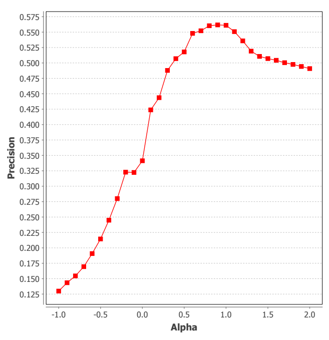
Our experiments show that the best-performing α value can reasonably vary for different networks. It can be observed that the α values, fixed by CN, AA, and RA, are not always even near to the best-performing α. Hence, a technique to adapt the α parameter to the network would be desirable. As far as we know, there are no previous studies about the best-performing degree penalization relationship to structural network properties. The only step that has been taken in this direction is the observation that a higher performance can be obtained by increasing or decreasing the degree penalization depending on the node degree [26]. However, [26] suffers from the same limitations as CN, AA, and RA, since the particular values used for degree penalization are also fixed for every network.



(m) CGS

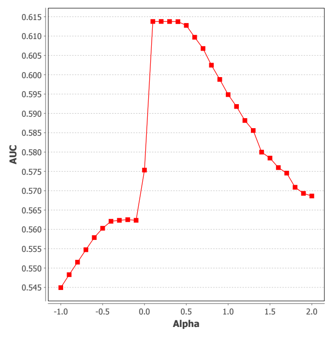


(n) FBK

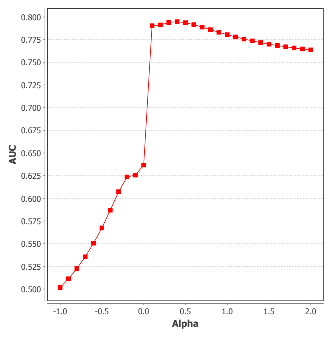


(o) CDM

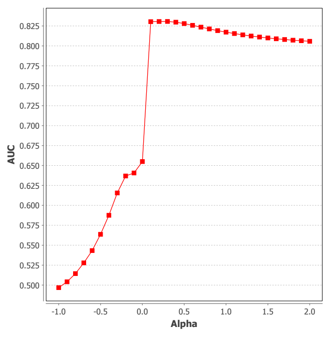
Figure 1: Precision obtained by our link predictor varying the alpha parameter for different networks (see appendix for a description of the networks used in our experiments).



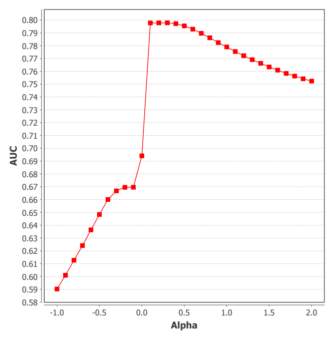
(a) UPG



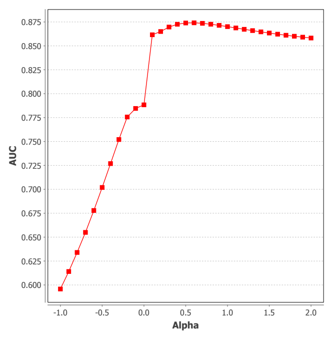
(b) HPD



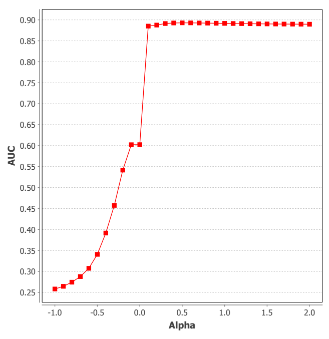
(c) ERD



(d) YST



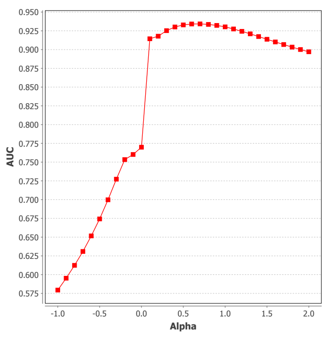
(e) ADV



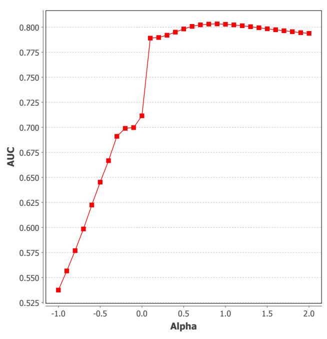
(f) KHN

4 Relating degree penalization to the network structural properties

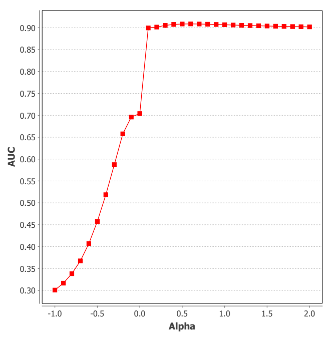
The best-performing α value for each network might depend on the network structure. Intuitively, you might conjecture that some topological properties of the network might be related to this value. Guided by this conjecture, we carried out an experiment to find the degree of correlation between the best-performing value of α and different quantifiable



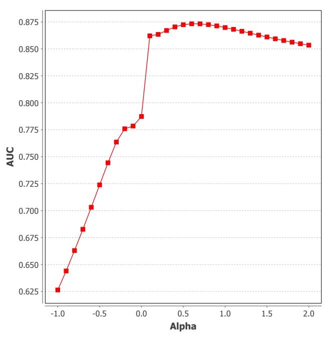
(g) PGP



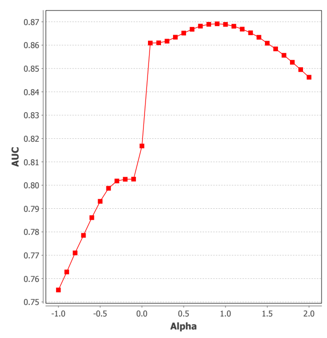
(h) CEG



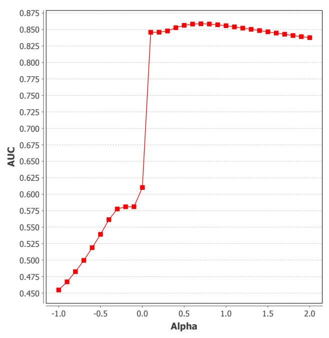
(i) LDG



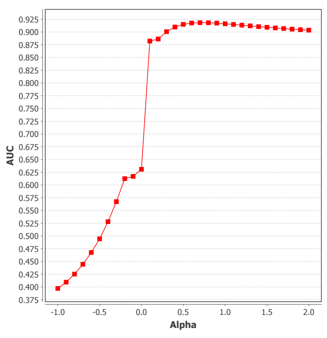
(j) ZWL



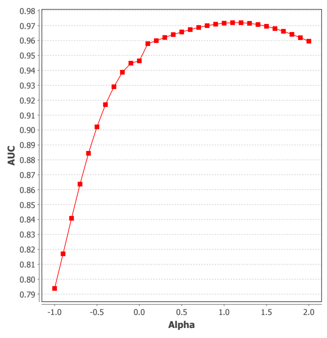
(k) INF



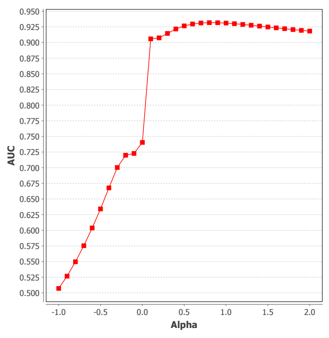
(l) HTC



(m) CGS



(n) FBK



(o) CDM

Figure 2: AUC obtained by our link predictor varying the alpha parameter for different networks (see appendix for a description of the networks used in our experiments).

global topological properties of networks, whose results can be found below.

We took the best-performing α value for each network and computed the Pearson correlation coefficient against different network properties to find potential correlations between the best-performing α value and the network structural properties. The value of each network topological property was computed for each training network generated in the 5-fold cross validation process used in our experiments, as described in the appendix. The obtained correlation coefficients are shown in Table 1.

Property	**Coefficients for precision**	**Coefficients for AUC**
$\|\mathbf{V}\|$	0.1477 (1.0)	−0.0553 (1.0)
$\|\mathbf{E}\|$	0.5980 (0.2966)	0.4288 (1.0)
$\langle\mathbf{k}\rangle$	0.6041 (0.2731)	0.6498 (0.1397)
C	$0.9374\ (4.06 * 10^{-6})$	0.8812 (0.0002)
ASPL	−0.4990 (0.9325)	−0.4205 (1.0)
D	−0.4203 (1.0)	−0.3758 (1.0)
H	−0.3425 (1.0)	−0.4250 (1.0)
r	0.4975 (0.9466)	0.4289 (1.0)

Table 1: Correlation coefficients (and Bonferroni-adjusted p-values) between network structural properties and the best-performing alpha value according to precision and AUC in our link prediction experiments. Properties, from top to bottom: number of nodes *(*$|V|$*)*, number of edges *(*$|E|$*)*, average degree *(*$\langle k\rangle$*)*, average clustering coefficient *(C)*, average shortest path length *(ASPL)*, diameter *(D)*, heterogeneity *(H)*, and assortativity *(r)*.

As expected, some structural network properties are slightly correlated to the best performing-degree penalization value. For example, the assortativity is weakly correlated to this value. This implies that the best-performing α tends to be higher as the nodes of the network tend to be connected to similar ones in terms of their degree. On the other hand, the average shortest path length shows a negative correlation. This suggests that the best-performing α tends to increase as the length of the shortest paths between nodes decreases. However, these correlations are weak and do not help us predict the α value that should be used to improve precision or AUC in link prediction.

On the other hand, the obtained results show that the average clustering coefficient of a network and the best-performing α are strongly correlated ($r = 0.9374$ for precision and $r = 0.8812$ for AUC). The average clustering coefficients have been plotted against the best-performing alpha values for precision in Figure 3 and for AUC in Figure 4. It can be seen that they follow an almost linear correlation. This implies that the best-performing α value can be estimated with a reasonable accuracy by attending just to the average clustering coefficient measured in the network.

As far as we know, this relationship has not been previously mentioned nor documented. The discovered relationship is coherent with previous results and allows us to explain why the Common Neighbors method usually obtains better results in low-clustered networks and the Resource Allocation method tends to obtain better results in highly-clustered ones.

5 Adaptive Degree Penalization

The generalized degree penalization measure, combined with the observation that the degree penalization that obtains better results in link prediction can be estimated by considering the clustering coefficient of the network, allows us to propose a new link prediction technique that automatically adapts to the network.

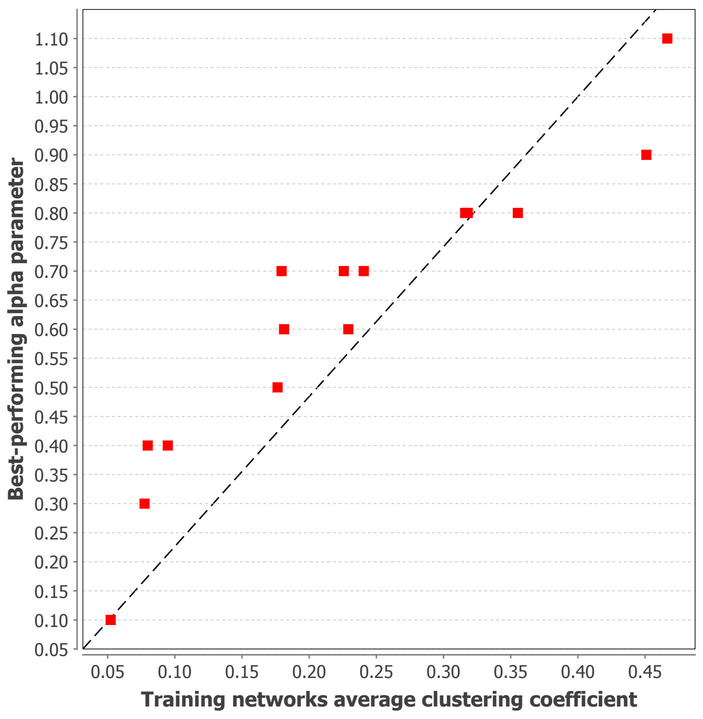


Figure 3: Average clustering coefficient shown against the best-performing alpha value according to precision for the networks used in our experiments.

We propose an Adaptive Degree Penalization link prediction technique whose definition of local similarity tries to estimate the best-performing degree penalization by using the average clustering coefficient observed in the network. We define our similarity measure as

$$s^{ADP}(x,y) = \sum_{z \in \Gamma_x \cap \Gamma_y} |\Gamma_z|^{-\beta C} \tag{5}$$

where C is the average clustering coefficient of the network and β is a constant. The average clustering coefficient only has to be measured once before applying our link prediction algorithm. In really large or very dynamic networks, it could be estimated by a sampling procedure. The β constant is determined beforehand using an heterogeneous set of networks, since all the networks we have tested seem to follow the same correlation pattern between the clustering coefficient and the best-performing degree penalization.

In order to test the performance of the proposed technique, we have carried out an experiment using an estimated β value. To determine this value, we performed a linear regression between the clustering coefficients of a set of networks, which we call training networks, and the best-performing alpha values. We obtained an slope of $\beta = 2.52$ for precision and a slope of $\beta = 2.47$ for AUC. Since the slope is consistent for both measures, we set the estimated slope to be $\beta = 2.5$.

Using this value for the β parameter, we drove a 5-fold cross validation experiment to determine the precision and AUC of the proposed link prediction method. This

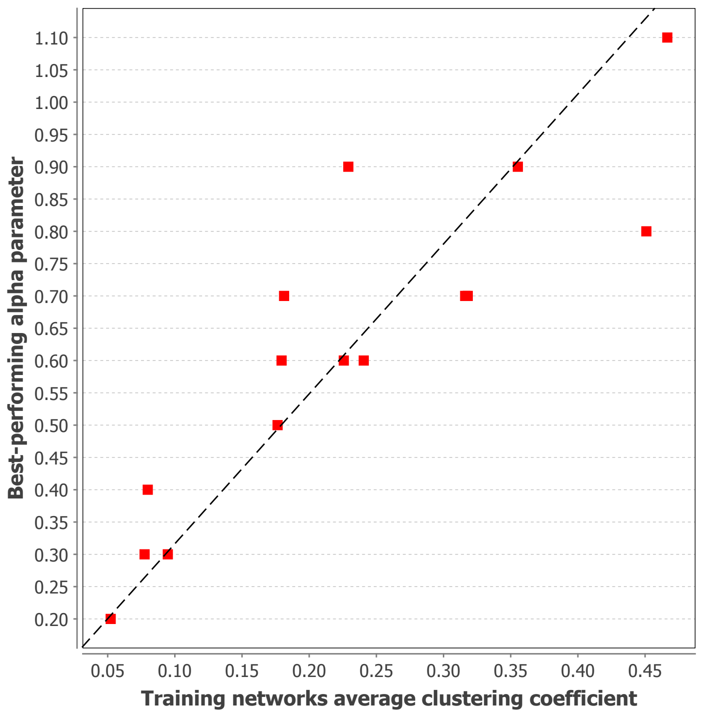


Figure 4: Average clustering coefficient shown against the best-performing alpha value according to AUC for the networks used in our experiments.

experimentation is carried out for the training set (networks used to determine the optimal β) and for a test set (networks not involved in the β estimation). We compared our method precision and AUC against the results obtained by the Common Neighbors method, the Adamic-Adar Index, and the Resource Allocation method. In addition, we also compared it with the best precision and AUC obtained by varying the α value. It must be taken into account that, since the best precision has been estimated with a step resolution of size 0.1, our β-based technique could achieve slightly better results than the estimated best-performing α. In fact, the precision obtained by Adaptive Degree Penalization is even higher than that determined by varying the α parameter in the FBK network, and the AUC is higher in the SMG and BUP networks.

The results that we obtained in our experiments are shown in Table 2 for precision and Table 3 for AUC. It can be seen how our Adaptive Degree Penalization method obtained the best results for precision in 18 of the 22 network when compared to CN, AA, and RA (13 from training set and 5 from test set). With respect to AUC, our method obtained the best results for 18 of the 22 networks (12 from training set and 6 from test set). In most of the remaining cases, it still stands as the second best method. On average, our method obtains a higher precision of 0.0082 than the Resource Allocation method, the best of the previously-proposed local techniques. This improvement may seem small. However, it can be seen that the best precision based on varying α for each network is only 0.0026 better than the average precision obtained by our ADP method. Considering AUC, ADP is also better than the best existing method, RA in this case, and almost indistinguishable from

	Network	$\hat{C}$	**Estimated** α	**Best** α	**Best precision**	**CN**	**AA**	**RA**	**ADP**
Training networks	**UPG**	0.05	0.13	0.1	0.0438	0.0411	0.0331	0.0306	**0.0438**
	HPD	0.08	0.20	0.4	0.0872	0.0748	0.0828	0.0669	**0.0842**
	ERD	0.08	0.19	0.3	0.0991	0.0883	0.0950	0.0803	**0.0986**
	YST	0.09	0.24	0.4	0.1136	0.1103	0.1080	0.0876	**0.1127**
	ADV	0.18	0.45	0.7	0.1947	0.1591	0.1783	0.1821	**0.1862**
	KHN	0.18	0.44	0.5	0.1418	0.1085	0.1382	0.1185	**0.1407**
	PGP	0.18	0.45	0.6	0.3861	0.3058	0.3655	0.3608	**0.3761**
	CEG	0.23	0.57	0.6	0.1532	0.1401	**0.1532**	0.1448	0.1527
	LDG	0.23	0.56	0.7	0.1740	0.1361	0.1662	0.1634	**0.1727**
	ZWL	0.24	0.60	0.7	0.2190	0.1891	0.2110	0.2075	**0.2188**
	INF	0.36	0.89	0.8	0.4210	0.3978	0.4080	0.4163	**0.4192**
	HTC	0.32	0.80	0.8	0.3679	0.2406	0.3583	0.3667	**0.3673**
	CGS	0.32	0.79	0.8	0.4339	0.2037	0.3986	0.4265	**0.4337**
	FBK	0.47	1.17	1.1	0.5327	0.3946	0.4185	0.5272	**0.5334**
	CDM	0.45	1.13	0.9	0.5617	0.3878	0.5338	**0.5611**	0.5466
Test networks	**EML**	0.16	0.41	0.6	0.1990	0.1825	0.1923	0.1802	**0.1967**
	SMG	0.22	0.55	0.4	0.1611	0.1420	0.1587	0.1408	**0.1593**
	BUP	0.37	0.94	0.7	0.2608	0.2449	**0.2608**	0.2563	0.2540
	GRQ	0.36	0.90	0.8	0.5550	0.4002	0.5308	0.5531	**0.5539**
	HMT	0.40	1.00	0.9	0.3977	0.2580	0.3267	**0.3959**	**0.3959**
	UAL	0.46	1.15	1.0	0.5160	0.4388	0.4558	**0.5160**	0.5155
	NSC	0.47	1.18	1.2	0.6667	0.5058	0.6295	0.6659	**0.6667**
				Training set average	0.2620	0.1985	0.2432	0.2494	**0.2591**
				Test set average	0.3938	0.3103	0.3649	0.3869	**0.3917**
				Overall average	0.3039	0.2341	0.2820	0.2931	**0.3013**

Table 2: Results obtained from our method comparison. Columns, from left to right: network name, average clustering coefficient of the training network, estimated α by $\beta\hat{C}$, best-performing α value from experiment (see Figure 1), precision obtained by the best-performing α, Common Neighbors precision, Adamic-Adar Index precision, Resource Allocation precision, and Adaptive Degree Penalization precision.

the best result obtained by testing for multiple values of α.

We performed a Friedman test [27] to determine if there are statistically significant differences among the link prediction methods used in our experiments. The Friedman tests confirm that there are significant differences for precision and AUC, since the obtained p-values are 2.5571×10^{-8} and 1.1921×10^{-11}, respectively. The average ranks obtained in our experiments by the different methods for precision were: 1.36 for CN, 2.55 for AA, 2.34 for RA, 3.75 for our method (ADP). For AUC, the ranks were 1.00 for CN, 2.36 for AA, 2.84 for RA, 3.80 for ADP.

Finally, we performed a post-hoc test using the Wilcoxon signed-rank test [28] with our link prediction method as control. The alpha level (initially set to 0.05) was adjusted with the Holm method to control the familywise error rate (FWER). According to precision, when we compared our method to CN, AA, and RA, we obtained 8.8219×10^{-4}, 0.0034, and 0.0198 as corrected p-values, respectively. According to AUC, the corrected p-values were 8.8219×10^{-4}, 0.0020, and 0.0461. In other words, our method obtains significantly better

	Network	$\hat{C}$	**Estimated** α	**Best** α	**Best AUC**	**CN**	**AA**	**RA**	**ADP**
Training networks	**UPG**	0.05	0.13	0.2	0.6138	0.5882	0.6036	0.5944	**0.6137**
	HPD	0.08	0.20	0.4	0.7948	0.7080	**0.7941**	0.7805	0.7912
	ERD	0.08	0.19	0.3	0.8307	0.7355	0.8305	0.8172	**0.8306**
	YST	0.09	0.24	0.3	0.7978	0.7336	0.7961	0.7790	**0.7978**
	ADV	0.18	0.45	0.6	0.8741	0.8231	0.8683	0.8701	**0.8734**
	KHN	0.18	0.44	0.5	0.8929	0.7436	0.8909	0.8916	**0.8928**
	PGP	0.18	0.45	0.7	0.9342	0.8373	0.9288	0.9303	**0.9318**
	CEG	0.23	0.57	0.9	0.8033	0.7443	0.7931	**0.8029**	0.8002
	LDG	0.23	0.56	0.6	0.9088	0.7978	0.9053	0.9069	**0.9088**
	ZWL	0.24	0.60	0.6	0.8733	0.8202	0.8677	0.8698	**0.8733**
	INF	0.36	0.89	0.9	0.8692	0.8315	0.8642	0.8689	**0.8691**
	HTC	0.32	0.80	0.7	0.8587	0.7134	0.8579	0.8558	**0.8584**
	CGS	0.32	0.79	0.7	0.9184	0.7495	0.9123	0.9162	**0.9182**
	FBK	0.47	1.17	1.1	0.9720	0.9514	0.9615	0.9716	**0.9720**
	CDM	0.45	1.13	0.8	0.9318	0.8143	0.9243	**0.9311**	0.9298
Test networks	**EML**	0.16	0.41	0.5	0.7976	0.7561	0.7970	0.7885	**0.7974**
	SMG	0.22	0.55	0.5	0.8331	0.7535	0.8296	0.8290	**0.8332**
	BUP	0.37	0.94	1.1	0.7486	0.6977	0.7395	0.7486	**0.7489**
	GRQ	0.36	0.90	0.9	0.9259	0.8073	0.9171	0.9257	**0.9259**
	HMT	0.40	1.00	0.9	0.9115	0.8417	0.8954	**0.9112**	**0.9112**
	UAL	0.46	1.15	1.1	0.9313	0.8741	0.9141	0.9310	**0.9313**
	NSC	0.47	1.18	1.0	0.9374	0.8037	0.9327	**0.9374**	0.9366
			Training set average		0.8583	0.7728	0.8532	0.8524	**0.8574**
			Test set average		0.8693	0.7906	0.8608	0.8673	**0.8692**
			Overall average		0.8618	0.7785	0.8556	0.8572	**0.8612**

Table 3: Results obtained from our method comparison. Columns, from left to right: network name, average clustering coefficient of the training network, estimated α by $\beta\hat{C}$, best-performing α value from experiment (see Figure 2), AUC obtained by the best-performing α, Common Neighbors AUC, Adamic-Adar Index AUC, Resource Allocation AUC, and Adaptive Degree Penalization AUC.

results than previous methods. In addition, we obtained a corrected p-value of 0.0055 for precision and 0.0801 for AUC when we compared our adaptive degree penalization method to the method that tests many different α values. This result indicates that the test failed to reject the null hypothesis for AUC, so there is no evidence to suggest that the AUC obtained by our adaptive degree penalization method is significantly different from the precision obtained by an exhaustive search of possible values for the α penalization parameter.

6 Conclusions and future work

In this paper, we have presented a generalized adaptive degree penalization link prediction method that unifies previously-proposed local similarity-based techniques. We have also studied how the penalization parameter relates to some network structural properties, finding a strong correlation between the clustering coefficient and the best-performing value of the degree penalization parameter. This result allowed us to propose a new technique that automatically estimates this parameter based on the clustering coefficient measured in the network under study. Our new method obtains statistically significant better results than CN, AA, and RA in an experimental evaluation considering 22 networks from different application domains.

These results lead to a better understanding of local similarity-based techniques. Instead of considering different measures with different theoretical backgrounds, we consider them as instances of a more general technique. In addition, our observation that a higher degree penalization should be performed as the average clustering coefficient of the network increases is consistent with known results.

A further theoretical study of these empirical results remains as future work. A pure theoretical study could lead to interesting results. Some studies have started to try to understand the behavior of the clustering coefficient in graphs generated from network models [29, 30]. Connecting network formation and link prediction theory could help improve the results obtained by link prediction techniques.

Acknowledgments

Work partially supported by the Spanish Ministry of Economy and the European Regional Development Fund (FEDER), under grant TIN2012-36951, and partially by the Ministry of Education of Spain under the program "Ayudas para contratos predoctorales para la formación de doctores 2013" (grant BES-2013-064699).

Appendix

Network structural measures

Different structural network measures were considered to summarize network topologies. These measures help us understand networks at a macroscopic level. Different measures related to shortest paths were taken into account. For example, we computed the average shortest path length (ASPL) considering every possible pair of nodes. In addition, we also obtained the network diameter (D), which is the length of the longest shortest path in the network.

We also computed the clustering coefficient [31], which measures the tendency of a node to cluster with other nodes forming triangles. More intuitively, in a social network, the clustering coefficient would measure the tendency of the friends of a given person to be also friends among themselves. We computed the average clustering coefficient of the network as

$$\overline{C} = \frac{\sum_{x \in V} C_x}{|V|} \tag{6}$$

where C_x is the clustering coefficient, ranging from 0 to 1, of a node x, which is itself computed as

$$C_x = \frac{|\{e_{y,z} : y \in \Gamma_x, z \in \Gamma_x, e_{y,z} \in E\}|}{|\Gamma_x|(|\Gamma_x| - 1)}. \tag{7}$$

Heterogeneity is another interesting measure that is related to the node degree distribution of the network [17]. It measures the variance of node degrees. In a typical real-world network, a higher value for heterogeneity represents a larger number of high-degree nodes compared to the number of low-degree nodes. This value can be computed as

$$H = \frac{\langle \Gamma^2 \rangle}{\langle \Gamma \rangle^2} = \frac{\frac{1}{|V|} \sum_{x \in V} |\Gamma_x|^2}{(\frac{1}{|V|} \sum_{x \in V} |\Gamma_x|)^2}. \tag{8}$$

The assortativity coefficient, or assortative mixing, is a measure that assesses the preference of nodes in a network to attach to other similar ones [32]. The similarity definition can vary but it is usually computed using degree similarity between nodes as the correlation coefficient among the degrees of every pair of connected nodes. Considering the degree similarity, this score is computed as

$$\begin{aligned} r &= corr_{e_{x,y} \in E}(|\Gamma_x|, |\Gamma_y|) \\ &= \frac{\frac{1}{|E|} \sum_{e_{x,y} \in E} |\Gamma_x||\Gamma_y| - [\frac{1}{|E|} \sum_{e_{x,y} \in E} (|\Gamma_x| + |\Gamma_y|)/2]^2}{\frac{1}{|E|} \sum_{e_{x,y} \in E} (|\Gamma_x|^2 + |\Gamma_y|^2)/2 - [\frac{1}{|E|} \sum_{e_{x,y} \in E} (|\Gamma_x| + |\Gamma_y|)/2]^2}. \end{aligned} \tag{9}$$

Data sources

We have collected 22 networks from different sources and domains. These networks were carefully selected to cover a wide range of properties, including different sizes, average degrees, clustering coefficients, and heterogeneity indices. A summary of the collection of networks we used in our experiments can be found in Table 4. The collection of networks used in our experiments can be found at `http://noesis.ikor.org/datasets/link-prediction`. UPG is a power distribution network. HPD, YST, and CEG are biological networks. ERD, KNH, LDG, SMG, ZWL, HTC, CGS, CDM, NSC, and GRQ are co-authorship networks for different fields of study. HMT, FBK, and ADV are social networks. UAL is an airport traffic network. EML is a network of individuals who shared emails. PGP is an interaction network of users of the Pretty Good Privacy algorithm. BUP is a network of political blogs. Finally, INF is a network of face-to-face contacts in an exhibition.

Name	|V|	|E|	⟨k⟩	C	ASPL	D	H	r	Reference
UPG	4941	6594	2.67	0.08	18.99	46	1.4504	0.0035	[31]
HPD	8756	32331	7.38	0.11	4.19	14	4.5133	−0.051	[33]
ERD	6927	11850	3.42	0.12	3.78	4	12.6708	−0.1156	[34]
YST	2284	6646	5.82	0.13	4.29	11	2.8479	−0.0991	[35]
EML	1133	5451	9.62	0.22	3.61	8	1.9421	0.0782	[35]
ADV	5155	39285	15.24	0.25	3.22	9	5.4060	−0.0951	[36]
KHN	3772	12718	6.74	0.25	3.63	12	9.422	−0.1205	[34]
PGP	10680	24316	4.55	0.27	7.49	24	4.1465	0.2382	[37]
CEG	297	2148	14.46	0.29	2.46	5	1.8008	−0.1632	[31]
LDG	8324	41532	9.98	0.31	4.37	16	6.188	−0.0997	[34]
SMG	1024	4916	9.6	0.31	2.98	6	3.9475	−0.1925	[34]
ZWL	6651	54182	16.29	0.32	3.85	10	2.5851	0.0006	[34]
INF	410	2765	13.49	0.46	3.63	9	1.3876	0.2258	[38]
BUP	105	441	8.4	0.49	3.08	7	1.4207	−0.1279	[39]
HTC	7610	15751	4.14	0.49	5.68	19	2.0986	0.2939	[40]
CGS	6158	11898	3.86	0.49	3.62	14	3.9467	0.2426	[34]
GRQ	5241	14484	5.53	0.53	5.05	17	3.0523	0.6593	[41]
HMT	2426	16630	13.71	0.54	3.15	10	3.1011	0.0474	[42]
FBK	4024	87887	43.68	0.59	3.98	13	2.432	0.0707	[43]
UAL	332	2126	12.81	0.63	2.74	6	3.4639	−0.2079	[34]
CDM	16264	47594	5.85	0.64	5.82	18	2.2087	0.1846	[40]
NSC	1461	2742	3.75	0.69	2.59	17	1.8486	0.4616	[44]

Table 4: Network topological properties (with networks sorted by their clustering coefficient). Columns, from left to right: network name, number of nodes *($|V|$)*, number of edges *($|E|$)*, average degree *($\langle k \rangle$)*, average clustering coefficient *(C)*, average shortest path length *(ASPL)*, diameter *(D)*, heterogeneity *(H)*, and assortativity *(r)*.

Evaluating link prediction methods

A 5-fold cross-validation was carried out to measure the performance of link prediction techniques. Link prediction methods require a set of links to be used as a priori information to perform the inference process.

The original set of links E of each network was partitioned into k non-overlapping subsets $\{E_1, \ldots, E_k\}$ of equal size. On the ith iteration, only one of these sets was retained as validation set $E^V = E_i$ while the remaining sets were joined and used as training set $E^T = E - E_i$. Hence, $E^T \cup E^V = E$ and $E^T \cap E^V = \emptyset$. This process was repeated k times in order to use each subset once as validation set. The performance scores computed during each iteration were averaged to obtain a single score.

A missing link is formally defined as a link belonging to test set E^T. A non-observed link is defined as a link in the set $U_G - E^T$, where U_G is the complete graph of size $|V|$ containing the $\frac{|V||V-1|}{2}$ possible links that would exist if the network were fully connected.

Finally, a non-existent link is defined as a link from the set $U_G - E$.

The results obtained by a supervised machine learning algorithm can be summarized using a confusion matrix. A confusion matrix shows all possible actual and predicted class combinations. In the context of link prediction, a true positive (TP) is a predicted link that actually belongs to the test set, a false positive (FP) is a predicted link that does not belong to the validation set, a false negative (FN) is a non-predicted link that belongs to the test set, and a true negative (TN) is a non-predicted link that does not belong to the validation set. These values are used to define different scores that measure different desirable properties in a link predictor.

We used precision as the performance measure for link predictors. Precision, also known as positive predictive value, is defined as the fraction of true positive links among the set of links predicted as true:

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

Other measures have been used to evaluate the performance of link prediction techniques. For example, some authors have used the area under the curve (AUC) to compare different methods [8]. The AUC value is equal to the probability of the technique ranking a random missing link (a link from the validation set E^V) better than a random non-existent link (a link from $U_G - E$). Considering all pairs of missing and non-existent links, this value can be computed as

$$AUC = \frac{n' + 0.5n''}{n}$$

where n is the number of pairs, n' the number of pairs where the missing link was ranked better than the non-existent link and n'' the number of pairs where both links were ranked equally (for example, by obtaining the same score or probability of existence). As expected, the AUC value for a random classifier must be around 0.5.

Recall has also been used by some authors [45]. Recall is similar to precision but considers the number of false negatives instead of the number of false positives. However, due to the way we evaluate link prediction techniques, it can be seen that FN increases as FP increases, so distinguishing between precision and recall would be unnecessary in our context (as is the use of alternative measures such as the F-score).

References

[1] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[2] Víctor Martínez, Carlos Cano, and Armando Blanco. Prophnet: A generic prioritization method through propagation of information. *BMC bioinformatics*, 15(Suppl 1):S5, 2014.

[3] Milen Pavlov and Ryutaro Ichise. Finding experts by link prediction in co-authorship networks. *FEWS*, 290:42–55, 2007.

[4] Michael Fire, Lena Tenenboim, Ofrit Lesser, Rami Puzis, Lior Rokach, and Yuval Elovici. Link prediction in social networks using computationally efficient topological features. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*, pages 73–80. IEEE, 2011.

[5] Zan Huang, Xin Li, and Hsinchun Chen. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 141–142. ACM, 2005.

[6] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, 2001.

[7] Lise Getoor and Christopher P Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[8] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[9] Neil Zhenqiang Gong, Ameet Talwalkar, Lester Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Elaine Runting Shi, and Dawn Song. Joint link prediction and attribute inference using a social-attribute network. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(2):27, 2014.

[10] Zan Huang. Link prediction based on graph topology: The predictive value of the generalized clustering coefficient. In *Workshop on Link Analysis (KDD)*, 2006.

[11] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 322–331. IEEE, 2007.

[12] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[13] Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.

[14] Catherine A. Bliss, Morgan R. Frank, Christopher M. Danforth, and Peter Sheridan Dodds. An evolutionary algorithm approach to link prediction in dynamic social networks. *Journal of Computational Science*, 5(5):750 – 764, 2014.

[15] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[16] William Cukierski, Benjamin Hamner, and Bo Yang. Graph-based features for supervised link prediction. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1237–1244. IEEE, 2011.

[17] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B-Condensed Matter and Complex Systems*, 71(4):623–630, 2009.

[18] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[19] Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5):58007, 2010.

[20] Albert László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[21] Paul Jaccard. tude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Socit Vaudoise des Sciences Naturelles*, 37:547579, 1901.

[22] Gerard Salton and Michael J McGill. *Introduction to modern information retrieval.* McGraw-Hill New York, 1983.

[23] Thorvald Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. skr.*, 5:1–34, 1948.

[24] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *science*, 297(5586):1551–1555, 2002.

[25] EA Leicht, Petter Holme, and Mark EJ Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[26] Srinivas Virinchi and Pabitra Mitra. Similarity measures for link prediction using power law degree distribution. In *Neural Information Processing*, pages 257–264. Springer, 2013.

[27] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[28] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, pages 80–83, 1945.

[29] N. Eggemann and S.D. Noble. The clustering coefficient of a scale-free random graph. *Discrete Applied Mathematics*, 159(10):953 – 965, 2011.

[30] Pol Colomer-de Simon and Marián Boguná. Clustering of random scale-free networks. *Physical Review E*, 86(2):026120, 2012.

[31] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.

[32] Mark EJ Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

[33] Suraj Peri, J Daniel Navarro, Ramars Amanchy, Troels Z Kristiansen, Chandra Kiran Jonnalagadda, Vineeth Surendranath, Vidya Niranjan, Babylakshmi Muthusamy, TKB Gandhi, Mads Gronborg, et al. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome research*, 13(10):2363–2371, 2003.

[34] Vladimir Batagelj and Andrej Mrvar. Pajek datasets. *Web page http://vlado. fmf. uni-lj. si/pub/networks/data*, 2006.

[35] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, et al. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic acids research*, 31(9):2443–2450, 2003.

[36] Paolo Massa, Martino Salvetti, and Danilo Tomasoni. Bowling alone and trust decline in social network sites. In *Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on*, pages 658–663. IEEE, 2009.

[37] María Angeles Serrano, Marian Boguñá, Romualdo Pastor-Satorras, and Alejandro Vespignani. Correlations in complex networks. *Large scale structure and dynamics of complex networks: From information technology to finance and natural sciences*, pages 35–66, 2007.

[38] Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. What's in a crowd? analysis of face-to-face behavioral networks. *Journal of theoretical biology*, 271(1):166–180, 2011.

[39] Valdis Krebs. A network of books about recent us politics sold by the online bookseller amazon. com. *Unpublished http://www. orgnet. com*, 2008.

[40] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

[41] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[42] Jérôme Kunegis. Hamsterster full network dataset – KONECT. *Web page http://konect.uni-koblenz.de/networks/petster-hamster*, jun 2014.

[43] Julian J McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *NIPS*, volume 272, pages 548–556, 2012.

[44] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[45] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM06: Workshop on Link Analysis, Counter-terrorism and Security*, 2006.

1.3 Probabilistic local link prediction

The book chapter associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. Probabilistic local link prediction in complex networks. In Proceedings of the 11th International Conference on Scalable Uncertainty Management. Lecture Notes in Computer Science, 10564:391-396, 2017. DOI 10.1007/978-3-319-67582-4.

- Status: **Published**
- ISSN: 0302-9743
- ISBN: 978-3-319-67581-7
- Volume: 10564

Probabilistic Local Link Prediction in Complex Networks

Víctor Martínez Fernando Berzal Juan-Carlos Cubero

Abstract

Link prediction is the problem of inferring future or missing relationships between nodes in a given network. This problem has attracted great attention since it has a large number of applications. In this problem, there is always some degree of uncertainty because the absence of a link between a pair of nodes may be the result of the non-existence of the link or the result of it not being observed but actually existing. In this paper, we propose a local link prediction technique that aggregates the observed evidence to estimate the probability of each possible non-observed link. We also show how our scalable link prediction technique achieves higher precision than other well-established local techniques in several networks from very different domains.

1 Introduction

A problem that commonly appears in a large number of domains is, given a set of observed relationships or interactions between entities, predicting the most likely non-observed links. This task is known as the link prediction problem [1]. It has been applied with great success to a large number of tasks, including the prediction of interactions among proteins [2], the prediction of future author collaborations [3], the suggestion of people we may know in social networks [1], or the recommendation of commercial products [4].

Very different approaches have been proposed to deal with the link prediction problem [5]. In this work, we focus our attention on local techniques, which take into account only local information, leading to highly scalable algorithms. Efficiency is paramount because link prediction is commonly applied to massive networks, where scalability is a crucial requirement. Almost all local techniques consider the shared neighbors between nodes to estimate the likelihood of the existence of a link, weighting the contribution of each shared neighbor according to certain feature (such as the degree of the shared node). However, different networks may require different weightings for each shared neighbor contribution, instead of the fixed weighting most local techniques use. To estimate the contribution of each shared neighbor, we must take into account the uncertainty that is present due to the fact that an unobserved link may indicate the non-existence or the actual existence of the link.

We propose a novel local link prediction technique that takes the features of the current network into account and weighs the contribution of each node according to the expected contribution for nodes of its degree. Despite our technique requiring sampling the whole network to estimate the required parameters, this process can be done efficiently and changes in the network can be handled locally. We propose a probabilistic framework where the degree of belief on the existence of a link is increased as more evidence is accumulated in terms of shared neighbors.

This paper is organized as follows. Our proposal is described in full detail in Section 2. An empirical evaluation and the results we obtained are discussed in Section 3. Finally, conclusions extracted from our work are presented in Section 4.

2 Method

Let us assume that we have access to a snapshot of a complex network, observing links between nodes. L_{xy} denotes the event corresponding to the existence of a link between nodes x and y. Γ_x denotes the set of neighbors of a node x and $\Gamma_{x\cap y}$ denotes the set of shared neighbors of a pair of nodes x and y.

According to Bayes' theorem, we can express the complementary probability of a link between nodes x and y given the set of shared neighbors Γ_{xy} as

$$P(\overline{L_{xy}}|\Gamma_{x\cap y}) = \frac{P(\Gamma_{x\cap y}|\overline{L_{xy}})P(\overline{L_{xy}})}{P(\Gamma_{x\cap y})}.$$

Like most local link prediction techniques, we assume independence among shared neighbors, thus we can rewrite the previous expression as

$$P(\overline{L_{xy}}|\Gamma_{x\cap y}) = \prod_{z\in\Gamma_{x\cap y}} \frac{P(z|\overline{L_{xy}})}{P(z)} P(\overline{L_{xy}}).$$

According to Bayes' theorem, the term $\frac{P(z|\overline{L_{xy}})}{P(z)}$ is equal to the term $\frac{P(\overline{L_{xy}}|z)}{P(\overline{L_{xy}})}$ and, after applying this substitution, we obtain the expression

$$P(\overline{L_{xy}}|\Gamma_{x\cap y}) = \prod_{z\in\Gamma_{x\cap y}} \frac{P(\overline{L_{xy}}|z)}{P(\overline{L_{xy}})} P(\overline{L_{xy}}).$$

According to basic probability axioms, the probabilities of an event and of its complementary event always total 1; thus, we can express the previous equation in terms of the probability of the existence of links as

$$P(L_{xy}|\Gamma_{x\cap y}) = 1 - \prod_{z\in\Gamma_{x\cap y}} \frac{1-P(L_{xy}|z)}{1-P(L_{xy})}(1-P(L_{xy})),$$

where $P(L_{xy}|z)$ is the probability of the existence of a link between x and y given the shared neighbor z. This value can be estimated using different methods. In this work, we propose the estimation of this value as the probability of the existence of a link given a shared neighbor of the same degree than z in the network, which is computed as

$$P(L_{xy}|z) = \frac{1}{|N_{|\Gamma_z|}|} \sum_{k \in N_{|\Gamma_z|}} \frac{\sum_{i \neq j, i,j \in \Gamma_k} P(L_{ij})}{|\Gamma_k|(|\Gamma_k| - 1)},$$

where $N_{|\Gamma_z|}$ is the set of nodes with the same degree than z. It is important to note that this value has to be computed only once for each node degree value, rather than once for each shared neighbor.

For each pair of nodes x and y, we assume $P(L_{xy}) = 1$ if a link is currently observed. Since we are only interested in ranking links, the value of $P(L_{xy})$ when no link is observed is irrelevant for us provided that we keep it smaller than 1. Applications requiring a true probability estimation may need to estimate the prior probabilities of unobserved links.

In order to analyze the computational complexity of the proposed approach, we divide the analysis in two phases. In the first phase, the average probability of a link given a shared neighbor of a particular degree is computed. This phase has a computational complexity $O(vd^2)$, where v refers to the number of nodes in the network and d refers to the degree of these nodes. Once these values are computed, the computational complexity of the second phase, regarding the estimation of the likelihood of a link, is just $O(2d)$, since shared neighbors can be efficiently computed using hash sets. As we can see, our method is highly scalable, since the range of node degrees tends to be much smaller than the number of nodes in a complex network.

3 Evaluation and results

In order to measure the performance of our proposal, we performed a battery of tests by applying our technique to networks gathered from very different domains. We considered the following networks: a social network from a website called Advogato (ADV, [6]), a protein-protein interaction network in budding yeast (YST, [7]), a network of e-mail exchanges between university members (EML, [8]), the metabolic network of Caenorhabditis elegans (CEG, [9]), a human protein-protein interaction network (HPD, [10]), four citation networks (CGS, LDG, SMG, ZWL, [11]), and a power distribution network (UPG, [12]).

Our experimentation consisted of a 5-fold cross-validation where links in each network were randomly divided into five sets of the same size n. Considering each set as the test set, we used the four remaining sets as the training network to predict the most probable n links. As performance score, we used precision, which is computed as

$$precision = \frac{true\ positives}{true\ positives + false\ positives} = \frac{true\ positives}{n},$$

where a link instance is classified as positive if it is ranked within the top n links, and negative otherwise. Results from each run were averaged to obtain a single performance score for each method and network combination.

We compared our approach to some well-known similarity-based local link prediction techniques. Despite their simplicity, these techniques have shown to obtain good results in practice and they are usually considered to be reasonable choices for link prediction. The techniques included in our comparison are the following ones:

- **Common neighbors (CN):** In this method, the likelihood of the existence of a link between two nodes is proportional to the number of shared neighbors between both nodes [1]. This method was proposed after observing a correlation between the number of shared neighbors of two nodes and the probability that they will collaborate in the future in scientific collaboration networks. It is computed as

$$P(L_{xy}) \propto |\Gamma_x \cap \Gamma_y|.$$

- **The Adamic-Adar index (AA):** This method measures the likelihood of the link between two entities based on the logarithmically-penalized degree of each shared neighbor. This index weighs nodes with low degree much heavier than nodes with high degree, assuming that nodes with few neighbors are more likely to contribute to the formation of a link between a pair of nodes of their neighborhood [13]. It is computed as

$$P(L_{xy}) \propto \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{\log |\Gamma_z|}.$$

- **The resource allocation index (RA):** This method is similar to the Adamic-Adar index, yet considering the degree of each shared neighbor without a logarithmic penalization [14]. RA models the resource allocation process that takes place in complex networks, where two unconnected nodes exchange units of resources by equally distributing them among their neighbor nodes. The amount of exchanged resources between a pair of nodes can be viewed as a measure of similarity. It is defined as

$$P(L_{xy}) \propto \sum_{z \in \Gamma_x \cap \Gamma_y} \frac{1}{|\Gamma_z|}.$$

- **Local Naïve Bayes (LNB):** This method estimates the role or degree of influence of each shared neighbor using probability theory [15], computing the probability of the existence of a link between a pair of nodes as

$$P(L_{xy}) \propto \sum_{z \in \Gamma_x \cap \Gamma_y} f(z) \log (oR_z)$$

where o is a constant for the network computed as

$$o = \frac{p_{unconnected}}{p_{connected}} = \frac{\frac{1}{2}|V|(|V|-1)}{|E|} - 1$$

and R_z is the role or influence of the node computed as

$$R_z = \frac{2|\{e_{x,y} : x, y \in \Gamma_z, e_{x,y} \in E\}| + 1}{2|\{e_{x,y} : x, y \in \Gamma_z, e_{x,y} \notin E\}| + 1}.$$

The function $f(z)$ measures the influence of the shared neighbor. The authors suggest $f(z) = 1$ from common neighbors, $f(z) = \frac{1}{\log |\Gamma_z|}$ from the Adamic-Adar index, or $f(z) = \frac{1}{|\Gamma_z|}$ from the resource allocation method.

The results we obtained using each technique for each dataset are shown in Table 1. Our technique is listed as *PLLP*, an acronym for probabilistic local link prediction. It can be observed that our technique achieves a higher average precision than previous techniques for all the complex networks considered in our experimentation.

	ADV	**YST**	**EML**	**CEG**	**CGS**	**HPD**	**LDG**	**SMG**	**UPG**	**ZWL**	**Average**
CN	0.1489	0.0876	0.1304	0.1119	0.1810	0.0627	0.1113	0.1357	0.0296	0.1516	0.11507
AA	0.1783	0.1080	0.1923	0.1532	0.3985	0.0828	0.1662	0.1581	0.0165	0.2110	0.16649
RA	0.1821	0.0876	0.1800	0.1448	0.4178	0.0669	0.1633	0.1407	0.0132	0.2075	0.16039
LNB-CN	0.1717	0.1189	0.1912	**0.1569**	0.2507	0.0850	0.1529	0.1583	**0.0425**	0.2005	0.15286
LNB-AA	**0.1906**	**0.1190**	0.1972	0.1555	0.4068	0.0867	0.1723	**0.1597**	0.0176	0.2158	0.17212
LNB-RA	0.1874	0.0954	0.1778	0.1462	0.4186	0.0658	0.1615	0.1471	0.0150	0.2067	0.16215
PLLP	0.1875	0.1141	**0.1982**	0.1536	**0.4273**	**0.0874**	**0.1729**	0.1593	0.0347	**0.2167**	**0.17517**

Table 1: Precision obtained by each method (rows) for each dataset (columns). The best results are highlighted in bold.

4 Conclusions

In this paper, we have proposed a novel technique for link prediction. Our method aggregates local evidence to estimate the probability of an uncertain event such as the existence of a link. It works by increasing the degree of belief for each potential link by aggregating the evidence provided by each shared neighbor. Our proposal achieves better average precision results than some well-established local link prediction techniques for several networks from very different domains. Local Naïve Bayes outperforms our technique for certain networks under specific configurations. However, our method performs better in average without the need of testing parameters. Instead of including ad hoc parameters, our method achieves better results reasoning from first principles

In future work, we will explore and evaluate alternative approaches to the estimation of the probability of a link given a shared neighbor. Models to estimate the prior probability of links will also be studied, since they can be useful for those applications requiring true probability estimations.

Acknowledgments.

This work is partially supported by the Spanish Ministry of Economy and the European Regional Development Fund (FEDER), under grant TIN2012-36951 and the program "Ayudas para contratos predoctorales para la formación de doctores 2013" (grant BES-2013-064699).

References

[1] Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the Association for Information Science and Technology, 58(7), 1019–1031 (2007).

[2] Martínez, V., Cano, C., Blanco, A.: ProphNet: A generic prioritization method through propagation of information. BMC Bioinformatics, 15(1), S5 (2014).

[3] Pavlov, M., Ichise, R.: Finding experts by link prediction in coauthorship networks. FEWS 290, 42–55 (2007).

[4] Huang, Z., Li, X., Chen, H.: Link prediction approach to collaborative filtering. Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital libraries, ACM, 141–142 (2005).

[5] Martínez, V., Berzal, F., Cubero, J. C.: A Survey of Link Prediction in Complex Networks. ACM Computing Surveys, 49(4), 69 (2016).

[6] Massa, P., Salvetti, M., Tomasoni, D.: Bowling alone and trust decline in social network sites. In Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on (pp. 658-663). IEEE (2009).

[7] Bu, D., Zhao, Y., Cai, L., Xue, H., Zhu, X., Lu, H., et al.: Topological structure analysis of the protein-protein interaction network in budding yeast. Nucleic Acids Research, 31(9), 2443–2450 (2003).

[8] Guimera, R., Danon, L., Diaz–Guilera, A., Giralt, F., Arenas, A.: Self-similar community structure in a network of human interactions. Physical Review E, 68(6) (2003).

[9] Duch, J., Arenas, A.: Community detection in complex networks using extremal optimization. Physical Review E, 72(2) (2005).

[10] Peri, S., Navarro, J. D., Amanchy, R., Kristiansen, T. Z., Jonnalagadda, C. K., Surendranath, V., et al.: Development of human protein reference database as an initial platform for approaching systems biology in humans. Genome Research, 13(10), 2363–2371 (2003).

[11] Pajek datasets, `http://vlado.fmf.uni-lj.si/pub/networks/data`.

[12] Watts, D. J., Strogatz, S. H.: Collective dynamics of small-world networks. Nature, 393(6684), 440–442 (1998).

[13] Adamic, L. A., Adar, E.: Friends and neighbors on the web. Social Networks, 25(3), 211–230 (2003).

[14] Lü, L., Jin, C. H., Zhou, T.: Similarity index based on local paths for link prediction of complex networks. Physical Review E, 80(4) (2009).

[15] Liu, Z., Zhang, Q. M., L, L., Zhou, T.: Link prediction in complex networks: A local nave Bayes model. EPL (Europhysics Letters), 96(4) (2011).

2 Applications

This Section collects the publications related to the different applications of link prediction studied in this dissertation, including a journal paper proposing a technique for generic prioritization using heterogeneous data, the proposal of a novel approach for the disambiguation of semantic relations, and a technical report describing an automorphic distance metric for node role discovery.

2.1 Prioritization using heterogeneous data

The journal paper associated to this part of the dissertation is:

V. Martínez, C. Cano, A. Blanco. ProphNet: A generic prioritization method through propagation of information. BMC Bioinformatics 15(S-1):S5, 2014. DOI 10.1186/1471-2105-15-S1-S5.

- Status: **Published**
- ISSN: 1471-2105
- Impact Factor (JCR 2014): 2.576
- Subject Category:
  - Mathematical & Computational Biology. Ranking 10/57.
  - Biochemical Research Methods: Ranking 38/78.
  - Biotechnology & Applied Microbiology. Ranking 68/160.

ProphNet: A Generic Prioritization Method Through Propagation of Information

Víctor Martínez Carlos Cano Armando Blanco

Abstract

Background. Prioritization methods have become an useful tool for mining large amounts of data to suggest promising hypotheses in early research stages. Particularly, network-based prioritization tools use a network representation for the interactions between different biological entities to identify novel indirect relationships. However, current network-based prioritization tools are strongly tailored to specific domains of interest (e.g. gene-disease prioritization) and they do not allow to consider networks with more than two types of entities (e.g. genes and diseases). Therefore, the direct application of these methods to accomplish new prioritization tasks is limited.
Results. This work presents ProphNet, a generic network-based prioritization tool that allows to integrate an arbitrary number of interrelated biological entities to accomplish any prioritization task. We tested the performance of ProphNet in comparison with leading network-based prioritization methods, namely rcNet and DomainRBF, for gene-disease and domain-disease prioritization, respectively. The results obtained by ProphNet show a significant improvement in terms of sensitivity and specificity for both tasks. We also applied ProphNet to disease-gene prioritization on Alzheimer, Diabetes Mellitus Type 2 and Breast Cancer to validate the results and identify putative candidate genes involved in these diseases.
Conclusions. ProphNet works on top of any heterogeneous network by integrating information of different types of biological entities to rank entities of a specific type according to their degree of relationship with a query set of entities of another type. Our method works by propagating information across data networks and measuring the correlation between the propagated values for a query and a target sets of entities. ProphNet is available at: http://genome2.ugr.es/prophnet. A Matlab implementation of the algorithm is also available at the website.

1 Background

The advancements in high-throughput technologies such as DNA sequencing, linkage analysis, association studies and expression arrays have fostered the research towards an effective personalized medicine. To this end, the integration of pieces of evidence of different nature derived from diverse data sources is required, together with algorithms able to mine these data and identify novel biological facts of relevance.

Networks have been shown to be an useful representation for combining heterogeneous biological data. Currently, there is a huge availability of large molecular networks such as protein-protein interaction (PPI) networks, which model interactions between proteins.

Many methods have been proposed in the literature to represent and mine knowledge from biological networks [1]. For example, [2] proposes to apply text-mining in OMIM to generate a similarity network for human diseases and [3] builds a gene network based on the results of microarray experiments. These approaches have led to the emergence of new methods that exploit and integrate different data sources into networks in a variety of ways [4]. Inferring new knowledge from existent networks is usually based on "guilt-by-association" [5]. This extensively validated principle states that biological entities which are associated or interacting in a network are more likely to share a common function. This principle allows to infer new relationships from already known interactions.

In this context with massive amounts of highly interconnected data is where prioritization methods are required. Prioritization tools are based on computational approaches that use information retrieved from diverse sources in order to obtain ranked lists of candidate biological elements (genes, proteins, diseases, etc.) related with a certain target element. Gene-disease prioritization, in which genes are ranked according to their relevance to a disease of interest (or vice versa), is the most popular prioritization task, and many methods have been proposed in the recent literature to accomplish this task [6]. Most of these methods focus on the analysis of phenotype and PPI networks for gene-disease prioritization. These methods weight the arcs connecting two proteins or phenotypes according to a measure of the similarity between them. CIPHER [7] computes correlation coefficients based on linear regressions of phenotype and PPI profiles. PRINCE [8] computes the relevance of a gene by using network propagation methods. RWRH [9] scores genes and diseases using a random walk approach on PPI and phenotypes networks. rcNet [10] proposes a methodology for prioritization of candidate genes based on propagating node values and measuring the degree of correlation between two sets of nodes, one in the PPI/gene network and one in the phenotype network. Network-based gene-disease prioritization methods have been proven to provide better results than previous approaches [11–14].

Apart from gene-disease prioritization, other methods have been proposed to perform a prioritization of other biological entities. DomainRBF [15] performs a prioritization of protein domains for diseases using Bayesian linear regression. This method assumes a key role for protein domains in diseases as shown by previous studies [16]. Domains are basic structural and functional units of proteins, which in turn are composed of multiple structural domains, each one closely linked to a specific function. Although DomainRBF exploits the functional role of protein domains in phenotypes, it does not explore the simultaneous integration of PPI, domain and phenotype networks for gene or disease prioritization.

Despite the good performance obtained by the mentioned prioritization methods, they have clear limitations. First, existing network-based prioritization methods do not allow to consider more than two types of networks for performing the prioritization (e.g. gene and disease networks in rcNet and domain and disease networks in domainRBF). Only non-network-based methods have succeeded in integrating more than two different types of entities for prioritization. For example, Endeavour [13] performs an independent prioritization of different entities using multiple heterogeneous generic data sources which are integrated on a single global ranking using order statistics. However, previously mentioned network-based methods have been shown to outperform this method using a lower amount of data sources [7].

Second, existing prioritization methods are strongly tailored to a specific domain of interest (for example gene-disease prioritization for rcNet and protein domain-disease prioritization for domainRBF, respectively). In our opinion, these two drawbacks dramatically limit the applicability of these methods to other prioritization tasks and do not allow to improve the results by integrating information about other types of related entities.

In this work we present ProphNet, a generic prioritization method that outperforms previous methods by integrating and propagating information in an arbitrary number of heterogeneous data networks. Our method is generic since it allows to prioritize biological entities of any type with respect to biological entities of another type. Therefore, the user can customize the goal of the prioritization task (disease-gene, domain-disease, drug-disease, etc.). Furthermore, the user is not restricted to the use of only two entities, and can integrate as many biological networks as desired.

To compare the results obtained by ProphNet with those obtained by state-of-the-art methods, such as rcNet and domainRBF, we applied ProphNet to the prioritization of genes-diseases and domains-diseases, respectively, on a network built as the integration of protein domain, PPI and phenotype networks.

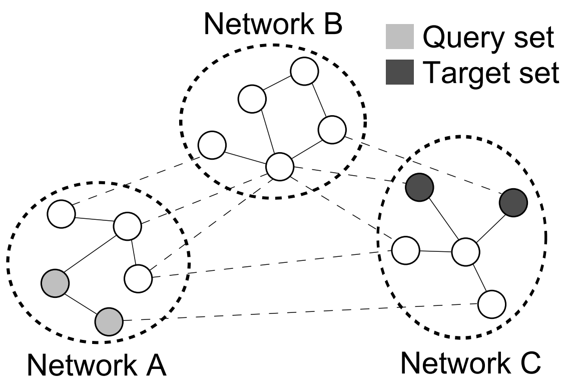


Figure 1: Our problem is to determine how related the query set and the target set are based on known relations between elements.

ProphNet measures the influence of a query set of biological entities of a certain type (e.g. genes or diseases) in a target set of entities of another type (e.g. diseases or genes, respectively). To this end, the algorithm uses a graph representation as shown in Figure 1. In this representation, each node corresponds to a biological entity of a domain of interest (gene/protein, disease, protein domain, etc.), and the arcs between two nodes are labelled with a weight representing the strength of the relationship between the connected entities. These weights are derived from different databases and other biological sources and their interpretation varies depending on the type of the connected entities and the final goal of the study (e.g. physical/structural similarity, regulatory dependence, similar functional roles, etc.). In our algorithm, the arc weights for each network are compiled in an adjacency matrix. The nodes of the graph are also labelled with a value (in $[0, 1]$), representing the degree of association to the query or the target set. There are two kinds of networks: a) networks representing interactions or similarities between entities of the same type, and b) networks representing interactions or similarities between entities of different type. Type b) networks are used to interconnect type a) networks.

The method we propose allows to propagate node values through paths along different data networks (representing different biological entities) in order to derive new information from the existing knowledge. This value propagation is performed in two directions. First, values are propagated within and between networks, through all the possible paths connecting the query set network to the target set network (not reaching the target set network). Second, values are also propagated within the target set network, starting from the target nodes. Both propagation processes follow the principle that the higher the weight of the arc between two entities is, the more similar the value of these two nodes should be. Therefore, these two label propagation processes derive a final graph in which the value assigned to a node represents its degree of relation with the query or target set, respectively. Finally, we measure the degree of relationship between the query and target sets by computing the correlation between the values assigned to the nodes in the target network and those assigned to their neighbour nodes in other networks, as proposed in previous works with good results for different prioritization tasks [7, 10]. This process is explained in detail in the following section.

This article is organized as follows. The method and the data sources are described in detail in section *Methods*. To validate the proposed methodology we integrate protein domain, PPI and phenotype networks and compare the results to those obtained by rcNet for gene prioritization and DomainRBF for domain prioritization. These results are presented in the *Results* section and show a significant improvement in terms of sensitivity and specificity. ProphNet is also applied to several case studies (namely Alzheimer, Diabetes Mellitus Type 2 and Breast Cancer) to identify putative candidate genes involved in these diseases. The results of these tests can be found in the section *Case Studies*. Finally, some conclusions and future work are presented.

2 Methods

Let D be a set of graphs (also referred to as networks) defined as $D_i = (V_i, E_i)$ for $i = 1, ..., n$, where V_i is a set of vertices which represent biological entities from a specific domain satisfying $V_i \cap V_j = \emptyset$, $\forall i, j$ such that $i \neq j$. Each node v_{ik} (with $k = 1, ..., |V_i|$) in D_i is labelled with a value $\Psi(v_{ik})$, initially set to zero, that indicates the degree of relationship to the query or target set, depending on the network v_{ik} belongs to. E_i is a set of weighted undirected arcs representing relationships, similarities or interactions between elements of V_i. The values of the nodes change while the weights of the arcs remain constant during the entire process. Let R be a set of graphs defined as $R_{ij} = (V_i \cup V_j, C_{ij})$, where C_{ij} is a set of weighted undirected arcs representing relationships, similarities or interactions between elements of V_i and V_j, with $i, j \in 1, ..., n$ and $i \neq j$. Therefore, R_{ij} describes the relationships between the biological entities from two different networks: D_i and D_j.

We define the heterogeneous global graph G as $G = (D, R)$. Let the graph $D_q \in D$ be the query network and let $D_t \in D$ be the target network. Given the global graph G, our goal is to find the degree of association between a set of nodes $Q \subseteq V_q$ called the query set and a set of nodes $T \subseteq V_t$ called the target set.

The initial values for the nodes in the query set are set to 1 ($\Psi(v_{qi}) = 1$ for all nodes $v_{qi} \in Q$), while the rest of the nodes are set to 0 ($\Psi(v_{qj}) = 0$ for nodes $v_{qj} \in V_q - Q$). The target network is initialized in the same way, but considering the nodes in V_t and T. The rest of nodes in G are initially set to 0.

As we explain below in more detail, our method performs a propagation within networks pumping information between nodes. This process is based on the Flow Propagation algorithm [17, 18], which uses the normalized Laplacian matrix to propagate labels between nodes in a network. The normalization takes into account the degree of each node to limit the bias toward annotations from high-degree nodes. This normalization is also critical for convergence. The Flow Propagation algorithm is similar to a Random Walk with Restart, basically differing in the normalization process that guides the propagation [18].

Let N be the non-normalized adjacency matrix of a network in G. Since $G = (D, R)$ and graphs in R are bipartite (i.e. the adjacency matrices of graphs in R are not squared), let assume N has r rows and c columns. A normalization for N can be computed as:

$$norm(N) = D^1_G N D^2_G,$$

where D^1_G and D^2_G are diagonal matrices where each component is defined as:

$$D^1_{G_{jj}} = \frac{1}{\sqrt{(\sum_{k=1}^{c} N_{jk})}}$$

for $j = 1, .., r$, and

$$D^2_{G_{kk}} = \frac{1}{\sqrt{(\sum_{j=1}^{r} N_{jk})}}$$

for $k = 1, .., c$.

We define $M = \{M_i \mid M_i = norm(D_i) \text{ where } i = 1, .., |D|\}$ as the set of normalized squared adjacency matrices of graphs in D. Similarly, we define $S = \{S_i \mid S_i = norm(R_i) \text{ where } i = 1, .., |R|\}$ as the set of normalized adjacency matrices of bipartite graphs in R.

Let $p_i = \{p_{i1}, ..., p_{ij}, ..., p_{il}\}$ be a path composed of networks from D connecting D_q and D_t, satisfying $p_{ij} \in D$, $p_{i1} = D_q$, $p_{il} = D_t$ and $p_{ij} \neq p_{ik}, \forall j \neq k$. To compute the degree of association between the query and target sets, we first propagate values from the query set within the query network, and from the target set within the target network, as described in Section *Value propagation inside networks*. Next, we identify all the possible paths $P = \{p_1, ..., p_{|P|}\}$ connecting the query network to the target network. Figure 2 shows an example of a global graph G composed of five different networks or domains, with three different paths connecting the query network to the target network. Since the number of networks is usually reduced, the computation of all the paths connecting D_q and D_t can be accomplished by a brute force algorithm.

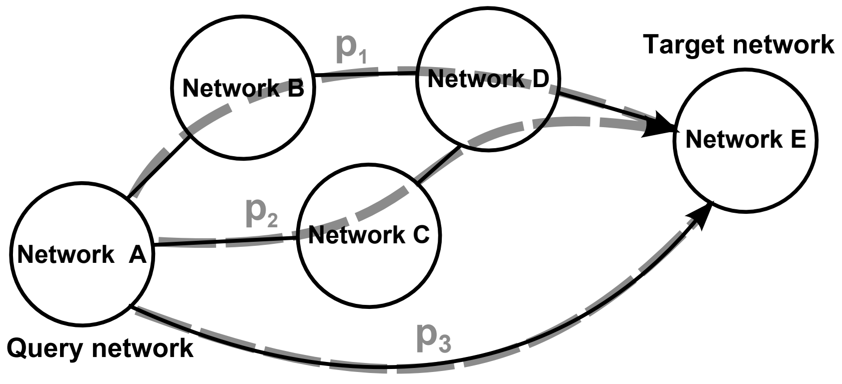


Figure 2: Example of computed paths. Three paths have been obtained connecting the query network to the target network.

For each computed path p_i, we propagate information from p_{ij} to the following network $p_{i(j+1)}$ in the path, as described in Section *Value propagation between networks.* Then we propagate information within the network $p_{i(j+1)}$, where $j = 1, 2, ..., l-2$. The propagation continues until it has been performed within the network $p_{i(l-1)}$ in the path.

Finally, after performing this propagation through each path in P, we correlate the values of the nodes in D_t against the values of the nodes in $p_{i(l-1)}$ directly connected to those in D_t, for all the paths $p_i \in P$. This step is described in Section *Value correlation between networks.* The obtained correlation value determines the degree of relationship between the query set and the target set.

Although measuring the degree of relationship between the query and target sets by correlating node values seems less intuitive than continuing the propagation of node values from the neighbours networks to the target nodes, the former approach has been shown to perform better than the latter (Supplementary material). Therefore, it was selected as the measure of similarity in our method. This approach was proposed in previous network-based prioritization methods with good results for different prioritization tasks [7, 10].

In order to accomplish prioritization tasks, in which only the query set $Q \in V_q$ and the target network V_t are provided by the user, we embed this pipeline into an iterative process to score each node in the target network according to its relationship with Q. This process is described in Section *Prioritization process.*

2.1 Value propagation inside networks

Several propagation methods have been proposed to compute the similarity or distance between nodes within a graph [4]. Methods based on local neighbourhood or shortest paths fail in capturing global interactions, in contrast to global methods that take into account the entire network topology [19].

ProphNet implements a flow propagation approach [17, 18] that uses a network's global information to perform a propagation of the values assigned to the nodes within this network. To carry out this propagation process within a network D_k, we first define the prior information set Z as those vertices v_{kj} with $\Psi(v_{kj}) \neq 0$. Therefore, the prior information set matches Q when propagating values within the query network, and the prior information set matches T when propagating values within the target network. The value $\Psi(v_{kj})$ of each node v_{kj} in Z ($j \in [1, |V_k|]$) is normalized as:

$$\Psi(v_{kj}) = \frac{\Psi(v_{kj})}{\sum^{v_{kx} \in V_k} \Psi(v_{kx})}$$

Let x_0 be a vector compiling the values initially assigned to each node in D_k, and $\widehat{x}$ a vector with the values assigned to each node in D_k after performing the propagation within D_k. To calculate $\widehat{x}$ we need to solve the following optimization problem:

$$\min_{\widehat{x}} \sum_{i,j} M_{k_{i,j}} (\widehat{x}_i - \widehat{x}_j)^2 + \frac{1-\alpha}{\alpha} \sum_i (\widehat{x}_i - x_{0_i})^2$$

where M_k is the network's normalized adjacency matrix. The closed-form solution of this equation is:

$$\widehat{x} = (1-\alpha)(I - \alpha M_k)^{-1} x_0.$$

This linear system can be solved exactly. However, there exists an iterative algorithm for solving this system which is faster for large networks [20]:

$$x_{i+1} = \alpha M_k x_i + (1-\alpha) x_0$$

with i starting from 0. This algorithm implements an iterative process where each node propagates its node value to its neighbours, based on the weights of the arc connecting them. The parameter $\alpha \in [0,1]$ determines the importance of the prior information set.

In order to further speed up this iterative process, we define the following stopping criterion: $|x_i - x_{i+1}| \leq \kappa$. This allows to stop the iterative process when it has almost converged, without the need of full convergence. Experimental tests (results not shown) prove that the best performance is obtained for $\kappa \leq 10^{-3}$.

For convenience, we refer to $\widehat{x}_{kj}$ as the vector obtained after convergence, where each component represents the value assigned to each node in the network D_k after performing the propagation within D_k, as part of a propagation process through the path p_j. Since the propagation values within the query and target networks are not affected by the propagation processes through the paths in P, we define $\widehat{x}_q$ as the vector obtained after propagating nodes values within the query network, and $\widehat{x}_t$ as the vector obtained after propagating nodes values within the target network.

2.2 Value propagation between networks

Given a network D_i whose vertices are already assigned a value according to $\widehat{x}_{il}$, we further propagate these values to the next network D_j in the current path p_l, with $D_j \neq D_t$. Since D_i and D_j are connected by R_{ij}, the information is propagated from the nodes of D_i to the nodes of D_j across the edges of R_{ij} by assigning each node v_{jk} from D_j a value computed as the mean of the nodes from D_i the node v_{jk} is connected with. This expression is formalized as:

$$\Psi(v_{jk}) = \frac{\sum^{v_{ix} \in neig_i(v_{jk})} \Psi(v_{ix})}{|neig_i(v_{jk})|}$$

where $neig_i(v_{jk})$ is the set of nodes from D_i that are directly connected to v_{jk} according to R_{ij}. A thresholding step is applied to this propagation process, since detailed experimentation suggested that nodes with very low values add noise to the process and reduce the performance (see supplementary material). To this end, a parameter $\gamma \in (0, 1]$ is included in the process so that the $\lceil |V_j|(1-\gamma) \rceil$ lowest node values from D_j after the propagation are updated to $\Psi(v_{jk}) = 0$. The rest of the node values are not changed.

2.3 Value correlation between networks

After the propagation process through one path finishes, the nodes in the networks which are adjacent to the target network present values that determine their degree of relationship to the query set. Also, the nodes in the target network are assigned a value that determines the degree of relationship with the target set. We can indirectly measure the relationship between the query set and the target set by measuring the similarity between the values of the nodes in the target network and those that are directly connected to them in adjacent networks. This can be calculated by simultaneously correlating these node values as derived by the propagation processes through all the different paths. For each path p_i with length l a vector $\overline{x}_i$ is computed as:

$$\overline{x}_i = S_a \widehat{x}_{(l-1)i}$$

where S_a is the normalized adjacency matrix of $R_{(l-1)(l)}$ and $\widehat{x}_{(l-1)i}$ is the vector obtained after propagating values inside the network D_{l-1}.

Since the values of the target network after the propagation process are represented by $\widehat{x}_t$, we define the vector $\overline{t}$ as:

$$\overline{t} = concat(\overbrace{\widehat{x}_t, ..., \widehat{x}_t}^{|P| \text{ times}})$$

and the vector $\overline{x}$ as:

$$\overline{x} = concat(\overline{x}_i) \quad \forall i \in [1, |P|]$$

where *concat* means concatenation. Both $\overline{x}$ and $\overline{t}$ are the same size.

Finally, the correlation value which derives a measure for the relationship between the query and target sets is computed as:

$$s = corr(\overline{x}, \overline{t})$$

where *corr* is Pearson's Correlation.

2.4 Prioritization process

In order to obtain a prioritized list of targets for a query set of nodes, we have to follow an iterative approach. For each target network node v_{te}, we set it as the target set T and compute the correlation value s as described in the previous section (we called this correlation s_e since it is computed for $T = \{v_{te}\}$). Once this correlation value has been computed for each target network node, these nodes are sorted in decreasing order according to their s_e value. Target nodes with high values of s_e are supposed to be strongly related to the query set. The entire algorithm is described in the pseudocode *Algorithm 1.*

Algorithm 1 ProphNet

```
**prioritize**($G$: global graph, $Q$: query set, $D_q$: query network, $D_t$: target network)
  Propagate values within $D_q$
  $P$: Compute the list of paths from $D_q$ to $D_t$ in $G$
  for each path $p_i = \{p_{i1}, ..., p_{ij}, ..., p_{il}\}$ in $P$ do
      for each network $p_{ij}$ in the path $p_i$ from $p_{i1}$ to $p_{i(l-1)}$ do
          Propagate values from $p_{ij}$ to $p_{i(j+1)}$
          Propagate values within $p_{i(j+1)}$
      end for
      Store the values of $D_{i(l-1)}$ after propagation through path $p_i$ as $\widehat{x}_{i(l-1)}$
  end for
  for each entity $e \in V_t$ in the target network $D_t$ do
      Set target set $T = \{e\}$
      Propagate values within $D_t$
      Compute correlation coefficient $s_e$ using the stored $\widehat{x}_{i(l-1)}$ for each path $p_i$.
  end for
  $L$: Sort all entities $e \in V_t$ by their $s_e$ values in descending order
  return $L$
```

2.5 Prioritization example

To facilitate the understanding of the algorithm, the Figure 3 shows a step-by-step representation of the ProphNet propagation processes. This figure shows two examples of a prioritization task involving three networks or domains, with the elements of each network represented by circles, squares and diamonds, respectively. For simplicity and clarity, node values are represented using a grey color scale (from white representing value 0 to black representing value 1) and the weight of an arc is represented by its line width. In the two examples, the prioritization task involves the same target set but different query sets. The query and target sets contain only one element in both cases, which is initially (step 1) set to 1 (black). Node values are propagated from the query nodes within the query network (step 2), and from the target nodes within the target network (step 3). There are two paths connecting the query network and the target network in these examples (circles-squares and circles-diamond-squares, respectively). Values from the query network are then iteratively propagated to adjacent non-target networks. Since the query network is directly connected to the target network in one of the paths, this step (step 4) is only applied to the path which includes an intermediate network (diamonds). Then, the propagation is performed within this intermediate network (step 5). This propagation continues until all the networks in all the paths connecting the query and target sets have been reached. Finally, we measure the strength of the connection between the query and the target sets as the correlation between the values assigned to the nodes in the target network and the values assigned to their neighbours from other networks (step 6).

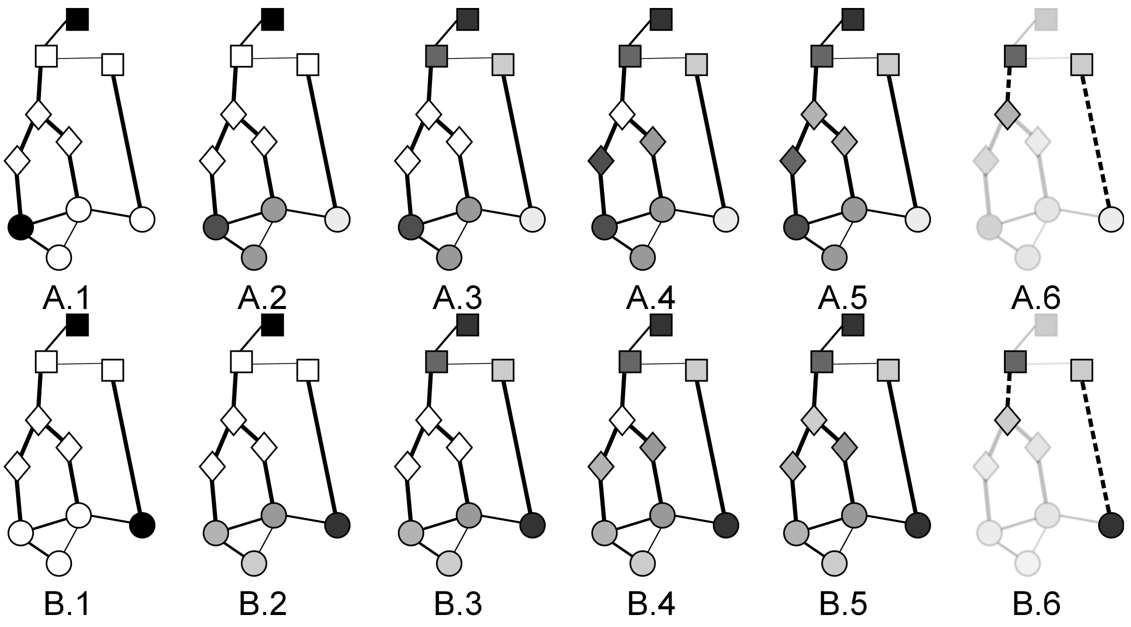


Figure 3: Step-by-step runs of ProphNet in two global graphs for the same target set but different query sets. Figure (a) shows an example in which propagated values from the query and target sets show a high correlation and therefore they seem to be related. In figure (b) propagated values from the query and target sets show low correlation, thus suggesting a weak relationship.

Figure 3a shows an example in which the values propagated from the query and target sets are highly correlated, suggesting a strong relationship between them. On the other hand, Figure 3b shows an example with low correlation values, which suggests that query and target sets are not related.

2.6 Algorithm complexity

The time complexity of the algorithm shown in the pseudocode *Algorithm 1* can be determined by aggregating the time complexity of each task it is composed of. Let n be the number of nodes in a network and m the number of networks in the global graph G. The task of propagating values within a network is $O(n^3)$. The propagation of values between networks is $O(n^2)$. The computation of the correlation coefficient for one path is

$O(n^3)$. The number of paths is bounded by $m!$ and their length by m. Therefore, the computational complexity of ProphNet is bounded by $O(m! \times m \times n^3)$. Despite this high complexity, typical execution times are a few seconds since the value of m is usually small in real applications. A summary of ProphNet execution times and memory usage for the experiments shown in this paper can be found in the Supplementary Material.

3 Results

As two specific case studies, we have applied ProphNet to prioritize candidate genes and protein domains associated to diseases. ProphNet has been compared with rcNet for gene-disease prioritization and with DomainRBF for domain-disease prioritization, since these methods were recently proposed and reported better results than previous methods [10,15]. ProphNet was run on a global graph composed of diseases, genes and protein-domains interconnected networks, while rcNet and DomainRBF were run on a genes-diseases and domains-diseases networks, respectively, as proposed by their authors. It is important to note that the ProphNet base case execution using only the gene and disease networks would obtain the same results than rcNet. The data sources used are described in detail in Section *Data sources.*

We tested the performance of the different methods on several leave-one-out (LOO) cross-validation experiments and for predicting new associations recently added to OMIM. To measure the performance of the different prioritization methods, we used Receiver Operating Characteristic (ROC) curves. ROC curves plot the true positive rate vs. the false positive rate at various threshold settings. The area under the ROC curve (AUC) was also computed. Finally, the average ranking position of the true target in the prioritized lists obtained by each method was also computed and normalized by dividing by the total number of elements in the list (5080 diseases in this case). We also computed p-values for comparing the average ranking values using two-tailed t-tests and the Bonferroni correction.

For the results reported for ProphNet, α was set to 0.9, the error threshold in the flow propagation was set to $\kappa = 10^{-5}$ and $\gamma = 0.00375$. For rcNet, we set the parameters to the values providing better results according to the authors: $\alpha = 0.9$, $\beta = 0.9$ and $\kappa = 10^{-5}$ [10] and used the enumeration-correlation based version.

3.1 Data sources

The disease phenotype network has been extracted from OMIM [21] using text mining techniques as described in [2]. Also, to perform a fair comparison of the results to those reported by rcNet, we used a version of OMIM from May, 2007 [10]. The obtained disease network contains 5080 OMIM disease phenotypes. The arcs are weighted with a value in the range $[0, 1]$. This weight measures the similarity between two phenotypes as the inverse of the distance between the feature vectors obtained by counting the occurrences of each term from the anatomy and disease sections of the Medical Subject Headings Vocabulary

(MeSH) in the description text for the corresponding entries in OMIM. The obtained disease network contains a total of 39,458 weighted interactions.

The gene network has been obtained from the Human Protein Reference Database (HPRD [22]). This protein-protein interaction network contains 64,662 unique interactions between 8,919 proteins. The network connecting genes and phenotypes has been extracted directly from OMIM (phenotype-gene relationship fields) obtaining 1,393 relationships.

The domain network has been derived from DOMINE [23] and InterDom [24] containing 48,778 unique relations between 5,490 domains. Relations between domains and genes were extracted from pFam [25]. Relations between domains and phenotypes have been extracted from Pfam and annotations of nsSNPs in the UniProt database [26].

The three networks (genes, protein domains and diseases) have simultaneously been used in the experiments performed with ProphNet. RcNet was executed using only the gene and disease networks, since this method does not allow to integrate more than two networks. DomainRBF was run on the domain and disease networks due to the same limitations.

3.2 Gene-Disease validation

To check whether the prioritization methods rcNet and ProphNet were able to retrieve a known relationship between a gene and a disease, we performed a leave-one-out cross-validation using gene-phenotype relations from OMIM. For each gene-phenotype relation reported in OMIM, we run the two algorithms on a network in which the explicit arc connecting the gene and phenotype of interest was removed. The gene of interest was set as the query set and the methods were asked to rank all the phenotypes associated to this query set.

The obtained ROC curves are shown in Figure 4. AUC values and avg. rank values for the target disease are displayed in *Table 1.* We can see that ProphNet outperforms rcNet in this test, ranking the target phenotype in a significantly higher position (corrected p-value < 0.05), with lower standard deviation and obtaining better AUC values. The high difference in terms of AUC value (over 10%) also suggests that the achieved improvement is not due to ProphNet prioritizing a little better those targets poorly prioritized by rcNet, but these targets being prioritized at the top by our method while they are poorly prioritized by rcNet. It is also important to note that, although a high percentage of the cases were prioritized in the top of the ranking, we found some results that were really worse ranked by both methods, significantly increasing the mean ranking and setting it far from the top 1 position. This also applies to experiments described in the following two sections.

3.3 Gene-Disease validation with new OMIM associations

Another validation that we have performed is predicting new associations between phenotypes and genes in 387 case studies from new entries added to OMIM between May 2007 and May 2010, since these relationships are not reported in the datasets used in our study. Each case study consists of a phenotype and a set of genes (mostly only one)

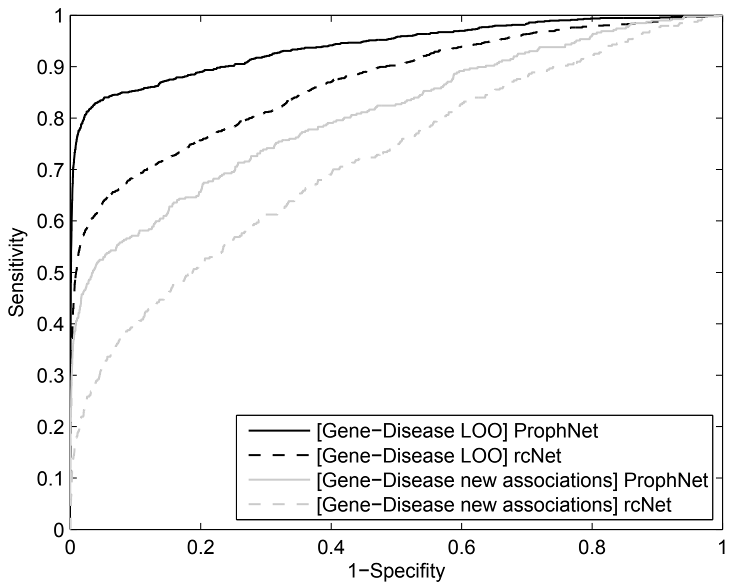


Figure 4: ROC curves for gene-disease prioritizations with ProphNet and rcNet.

Test	**Method**	**AUC**	**Normalized mean ranking (Std.Dev)**
Gene-disease LOO	ProphNet	0.9393	0.0609 (0.1597)
	rcNet	0.80572	0.1944 (0.2448)
Gene-disease new associations	ProphNet	0.80717	0.1930 (0.2618)
	rcNet	0.71636	0.2835 (0.2907)
Domain-disease LOO	ProphNet	0.9319	0.0683 (0.1537)
	domainRBF	0.8678	0.1322 (0.2361)

Table 1: Performance comparison for leave-one-out cross-validation prioritization experiments using OMIM.

associated with it. Results of the comparison can be seen in Figure 4. AUC values are shown in *Table 1.* The results show that our algorithm clearly outperforms rcNet (corrected p-value < 0.05) predicting new relationships not explicitly represented in the data network.

3.4 Domain-Disease validation

To prove that our algorithm not only prioritizes genes, but can prioritize other biological entities, we have performed a leave-one-out domain-disease validation test. For each relation between a domain and a phenotype in our datasets, we run the prioritization methods on a global network in which the direct arc connecting the protein domain and phenotype of interest was removed. The domain of interest was set as the query set and the methods were asked to rank all the phenotypes associated to this query set.

Our method has been compared with domainRBF for this task, since this method has been recently proposed for domain-disease prioritization and builds the phenotype-domain network using the same data sources considered in this study. We set the parameters of domainRBF testing for best performance. A diffusion kernel was selected to compute distances in interactions matrices. B_0 and V_0 were set to 0 and 1, respectively.

Results show that our method significantly improves the results provided by domainRBF for disease-domain prioritization (corrected p-value < 0.05). The highest difference in performance is around AUC 10%, which suggests that our method prioritizes more target phenotypes in the top of the ranking. ROC curves for this comparison can be seen in Figure 5.

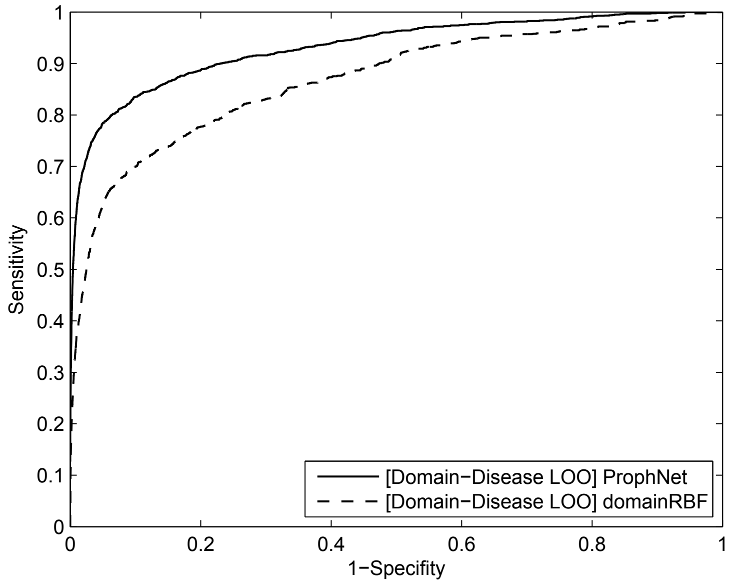


Figure 5: ROC curves for domain-disease prioritizations with ProphNet and domainRBF.

3.5 Robustness analysis

We carried out several experiments to test the robustness of ProphNet to parameter variation. First, we checked that varying the parameter α, which controls the importance of prior information in label propagation, does not significantly affect performance, as previous works suggested for other methods [10, 17]. Values ranging between 0.5 and 0.9 reported similar performance for ProphNet, but best result were obtained with α set to 0.9.

Second, we tested the impact of varying the parameter γ in the results. γ was used to limit the propagation of noise in the label propagation between different networks. The experiments showed that, although for any value of γ (in $[0, 1]$) ProphNet reported a good performance, the best results were obtained for γ in $[0.002, 0.004]$.

Results from these experiments can be found in the supplementary data.

3.6 Case studies

In order to show the applicability of the proposed method on real case studies, we have used it for gene-disease prioritization of some multifactorial disorders such as Alzheimer Disease, Diabetes Mellitus Type 2 and Breast Cancer, using the data sources described in section *Data sources.* Parameters were set to those which reported better results in the validation experiments ($\alpha = 0.9, \gamma = 0.00375$ and $\kappa = 10^{-5}$). A list of the genes ranked in the top positions for each disease are shown in Table 2, together with their assigned score. A detailed discussion about the role of these genes in the associated diseases can also be found in the Supplementary Material. A full list can be obtained by running the tool at the ProphNet website.

3.6.1 Results for Alzheimer Disease

Our method was used to prioritize genes related to Alzheimer Disease (MIM:104300). Table 2 shows genes ranked in the top positions which were previously known (OMIM records) to be connected with Alzheimer, such as *APP* and *PSEN2*. Furthermore, new relationships not explicitly reported in OMIM are suggested by analysing other genes in the top 10. For example, *MAPT* was ranked 3th in the obtained prioritized list. This gene provides the instructions for making a protein called *tau* that can be found throughout the nervous system (including neurons of the brain) so it has been associated with Alzheimer [27]. *PSEN1*, with known relations to Alzheimer type 3 [28] was ranked 4th. *TREM2* was ranked 5th, suggesting an important role in Alzheimer as shown by some population studies [29,30]. *HD/HTT* was ranked 6th, and although this gene has not yet been directly associated with Alzheimer, it has been shown to affect Huntington's disease [31]. More details about the other genes in the top 10 are provided in the Supplementary Material.

3.6.2 Results for Diabetes Mellitus Type 2

Our method was used to prioritize genes related to Diabetes Mellitus (DM) Type 2 (MIM:125853). Genes previously known to be connected with the disease, according to OMIM records, are: *IRS1*, *INSR*, *IPF1*, *SLC2A4*, *PPP1R3A* and *TCF1*, all ranked in the top 6 of the obtained prioritized list of genes. New putative candidate genes were discovered in the top 10. *PLN* (ranked 7th) was not related to Diabetes in the corresponding OMIM entry, however [32] reports a role of PLN in diabetic cardiomyopathy. *HADHSC* was ranked 8th since it has been related to Hyperinsulinemic hypoglycemia [33, 34]. The inferred relationship between Diabetes and *LEPRE1* (ranked 9th) cannot be derived from the published literature and further studies are required to study the possible connections of this gene to DM. Other interesting genes were ranked

Alzheimer Disease (MIM:104300)					
Gene	**Rank**	**Score**	**Gene**	**Rank**	**Score**
APP*	1	0.6639	CST3	7	0.1511
PSEN2*	2	0.5462	ITM2B	8	0.1468
MAPT	3	0.2531	TYROBP	9	0.1296
PSEN1	4	0.1946	SNCA	10	0.1276
TREM2	5	0.1700	APOE	11	0.1141
HD/HTT	6	0.1585	NCSTN	12	0.1114
Diabetes Mellitus Type 2 (MIM:125853)					
Gene	**Rank**	**Score**	**Gene**	**Rank**	**Score**
IRS1*	1	0.4744	HADHSC	8	0.0976
PPP1R3A*	2	0.4660	LEPRE1	9	0.0976
SLC2A4*	3	0.4194	LEPREL4	10	0.0976
IPF1*	4	0.3308	NEUROD1	14	0.0905
INSR*	5	0.2950	KCNJ11	30	0.0595
TCF1*	6	0.2168	ABCC8	37	0.0456
PLN	7	0.1164			
Breast Cancer (MIM:114480)					
Gene	**Rank**	**Score**	**Gene**	**Rank**	**Score**
BRCA1*	1	0.5019	TP53	8	0.1307
RAD51*	2	0.4919	ELAC2	9	0.1038
BRCA2*	3	0.4813	RAD51AP1	10	0.1031
NBN/NBS1*	4	0.3547	RAD54L	11	0.1031
PIK3CA*	5	0.3199	FANCD2	12	0.1017
MSH2	6	0.1636	ATM	13	0.0934
RB1	7	0.1607	CHEK2	29	0.0551

Table 2: Gene symbol, rank position and assigned score for genes in the top of the ranking for each case study. Entries marked with asterisks were directly connected by an arc to the disease of interest in the data network.

high, such as *KCNJ11*, ranked 30th, which presents polymorphisms that confer susceptibility to Diabetes mellitus type 2 [35]; or *ABCC8*, ranked 37th, whose mutations increase the risk of diabetes as suggested by [36].

3.6.3 Results for Breast Cancer

We performed a prioritization for Breast cancer (MIM:114480). Previously known genes related to this disease according to OMIM are: *BRCA1*, *RAD51*, *BRCA2*, *NBN* and *PIK3CA*, all included in the top 5 returned by ProphNet for this disease.

New relations not explicitly represented in the data network were discovered in the top ranking. Defects in *MSH2* (ranked 6th) can cause different types of cancer as pointed out by [37]. *RB1* (ranked 7th) and *TP53* (ranked 8th) act as tumour suppressors [38]. *ELAC2* (ranked 9th) has not been associated with breast cancer but with prostate cancer [39]. *RAD51AP1* (10th) is closely related with *RAD51* (2nd) [40]. *RAD54L* (11th) plays an important role repairing and recombining DNA in mammalian cells [41]. *FANCD2* (12th) interacts with the *BRCA1* and *BRCA2* genes in the DNA repair process to reduce the risk of breast cancer [42]. *ATM* (13th) has been associated with the disease in various studies [43]. Other relevant genes were found below in the top list, such as *CHEK2* (ranked 29th), also associated to propensity to suffer breast cancer as shown by [44].

4 Conclusion

In this paper we have introduced ProphNet, a novel network-based method that allows to accomplish any prioritization task from a network representing the corresponding data interactions. Our method is flexible and can be run on a graph composed of an arbitrary number of data networks representing biological entities of different type. This paper illustrates how to run ProphNet to perform gene-disease and domain-disease prioritization tasks, and provides experimental evidence that ProphNet outperforms other prioritization algorithms specifically designed for these tasks. A ProphNet web application has also been developed as a result of this work (the user guide can be found in the Supplementary Material).

The results obtained by ProphNet on real case studies on Alzheimer, DM and Breast Cancer show the potential of the method to suggest putative candidate genes to be involved in a disease. A detailed analysis of the literature also allowed us to validate the results provided by the algorithm, since many of the top ranked genes were already known to be related to the diseases. We consider that prioritization methods are useful for assisting scientists at early research stages and to formulate novel hypotheses of interest.

The extensive experimentation also allowed us to study the indirect influence of considering protein domains for the prioritization of candidate genes and diseases. Results show that the addition of domain interactions produces an obvious improvement with respect to existent algorithms, revealing the importance of this source of information (barely used before in this task). In the future, one of our main goals is to see how our method behaves in other prioritization problems and using different entities and sources of data not covered in this work. Furthermore, we plan to study in more detail the quality of the datasets and their influence on performance, and apply new methods of propagation to try to improve the results.

Competing interests

The authors declare that they have no competing interests.

Author's contributions

VM developed the ProphNet prioritization algorithm, carried out the experiments and wrote the paper. CC and AB guided and supervised the project and participated in writing the paper. All authors read and approved the final manuscript.

Acknowledgements

This work has been carried out as part of projects P08-TIC-4299 of J. A., Sevilla and TIN2009-13489 of DGICT, Madrid. VM was supported by Plan Propio de Investigacion, University of Granada.

References

[1] Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. Network medicine: a network-based approach to human disease. *Nature Reviews Genetics*, 12(1):56–68, 2011.

[2] Marc A van Driel, Jorn Bruggeman, Gert Vriend, Han G Brunner, and Jack AM Leunissen. A text-mining analysis of the human phenome. *European journal of human genetics*, 14(5):535–542, 2006.

[3] Yong Wang, Trupti Joshi, Xiang-Sun Zhang, Dong Xu, and Luonan Chen. Inferring gene regulatory networks from multiple microarray datasets. *Bioinformatics*, 22(19):2413–2420, 2006.

[4] Xiujuan Wang, Natali Gulbahce, and Haiyuan Yu. Network-based methods for human disease gene prediction. *Briefings in functional genomics*, 10(5):280–293, 2011.

[5] Gavin S Wilkie and Eric C Schirmer. Guilt by association the nuclear envelope proteome and disease. *Molecular & Cellular Proteomics*, 5(10):1865–1875, 2006.

[6] Yves Moreau and Léon-Charles Tranchevent. Computational tools for prioritizing candidate genes: boosting disease gene discovery. *Nature Reviews Genetics*, 2012.

[7] Xuebing Wu, Rui Jiang, Michael Q Zhang, and Shao Li. Network-based global inference of human disease genes. *Molecular Systems Biology*, 4(1), 2008.

[8] Oron Vanunu, Oded Magger, Eytan Ruppin, Tomer Shlomi, and Roded Sharan. Associating genes and protein complexes with disease via network propagation. *PLoS computational biology*, 6(1):e1000641, 2010.

[9] Yongjin Li and Jagdish C Patra. Genome-wide inferring gene–phenotype relationship by walking on the heterogeneous network. *Bioinformatics*, 26(9):1219–1224, 2010.

[10] TaeHyun Hwang, Wei Zhang, Maoqiang Xie, Jinfeng Liu, and Rui Kuang. Inferring disease and gene set associations with rank coherence in networks. *Bioinformatics*, 27(19):2692–2699, 2011.

[11] Euan A Adie, Richard R Adams, Kathryn L Evans, David J Porteous, and Ben S Pickard. Speeding disease gene discovery by sequence based candidate prioritization. *BMC bioinformatics*, 6(1):55, 2005.

[12] Euan A Adie, Richard R Adams, Kathryn L Evans, David J Porteous, and Ben S Pickard. Suspects: enabling fast and effective prioritization of positional candidates. *Bioinformatics*, 22(6):773–774, 2006.

[13] Stein Aerts, Diether Lambrechts, Sunit Maity, Peter Van Loo, Bert Coessens, Frederik De Smet, Leon-Charles Tranchevent, Bart De Moor, Peter Marynen, Bassem Hassan, et al. Gene prioritization through genomic data fusion. *Nature biotechnology*, 24(5):537–544, 2006.

[14] Kyle J Gaulton, Karen L Mohlke, and Todd J Vision. A computational system to select candidate genes for complex human traits. *Bioinformatics*, 23(9):1132–1140, 2007.

[15] Wangshu Zhang, Yong Chen, Fengzhu Sun, and Rui Jiang. Domainrbf: a bayesian regression approach to the prioritization of candidate domains for complex diseases. *BMC systems biology*, 5(1):55, 2011.

[16] W Wang, W Zhang, R Jiang, and Y Luan. Prioritisation of associations between protein domains and complex diseases using domain-domain interaction networks. *Systems Biology, IET*, 4(3):212–222, 2010.

[17] Oron Vanunu and Roded Sharan. A propagation based algorithm for inferring genedisease associations. In *German Conference on Bioinformatics*, pages 54–62. Citeseer, 2008.

[18] Saket Navlakha and Carl Kingsford. The power of protein interaction networks for associating genes with diseases. *Bioinformatics*, 26(8):1057–1063, 2010.

[19] Sebastian Köhler, Sebastian Bauer, Denise Horn, and Peter N Robinson. Walking the interactome for prioritization of candidate disease genes. *American journal of human genetics*, 82(4):949, 2008.

[20] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(753760):284, 2004.

[21] Ada Hamosh, Alan F Scott, Joanna S Amberger, Carol A Bocchini, and Victor A McKusick. Online mendelian inheritance in man (omim), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, 33(suppl 1):D514–D517, 2005.

[22] Suraj Peri, J Daniel Navarro, Ramars Amanchy, Troels Z Kristiansen, Chandra Kiran Jonnalagadda, Vineeth Surendranath, Vidya Niranjan, Babylakshmi Muthusamy, TKB Gandhi, Mads Gronborg, et al. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome research*, 13(10):2363–2371, 2003.

[23] Balaji Raghavachari, Asba Tasneem, Teresa M Przytycka, and Raja Jothi. Domine: a database of protein domain interactions. *Nucleic acids research*, 36(suppl 1):D656–D661, 2008.

[24] See-Kiong Ng, Zhuo Zhang, Soon-Heng Tan, and Kui Lin. Interdom: a database of putative interacting protein domains for validating predicted protein interactions and complexes. *Nucleic acids research*, 31(1):251–254, 2003.

[25] Robert D Finn, Jaina Mistry, Benjamin Schuster-Böckler, Sam Griffiths-Jones, Volker Hollich, Timo Lassmann, Simon Moxon, Mhairi Marshall, Ajay Khanna, Richard Durbin, et al. Pfam: clans, web tools and services. *Nucleic acids research*, 34(suppl 1):D247–D251, 2006.

[26] Eric Jain, Amos Bairoch, Severine Duvaud, Isabelle Phan, Nicole Redaschi, Baris E Suzek, Maria J Martin, Peter McGarvey, and Elisabeth Gasteiger. Infrastructure for the life sciences: design and implementation of the uniprot website. *BMC bioinformatics*, 10(1):136, 2009.

[27] John BJ Kwok, Clement T Loy, Gillian Hamilton, Edmond Lau, Marianne Hallupp, Julie Williams, Michael J Owen, G Anthony Broe, Nelson Tang, Linda Lam, et al. Glycogen synthase kinase-3β and tau genes interact in alzheimer's disease. *Annals of neurology*, 64(4):446–454, 2008.

[28] CM Van Duijn, Marc Cruts, Jessie Theuns, Geert Van Gassen, Hubert Backhovens, Marleen van den Broeck, Anita Wehnert, Sally Serneels, Albert Hofman, and Christine Van Broeckhoven. Genetic association of the presenilin-1 regulatory region with early-onset alzheimer's disease in a population-based sample. *European journal of human genetics: EJHG*, 7(7):801, 1999.

[29] Chiara Fenoglio, Daniela Galimberti, Laura Piccio, Diego Scalabrini, Paola Panina, Cecilia Buonsanti, Eliana Venturelli, Carlo Lovati, Gianluigi Forloni, Claudio Mariani, et al. Absence of trem2 polymorphisms in patients with alzheimer's disease and frontotemporal lobar degeneration. *Neuroscience letters*, 411(2):133–137, 2007.

[30] Benoit Melchior, Angie E Garcia, Bor-Kai Hsiung, Katherine M Lo, Jonathan M Doose, J Cameron Thrash, Anna K Stalder, Matthias Staufenbiel, Harald Neumann, and Monica J Carson. Dual induction of trem2 and tolerance-related transcript, tmem176b, in amyloid transgenic mice: implications for vaccine-based therapies for alzheimer's disease. *ASN neuro*, 2(3), 2010.

[31] NA Aziz, CK Jurgens, GB Landwehrmeyer, WMC van Roon-Mom, GJB Van Ommen, T Stijnen, RAC Roos, et al. Normal and mutant htt interact to affect clinical severity and progression in huntington disease. *Neurology*, 73(16):1280–1285, 2009.

[32] Loren E Wold, Asli F Ceylan-Isik, Cindy X Fang, Xiaoping Yang, Shi-Yan Li, Nair Sreejayan, Jamie R Privratsky, and Jun Ren. Metallothionein alleviates cardiac dysfunction in streptozotocin-induced diabetes: Role of ca¡ sup¿ 2+¡/sup¿ cycling proteins, nadph oxidase, poly (adp-ribose) polymerase and myosin heavy chain isozyme. *Free Radical Biology and Medicine*, 40(8):1419–1429, 2006.

[33] Anders Molven, Guri E Matre, Marinus Duran, Ronald J Wanders, Unni Rishaug, Pål R Njølstad, Egil Jellum, and Oddmund Søvik. Familial hyperinsulinemic hypoglycemia caused by a defect in the schad enzyme of mitochondrial fatty acid oxidation. *Diabetes*, 53(1):221–227, 2004.

[34] Els C van Hove, Torben Hansen, Jacqueline M Dekker, Erwin Reiling, Giel Nijpels, Torben Jørgensen, Knut Borch-Johnsen, Yasmin H Hamid, Robert J Heine, Oluf Pedersen, et al. The hadhsc gene encoding short-chain l-3-hydroxyacyl-coa dehydrogenase (schad) and type 2 diabetes susceptibility the damage study. *Diabetes*, 55(11):3193–3196, 2006.

[35] Anna L Gloyn, Michael N Weedon, Katharine R Owen, Martina J Turner, Bridget A Knight, Graham Hitman, Mark Walker, Jonathan C Levy, Mike Sampson, Stephanie Halford, et al. Large-scale association studies of variants in genes encoding the pancreatic β-cell katp channel subunits kir6. 2 (kcnj11) and sur1 (abcc8) confirm that the kcnj11 e23k variant is associated with type 2 diabetes. *Diabetes*, 52(2):568–572, 2003.

[36] Andrey P Babenko, Michel Polak, Hélène Cavé, Kanetee Busiah, Paul Czernichow, Raphael Scharfmann, Joseph Bryan, Lydia Aguilar-Bryan, Martine Vaxillaire, and Philippe Froguel. Activating mutations in the abcc8 gene in neonatal diabetes mellitus. *New England Journal of Medicine*, 355(5):456–466, 2006.

[37] Pieter J Westenend, Ronald Schütte, Monique MCP Hoogmans, Anja Wagner, and Winand NM Dinjens. Breast cancer in an msh2 gene mutation carrier. *Human pathology*, 36(12):1322–1326, 2005.

[38] Tom Walsh, Silvia Casadei, Kathryn Hale Coats, Elizabeth Swisher, Sunday M Stray, Jake Higgins, Kevin C Roach, Jessica Mandell, Ming K Lee, Sona Ciernikova, et al. Spectrum of mutations in brca1, brca2, chek2, and tp53 in families at high risk of breast cancer. *JAMA: the journal of the American Medical Association*, 295(12):1379–1388, 2006.

[39] Annika Rökman, Tarja Ikonen, Nina Mononen, Ville Autio, Mika P Matikainen, Pasi A Koivisto, Teuvo LJ Tammela, Olli-P Kallioniemi, and Johanna Schleutker. Elac2/hpc2 involvement in hereditary and sporadic prostate cancer. *Cancer research*, 61(16):6038–6041, 2001.

[40] Tonko Buterin, Caroline Koch, and Hanspeter Naegeli. Convergent transcriptional profiles induced by endogenous estrogen and distinct xenoestrogens in breast cancer cells. *Carcinogenesis*, 27(8):1567–1578, 2006.

[41] A Naderi, AE Teschendorff, NL Barbosa-Morais, SE Pinder, AR Green, DG Powe, JFR Robertson, S Aparicio, IO Ellis, JD Brenton, et al. A gene-expression signature to predict survival in breast cancer across independent data sets. *Oncogene*, 26(10):1507–1516, 2006.

[42] Nazneen Rahman, Sheila Seal, Deborah Thompson, Patrick Kelly, Anthony Renwick, Anna Elliott, Sarah Reid, Katarina Spanova, Rita Barfoot, Tasnim Chagtai, et al.

Palb2, which encodes a brca2-interacting protein, is a breast cancer susceptibility gene. *Nature genetics*, 39(2):165–167, 2006.

[43] Georgia Chenevix-Trench, Amanda B Spurdle, Magtouf Gatei, Helena Kelly, Anna Marsh, Xiaoqing Chen, Karen Donn, Margaret Cummings, Dale Nyholt, Mark A Jenkins, et al. Dominant negative atm mutations in breast cancer families. *Journal of the National Cancer Institute*, 94(3):205–215, 2002.

[44] Pia Vahteristo, Jirina Bartkova, Hannaleena Eerola, Kirsi Syrjäkoski, Salla Ojala, Outi Kilpivaara, Anitta Tamminen, Juha Kononen, Kristiina Aittomäki, Päivi Heikkilä, et al. A chek2 genetic variant contributing to a substantial fraction of familial breast cancer. *The American Journal of Human Genetics*, 71(2):432–438, 2002.

Supplementary material

Top 50 genes prioritized by ProphNet

Rank	Gene	Score	Rank	Gene	Score
1	APP	0.66395	26	SIRPB1	0.069677
2	PSEN2	0.54624	27	KCNIP4	0.067622
3	MAPT	0.25306	28	CHMP2B	0.066392
4	PSEN1	0.19459	29	ICAM5	0.06531
5	TREM2	0.17	30	APBB3	0.063825
6	HD	0.15854	31	APBA3	0.061983
7	CST3	0.15113	32	PRNP	0.06149
8	ITM2B	0.14676	33	COL25A1	0.059462
9	TYROBP	0.1296	34	HADHB	0.05945
10	SNCA	0.12759	35	CASP6	0.056407
11	APOE	0.11406	36	KCNIP3	0.055112
12	NCSTN	0.11144	37	SNCB	0.054846
13	PSENEN	0.09454	38	KIR2DS2	0.054442
14	APH1A	0.09454	39	NOTCH3	0.052885
15	APH1B	0.093457	40	KLRC3	0.052401
16	METTL2B	0.091739	41	PRSS3	0.05117
17	HADH2	0.088582	42	CHRNA7	0.050067
18	SPON1	0.086666	43	APBB2	0.04729
19	TM2D1	0.086666	44	NOTCH4	0.046163
20	BACE2	0.086666	45	CTNND2	0.044223
21	DOCK3	0.077497	46	SERPINI1	0.043768
22	CD300E	0.069677	47	FLNB	0.043587
23	CLEC5A	0.069677	48	CASP8	0.043244
24	NCR2	0.069677	49	APPBP1	0.042937
25	TREM1	0.069677	50	CTSD	0.042268

Table 1: Top 50 Alzheimer (MIM:104300) genes prioritized by ProphNet.

Rank	**Gene**	**Score**	**Rank**	**Gene**	**Score**
1	IRS1	0.4744	26	PHIP	0.062078
2	PPP1R3A	0.46597	27	ATP2A3	0.062078
3	SLC2A4	0.41937	28	ATP2A1	0.062029
4	IPF1	0.33078	29	ACOX1	0.060055
5	INSR	0.29497	30	KCNJ11	0.059551
6	TCF1	0.21682	31	C1QTNF5	0.059257
7	PLN	0.11641	32	-	0.050414
8	HADHSC	0.097584	33	ATP2A2	0.050161
9	LEPRE1	0.097584	34	STRN3	0.048852
10	-	0.097584	35	ARF3	0.048383
11	MLSTD2	0.097584	36	SLN	0.048282
12	FAM62B	0.097584	37	ABCC8	0.04558
13	IDH2	0.097584	38	PSMD7	0.044634
14	NEUROD1	0.090461	39	MVP	0.043341
15	PCSK1	0.077833	40	SNF1LK2	0.042414
16	SLC2A2	0.075549	41	EHD2	0.042374
17	TCF2	0.074199	42	PPARG	0.041591
18	MAFA	0.070823	43	PCSK1N	0.041119
19	PDIA6	0.070718	44	HK2	0.04051
20	FKBP10	0.069568	45	BPY2IP1	0.0403
21	KBTBD10	0.067907	46	SPOP	0.039407
22	DLD	0.066068	47	ENPP1	0.038238
23	ARFIP1	0.064772	48	MFRP	0.038163
24	RPS6KA1	0.064346	49	RAB7	0.037582
25	IAPP	0.062876	50	GATA5	0.036751

Table 2: Top 50 diabetes mellitus type II (MIM: 125853) genes prioritized by ProphNet.

Rank	**Gene**	**Score**	**Rank**	**Gene**	**Score**
1	BRCA1	0.50185	26	TREX1	0.060285
2	RAD51	0.49193	27	HMG20B	0.058471
3	BRCA2	0.4813	28	MRE11A	0.057521
4	NBN	0.3547	29	CHEK2	0.055145
5	PIK3CA	0.31986	30	CDK4	0.054203
6	MSH2	0.16361	31	ERCC2	0.052391
7	RB1	0.16072	32	BAP1	0.051401
8	TP53	0.13068	33	MSH6	0.051175
9	ELAC2	0.10385	34	-	0.050248
10	RAD51AP1	0.10305	35	MLH1	0.049023
11	RAD54L	0.10305	36	MUTYH	0.048841
12	FANCD2	0.10167	37	RPA1	0.048761
13	ATM	0.093381	38	C17orf28	0.048712
14	RNASEL	0.083185	39	TP53BP1	0.048516
15	BCCIP	0.0778	40	AXIN2	0.048076
16	SHFM1	0.077161	41	SMC1L1	0.047339
17	DCLRE1C	0.076188	42	BUB1B	0.046962
18	C11orf30	0.075987	43	FANCG	0.04636
19	BLM	0.074494	44	RAD52	0.04634
20	DMC1	0.070191	45	ATRX	0.045625
21	MDC1	0.069038	46	STK11	0.044099
22	RAD50	0.065818	47	BCL2	0.042166
23	H2AFX	0.06564	48	EXO1	0.041624
24	ATR	0.065553	49	GABRB1	0.04145
25	PTEN	0.060815	50	RET	0.040784

Table 3: Top 50 breast cancer (MIM:114480) genes prioritized by ProphNet.

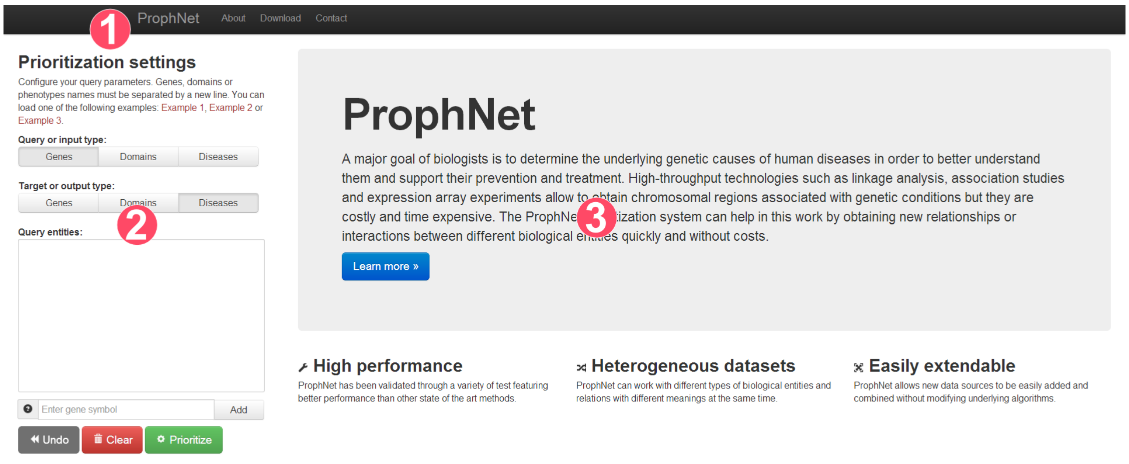
Robustness analysis results

Gamma	AUC	Normalized mean ranking (Std. Dev)
0.0005	0.79834	0.2018(0.2613)
0.0010	0.80599	0.1942(0.2604)
0.0020	0.80792	0.1922(0.2615)
0.0030	0.80749	0.1927(0.2621)
0.0040	0.80685	0.1933(0.2619)
0.0050	0.80669	0.1935(0.2617)
0.0100	0.80526	0.1949(0.2619)
0.0300	0.79982	0.2003(0.2644)
0.0500	0.79684	0.2033(0.2649)
0.1000	0.79160	0.2086(0.2666)
0.5000	0.78169	0.2185(0.2696)
1.0000	0.78169	0.2185(0.2696)

Table 4: Robustness analysis results for gene-disease validation with new OMIM associations).

ProphNet web tool user guide

ProphNet web tool allows users to perform prioritizations of genes-diseases-proteins domains. The tool has been designed to be easy to use. ProphNet does not require any registration or identification for use. When we enter prophnet we see the following screen.



Prioritization settings

Configure your query parameters. Genes, domains or phenotypes names must be separated by a new line. You can load one of the following examples: Example 1, Example 2 or Example 3.

Query or input type: Genes | Domains | Diseases

Target or output type: Genes | Domains | Diseases

Query entities:

ProphNet

A major goal of biologists is to determine the underlying genetic causes of human diseases in order to better understand them and support their prevention and treatment. High-throughput technologies such as linkage analysis, association studies and expression array experiments allow to obtain chromosomal regions associated with genetic conditions but they are costly and time expensive. The ProphNet prioritization system can help in this work by obtaining new relationships or interactions between different biological entities quickly and without costs.

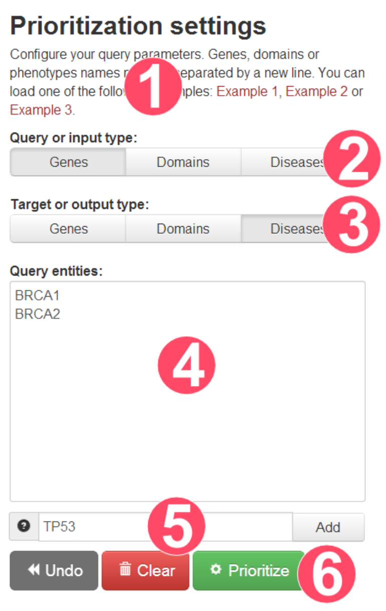
High performance
ProphNet has been validated through a variety of test featuring better performance than other state of the art methods.

Heterogeneous datasets
ProphNet can work with different types of biological entities and relations with different meanings at the same time.

Easily extendable
ProphNet allows new data sources to be easily added and combined without modifying underlying algorithms.

We can see three different parts in the interface. The action bar [1] allows direct access to a number of interesting additional information about ProphNet and Matlab source code. The prioritization settings [2] area is the space where the user sets the parameters of prioritization to be performed. Finally, the work space [3], which initially only contains information about ProphNet, is the area where the user will see the results of the prioritization.

Users interact with the tool using prioritization settings area. This area can be seen in more detail in the following figure.



Prioritization settings

Configure your query parameters. Genes, domains or phenotypes names r eparated by a new line. You can load one of the follo ples: Example 1, Example 2 or Example 3.

Query or input type:

Genes | Domains | Diseases

Target or output type:

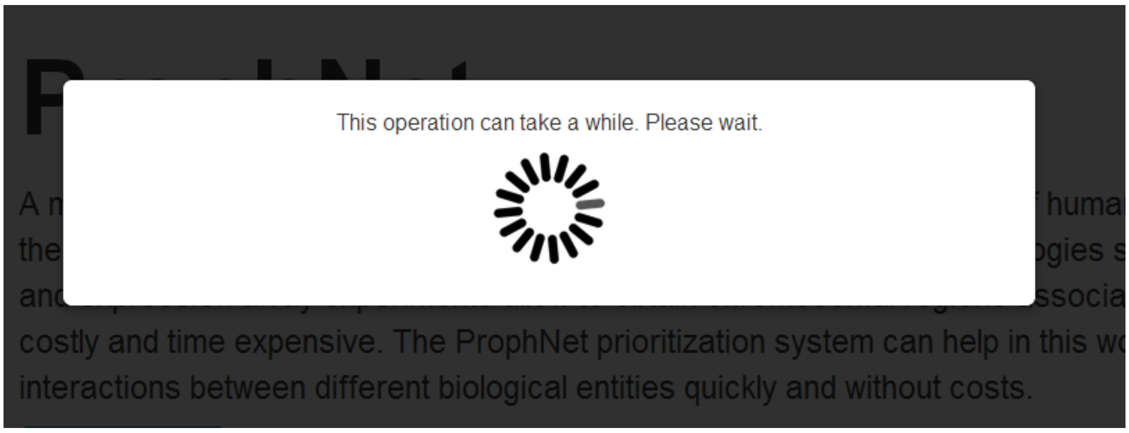
Genes | Domains | Diseases

Query entities:

```
BRCA1
BRCA2
```

TP53 | Add

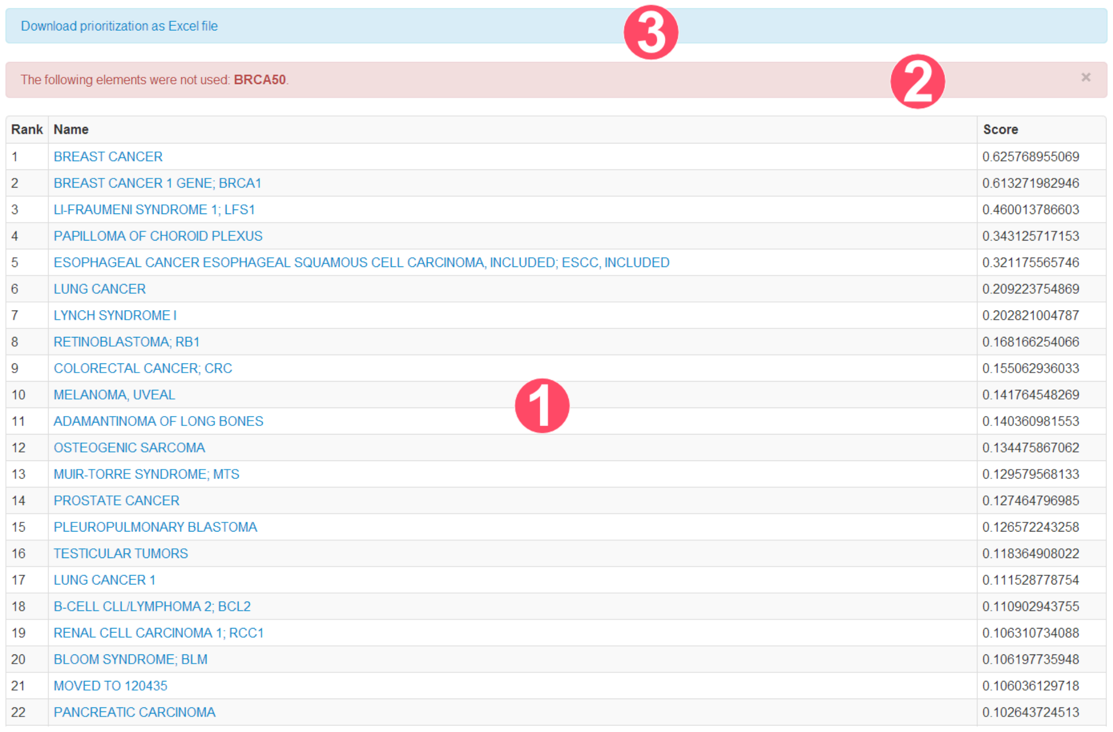
This area contains multiple elements. The first one, is a help text [1]. This text explains what type of input is expected. The user can also load some examples clicking on the links. To configure a prioritization the user first have to specify a query or input type [2] and a target or output type [3]. With the configuration of the figure, the user would introduce genes identifiers and would obtain a list of ranked diseases. The input list [4] is the component where the user introduces the list of identifiers of the elements for the query. Each element must be introduced in a new line. To help users to input correct names, an auto-complete component is available [5]. Based on user input, this component will suggest matching names. For details on what types of identifiers are supported, the user can press the button with the exclamation mark. Finally, some action buttons [6] are provided. After entering prioritization settings the user must press the *Prioritize* button. While performing prioritization the following dialog will be displayed.



This operation can take a while. Please wait.

costly and time expensive. The ProphNet prioritization system can help in this w
interactions between different biological entities quickly and without costs.

Prioritization can take several minutes depending on the server load. After this wait, results are shown as the following figure.



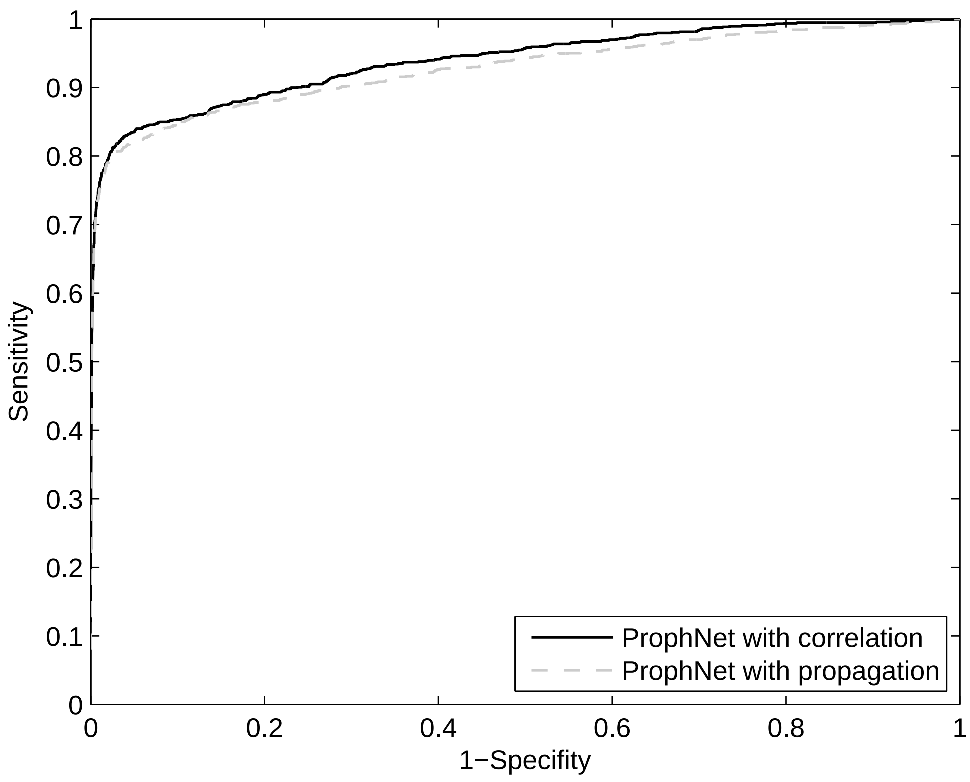
Download prioritization as Excel file

The following elements were not used: **BRCA50**.

Rank	Name	Score
1	BREAST CANCER	0.625768955069
2	BREAST CANCER 1 GENE; BRCA1	0.613271982946
3	LI-FRAUMENI SYNDROME 1; LFS1	0.460013786603
4	PAPILLOMA OF CHOROID PLEXUS	0.343125717153
5	ESOPHAGEAL CANCER ESOPHAGEAL SQUAMOUS CELL CARCINOMA, INCLUDED; ESCC, INCLUDED	0.321175565746
6	LUNG CANCER	0.209223754869
7	LYNCH SYNDROME I	0.202821004787
8	RETINOBLASTOMA; RB1	0.168166254066
9	COLORECTAL CANCER; CRC	0.155062936033
10	MELANOMA, UVEAL	0.141764548269
11	ADAMANTINOMA OF LONG BONES	0.140360981553
12	OSTEOGENIC SARCOMA	0.134475867062
13	MUIR-TORRE SYNDROME; MTS	0.129579568133
14	PROSTATE CANCER	0.127464796985
15	PLEUROPULMONARY BLASTOMA	0.126572243258
16	TESTICULAR TUMORS	0.118364908022
17	LUNG CANCER 1	0.111528778754
18	B-CELL CLL/LYMPHOMA 2; BCL2	0.110902943755
19	RENAL CELL CARCINOMA 1; RCC1	0.106310734088
20	BLOOM SYNDROME; BLM	0.106197735948
21	MOVED TO 120435	0.106036129718
22	PANCREATIC CARCINOMA	0.102643724513

We can see different parts in the results area. The ranked table [1] is the essential part. This table shows the rank, the name or identifier and the assigned score to each element. Names can be clicked for more details about each entry. The elements in the red bar [2] are those that have been excluded from the query due to not being in ProphNet database. Finally, blue bar [3] lets users download results as Excel files.

Propagation and correlation analysis results



Method	**Correlation**	**Propagation**
AUC	0.9393	0.9277
Mean ranking	309	368
Standard deviation	811	944

Table 5: Propagation and correlation analysis results for gene-disease leave-one-out validation.

ProphNet execution times and memory usage

Test	**Seconds per case**	**Memory usage (MB)**
Gene-disease LOO	1.76	2077.9
Gene-disease new associations	1.85	2077.8
Domain-disease LOO	2.09	2077.9

Table 6: Algorithm execution times and memory usage.

2.2 Disambiguation of semantic relations

The journal paper associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. Disambiguation of semantic relations using evidence aggregation according to a sense inventory. *Submitted to IEEE Transactions on Knowledge and Data Engineering (ISSN: 1041-4347).*

- Status: **Submitted**
- Impact Factor (JCR 2016): 3.438
- Subject Category:
  - Computer Science, Artificial Intelligence. Ranking 23/133.
  - Computer Science, Information Systems. Ranking 21/146.
  - Engineering, Electrical, & Electronic. Ranking 47/262.

Disambiguation of Semantic Relations Using Evidence Aggregation According to a Sense Inventory

Víctor Martínez Fernando Berzal Juan-Carlos Cubero

Abstract

This paper describes EPROP, a novel technique requiring little prior knowledge for word sense disambiguation of semantic relations between pairs of ambiguous concepts in knowledge bases. Our method makes inferences by aggregating evidences from ambiguous word interpretations and propagating the acquired knowledge over a taxonomy to generalize or specialize this knowledge. This propagation process allows the estimation of the degree of belief for each possible word sense assignment given the available evidence. EPROP only requires a sense inventory structured as a taxonomy to disambiguate a knowledge base by combining evidence from the ambiguous facts stored in the knowledge base. We have performed different experiments that show that our method achieves good results on the disambiguation of the semantic relations included in WordNet and ConceptNet. We also show how our method can be used to improve the performance of state-of-the-art word sense disambiguation methods.

1 Introduction

In natural language processing (NLP), word sense disambiguation (WSD) is the task of automatically determining the right sense of a polysemous word given its context [1, 2]. The problem of WSD is of great interest due to the intrinsic ambiguity of natural languages. Other fields of NLP can potentially benefit from advances in WSD, such as question answering [3, 4] or sentiment analysis [5, 6], just to mention two examples.

A large number of approaches with very different properties have been proposed to solve the WSD problem [7, 8]. A widely accepted categorization of WSD techniques is comprised of the following categories: supervised approaches, unsupervised approaches, and knowledge-based approaches.

Supervised approaches are based on applying machine learning techniques over sense-labeled training data to build classifiers that predict the sense of each word considering certain features [9,10]. Their applicability is highly limited due to the need for labeled data (typically hand-annotated by human experts), which is scarce and expensive to obtain, leading to the *knowledge acquisition bottleneck* [11,12]. The main argument for their usage is that these methods seemed to report the best performance in practice. However, in recent years, other approaches are rivaling supervised methods without requiring large sets of labeled data, relegating supervised methods to a second plane [13, 14].

Given the limitations of the supervised approach to WSD, it would be reasonable to think of purely unsupervised approaches as an adequate alternative. Completely unsupervised approaches work by automatically analyzing large unlabeled text corpora, which are usually easy to obtain, and clustering words occurrences [15, 16]. Different clusters for the same word are assumed to represent different senses for that word. Despite totally unsupervised approaches are very interesting, since they do not require prior knowledge, their main limitation is that the induced senses cannot be mapped to reference sense inventories. This leads to important difficulties when applying them and evaluating their performance in practice.

Unsurprisingly, knowledge-based approaches have become very popular in the field of WSD. They tend to offer slightly lower performance results than supervised approaches, but a remarkable wider coverage with less prior knowledge [17]. Knowledge-based approaches exploit knowledge resources (such as dictionaries, thesauri, and ontologies, among others) to predict the correct word sense according to an established sense inventory. Four categories of knowledge-based techniques are usually considered:

- Lesk's algorithm and its variations relying on maximizing the overlap of considered senses given their corresponding dictionary entries [18, 19, 20].
- Approaches based on choosing the senses that maximize semantic similarity, usually by minimizing distances in a given semantic graph [21, 22, 23, 24, 25].
- Selectional preference-based methods try to capture knowledge about which sense is more adequate given a predicate [26, 27].
- Approaches relying on properties usually observed in text, such as one sense per discourse, one sense per collocation, or just considering the most frequent sense [28,29].

Knowledge-based methods usually rely on WordNet, a large lexical database of English which has become a *de facto* standard in the field of WSD [30, 31]. In WordNet, senses are usually represented as sets of synonym words, often called synsets, disposed in a taxonomy through hyponymy and hypernymy relations. Other semantic relations between senses are also included in WordNet, such as part holonymy or antonymy. These semantic relations are exploited by knowledge-based methods for the task of WSD. However, these semantic relations are limited, sparse, and require difficult to gather expert knowledge. An open research problem is the automatic extension or enhancement of WordNet with more semantic relations, with the goal of improving the performance of WSD techniques.

In this paper, we propose EPROP (short for evidence propagation), a novel technique for the disambiguation of relational knowledge bases relying on little prior knowledge to perform sense disambiguation of its concepts according to a sense inventory. Our technique relies only on a taxonomic representation of word senses and the set of unlabeled or ambiguous instances stored in the relational knowledge base. Our method accumulates evidence over ambiguous instances while exploiting the taxonomy to extract unobserved knowledge. The accumulated evidence is finally applied to disambiguate the sense of two concepts related by a semantic relation.

This paper is organized as follows. Section 2 introduces the previous work related to our proposal. Section 3 describes the ideas that motivate our proposal, the mathematical derivation of our method, and the description of an algorithm for its efficient implementation. In Section 4, we describe the evaluation procedure followed in this manuscript. In section 5, we evaluate the performance and robustness of our technique compared to some baseline methods and a state-of-the art technique in the task of disambiguating ambiguous relations derived from WordNet. In Section 6, our approach is applied to the disambiguation of some hand-annotated ConceptNet relations and its performance is evaluated and compared to baseline methods and a state-of-the-art technique. In section 7, we show how the instances disambiguated by our approach can be used to improve the results obtained by state-of-the-art knowledge-based word sense disambiguation techniques. Conclusions and suggestions for future research are presented in Section 8.

2 Related work

Different approaches have been proposed for the automatic extension of WordNet with new semantic relations to overcome the knowledge acquisition bottleneck problem.

A growing line of research from recent years is mining Wikipedia to extract relations, which are mapped to WordNet synsets. For example, [14] computes a mapping from Wikipedia pages to WordNet synsets allowing the use of the links between Wikipedia pages as WordNet semantic relations. A major effort in this direction has been the construction of BabelNet using these mappings, which led to state-of-the-art WSD [32, 33]. Other authors have explicitly expanded WordNet relations using implicit information. For example, [34] extracts relations from synset glosses exploiting different properties, such as the transitivity of some relations. Other approaches have been proposed to extract relations from raw text, such as [35]. This approach parses a corpus of sentences and merges dependency relations to capture semantic relations.

These approaches have proven to be useful for the construction of an enhanced WordNet and its application to the WSD problem. However, they are omitting interesting information captured by relational knowledge bases, which explicitly store valuable knowledge using a structured representation. Different large relational knowledge bases are currently available, such as ConceptNet [36, 37] or YAGO [38]. Knowledge bases are semantic networks containing different kinds of useful knowledge, including everyday basic knowledge ("water is used for drinking"), cultural knowledge ("Alhambra is part of Granada"), and scientific knowledge ("atom is part of molecule"). The information included in these relational knowledge bases could be exploited to improve the performance of the state-of-the-art knowledge-based WSD techniques.

However, most of these knowledge bases are ambiguous and concepts are not mapped to a sense inventory as commonly required by WSD techniques, limiting their applicability to solve NLP-related problems. An approach for the automatic disambiguation of these concepts could be potentially useful for the NLP community. Unfortunately, little work has been done on this topic. To the best of our knowledge, the only previous attempt to disambiguate ConceptNet was done by [39, 40]. However, their method relies on the Normalized Google Distance (NGD), requiring a large number of queries to Google's search engine, which is not possible according to their current terms of service and does not allow for the design of repeatable experiments, since search results change every time Google updates its ranking algorithm.

3 Method

In this Section, we describe all the details related to the derivation and implementation of our proposed approach. First, we exemplify the intuitive ideas that motivate the reasoning behind our approach. Next, we describe the step-by-step mathematical derivation of our method. Finally, we describe the algorithm we propose in full detail.

3.1 Motivation

Let us assume that we observe the ambiguous relation *"a bus has a trunk"*. Without further prior knowledge, we do not know whether *bus* refers to *"a vehicle carrying many passengers"* or *"an electrical conductor that makes a common connection between several circuits"*, nor whether *trunk* refers to *"the main stem of a tree"* or refers to *"compartment in an automobile for luggage"* (other senses are left out for the sake of simplicity). Let us assume that we then observe another ambiguous relation *"an omnibus has a boot"*. For this new relation, we do not know whether *omnibus* refers to *"a vehicle carrying many passengers"* or to *"an anthology of articles"*, nor whether *boot* refers to *"compartment in an automobile for luggage"* or to *"footwear that covers the whole foot and lower leg"*. However, since both relations have overlapping sense assignments for their concepts, it is reasonable to increase our belief that the actual senses are *"a vehicle carrying many passengers"* for *bus* and *omnibus*, and *"compartment in an automobile for luggage"* for *trunk* and *boot.* As we observe in this example, two ambiguous instances can increase our degree of belief in one or more specific sense assignments.

However, observing the exact same semantic relation with different lemmas can be difficult even when using a large training corpus. Even worse, some senses may be represented by only a polysemous word or lemma, not allowing us to exploit this redundancy. In the following examples, we illustrate some potential solutions to this problem. Let us assume again that we only observe *"a bus has a trunk"*, thus we have the same degree of belief for all possible sense assignments. Then, we observe *"a vehicle has a boot"*, where we do not know if *vehicle* refers to *"a motor vehicle with four wheels"* or to *"any substance that facilitates the use of a drug, pigment, or other material"*. Since a *bus* is also a *vehicle*, we increase our degree of belief in the previously mentioned sense assignment. This exemplifies that pairs of senses with hyponymy and hypernymy relations share some properties and that they can potentially play the same role in a semantic relation. Even more, observing *"a car has a boot"* could increase our degree of belief in the correct sense assignment, since one of the senses of *car* is also a vehicle, leading to *bus* and *car* semantically having some very similar specific sense assignments. In this context, it is reasonable to assume that *bus* and *car* share properties, e.g. having a compartment for luggage in our example. Observing relations with overlapping senses at different abstraction levels or with semantic similarity is useful in the disambiguation task, as shown in these examples, and these observations allow us to overcome some problems of existing techniques.

As it has been shown in the previous examples, the accumulation of ambiguous data can increase our confidence on particular sense assignments. We can exploit the taxonomical structure of senses to propagate evidence to other semantically similar senses, thereby overcoming the sparseness and scarcity of data. In this context, we face the problem of gradually increasing our degree of belief on certain facts given some evidence. To solve this problem, we propose a framework with a derivation similar to the certainty factor (CF) model [41], a popular method of approximate reasoning with uncertainty. Our framework only manages belief updates in a much more restricted way than the CF model does, without requiring ad hoc elements with little theoretical background and avoiding some assumptions that are controversial when applying certainty factors to reasoning, which have been criticized despite being useful in practice [42].

3.2 Approach description

Let $\mathcal{T} = (S, H)$ be an is-a taxonomy of senses permitting multiple inheritance, where S is the set of senses and H the set of is-a triplets where $h_i \in H$ is defined as $h_i = (s_i^s, IsA, s_i^t)$, with $s_i^s, s_i^t \in S$, and h_i represents "s_i^s is a s_i^t". The left member of a triplet is called the source and the right member is called the target. For convenience, we define a function $parents(\mathcal{T}, s)$ and a function $children(\mathcal{T}, s)$. The function $parents(\mathcal{T}, s)$ takes a taxonomy $\mathcal{T}$ and a sense s as arguments, and it returns the set of senses being the target of each is-a relation in the taxonomy where the sense s is the source. The function $children(\mathcal{T}, s)$ takes again a taxonomy $\mathcal{T}$ and a sense s as arguments, but now it returns the set of senses being the source of each is-a triplet in the taxonomy where the sense s is the target.

Our approach assumes that there is an unobserved set U of unambiguous semantic relations that represents what can be known. However, we only observe this set U through a set of ambiguous observations $O = \{o_1 \dots o_{|O|}\}$ where $o_i = (l_i^s, type, l_i^t)$ with l_i^s and l_i^t being lemmas, which can be mapped to subsets of senses in S. This mapping is done through a function $senses(lemma)$, which takes a lemma and returns all the senses that match the lemma according to its associated part-of-speech (POS) tag.

The problem we want to solve is, given the taxonomy of senses $\mathcal{T}$, the set of ambiguous observations O, and an ambiguous instance $z = (l^s, type, l^t)$ to disambiguate, computing the current degree of belief for each possible sense assignment $y = (s^s, type, s^t)$ with $s^s \in senses(l^s)$ and $s^t \in senses(l^t)$. We treat this problem as equivalent to computing $P(y \in U|O)$, the degree of belief on y belonging to the unobserved set U of true facts, denoted as $P(y|O)$ for the sake of brevity. It is important to note that only observations of the same type as the instance under disambiguation are relevant, allowing us to ignore observations involving other types of semantic relations.

First, we need to derive a method for computing the degree of belief for each possible solution y given the current evidence O, noted as $P(y|O)$. We start from Bayes' theorem

$$P(\overline{y}|O)P(O) = P(O|\overline{y})P(\overline{y}),$$

that we can trivially transform into the Bayes' formula as

$$P(\overline{y}|O) = \frac{P(O|\overline{y})P(\overline{y})}{P(O)}.$$

Assuming the conditional independence of observations, we can rewrite this expression as

$$P(\overline{y}|O) = \prod_{o \in O} \frac{P(o|\overline{y})}{P(o)} P(\overline{y}).$$

According to Bayes' theorem, the term $\frac{P(o|\overline{y})}{P(o)}$ is equal to $\frac{P(\overline{y}|o)}{P(\overline{y})}$, thus we can rewrite the previous expression as

$$P(\overline{y}|O) = \prod_{o \in O} \frac{P(\overline{y}|o)}{P(\overline{y})} P(\overline{y}).$$

Now, given the basic probability axiom $P(\overline{y}) = 1 - P(y)$, we can obtain the following expression:

$$P(y|O) = 1 - \prod_{o \in O} \frac{1 - P(y|o)}{1 - P(y)} (1 - P(y)).$$

Since the prior probability of each possible triplet is the same without prior knowledge, part of this expression can be transformed into a constant C as follows

$$P(y|O) = 1 - C \prod_{o \in O} (1 - P(y|o)).$$

where $C = \prod_{o \in O} \frac{1}{1-P(y)} (1 - P(y))$. This constant can be approximated to 1, since, assuming that $P(y) \to 0$ due to the huge space of combinations compared to the triplets that actually exist, it is expected that $C \to 1$.

The only remaining computation required to apply this expression is the estimate $P(y|o)$, which can be read as the probability of y belonging to the unobserved set of real facts given observation o. For convenience, we can express this value in terms of the probability of observing certain hidden evidence e given the observation o, the probability of y^s being the source of the triplet given evidence e and the probability of y^t being the target of the triplet given evidence e. We can write this as:

$$P(y|o) = P(y|e)P(e|o) = P(y^s \in y, y^t \in y|e)P(e|o).$$

Assuming that the probability of the source and the probability of the target are conditionally independent, we obtain

$$P(y|o) = P(y^s \in y|e)P(y^t \in y|e)P(e|o).$$

The term $P(e|o)$, which we call the weight of the observation, is useful to weigh observations according to the confidence we have in them. If no confidence scores are given for the observations, an uniform probability distribution can be used in order to weigh all observations equally.

To compute $P(y^s \in y|e)$ and $P(y^t \in y|e)$, we estimate the probability of each possible mapping of each side of the observation o and the probability of y^s and y^t being the source and the target of the triplet given that mapping, respectively. Since we do not have prior knowledge for the observations and we assume that they are conditionally independent, all mappings are initially assumed to have the same probabilities and we can compute them as

$$\begin{aligned} P(y^s \in y|e) &= \sum_{s^s \in senses(o^s)} P(y^s \in y|s^s \in o)P(s^s \in o|e) \\ &= \sum_{s^s \in senses(o^s)} \frac{P(y^s \in y|s^s \in o)}{|senses(o^s)|} \end{aligned}$$

for the source of the triplet and as

$$\begin{aligned} P(y^t \in y|e) &= \sum_{s^t \in senses(o^t)} P(y^t \in y|s^t \in o)P(s^t \in o|e) \\ &= \sum_{s^t \in senses(o^t)} \frac{P(y^t \in y|s^t \in o)}{|senses(o^t)|} \end{aligned}$$

for the target of the triplet. The terms $P(y^s \in y|s^s \in o)$ and $P(y^t \in y|s^t \in o)$ are equal to the probability of y^s and y^t being the source and the target of the triplet given that s^s is the source and s^t is the target in the current observation, respectively.

In order to model the behavior observed in the example we used to motivate our approach, where senses may inherit properties from more abstract or general senses, we introduce a propagation process where a node has a probability α of sharing a property with a children in the taxonomy and a probability β of inheriting a property from a parent in the taxonomy, where the property in our context is being the source or the target of a triplet. Let α be the upward propagation factor, which is the probability of a sense u to be the source or the target given it is the parent of a sense v that is the source or the target with certain probability, that is $\alpha = P(u \in y|u \in parents(\mathcal{T}, v), v \in y)$. Let β

be the downward propagation factor, which is the probability of a sense u to be the source or the target given it is the child of a sense v that is the source or the target with certain probability, this is $\beta = P(u \in y | u \in children(\mathcal{T}, v), v \in y)$. Therefore, we can compute $P(y^s \in y | s^s \in o)$ and $P(y^t \in y | s^t \in o)$ as a propagation process starting, respectively, from s^s and s^t. Starting with probability 1, crossing each is-a link attenuates this probability by α and crossing each is-a reverse link attenuates this probability by β. Therefore, the α and β damping parameters model how often properties can be generalized (promoted as properties of the parents in the is-a taxonomy) and specialized (applied to children in the is-a taxonomy), respectively. A simple example of this propagation process is shown in Figure 1. We heuristically derive the values of parameters α and β from the assumption that the properties of the parent depend on the joint properties of its children, and each children provides the same amount of information. Therefore, we estimate these parameters as $\alpha = \beta = 1/c$, where c is the average number of children for non-leaf nodes. These two parameters are set to 0.2 in our experiments since the value of c is close to 5 in WordNet.

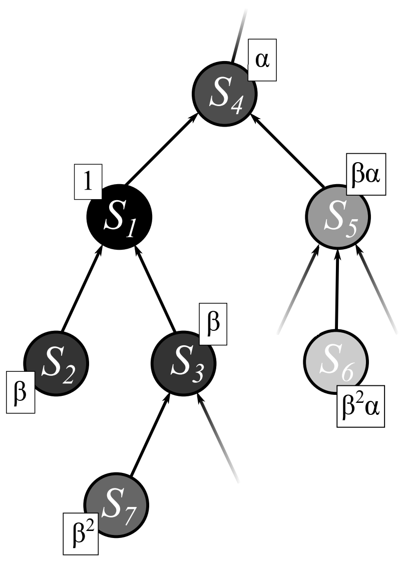


Figure 1: Evidence propagation from sense s_1 to other senses. The value in the box attached to each sense s_i is equal to $P(s_i \in y | s_1 \in o)$. Node gray level is proportional to this value to ease visualization. Arrows represent is-a links.

We can observe that propagating evidence across the network in this way has a high computational complexity, since it requires to cross every is-a link in H for each possible sense mapping for each observation, leading to $O(ohs)$ computational complexity, where o is the number of observations, s the number of senses for each lemma, and h the number of is-a links in the taxonomy. Precomputing the propagation from each sense is also unfeasible, if we consider its time and spatial computational complexity. Optionally, a threshold κ could be set to stop the propagation of beliefs when it falls below the threshold, making the algorithm more efficient (and reducing the noise from weak evidence). In the next section, we propose an algorithmic approach that, with or without a threshold, reduces the number of required computations.

3.3 Algorithm Description

We have seen that a naïve application of our proposed approach would lead to a too high computational complexity, in both time and space. Instead, we propose an algorithm that yields the same results in a much more efficient way. Its basic idea is to start by propagating information upwards and, when computing the degree of belief on the existence of a connection of a certain type between a pair of senses, traverse from these nodes to the root, gathering all the evidence that should have been previously propagated downwards. Finally, all the gathered evidence is aggregated to compute the degree of belief on that specific sense mapping.

Our algorithmic approach is composed of three main building blocks: upward evidence propagation, evidence gathering, and evidence aggregation. These algorithmic steps are described in full detail below. We also provide the computational complexity for each step.

The first step is upward evidence propagation, described in Algorithm 1. An annotation index, containing each evidence for each sense, is created and populated in this step. The evidence stored in the annotation index is the result of spreading the evidence from each observation from lower senses (those the observation directly maps to) to higher senses, until reaching the root of the taxonomy. This process consists of launching a recursive propagation to parent nodes, applying the upward propagation factor α in each step until reaching the root node or falling below the propagation threshold κ when such a threshold is used. If the same evidence is going to be annotated in the same annotation index registry multiple times, the highest propagated score is kept. The computational complexity of propagating the evidence for all observations is $O(osd)$, where o is the number of observations, s the number of senses each concepts maps to, and d the depth of the taxonomy.

Given the populated annotation index obtained by the previous algorithm, evidence for a particular sense can be queried using Algorithm 2. The key of this algorithm is traversing the taxonomy from the query sense to the root by gathering evidence, attenuating by the downward propagation factor β in each step. As in the upward propagation step, this process can also be stopped if attenuation falls below an optional propagation threshold κ. This algorithm returns an annotation list, a structure containing all the evidence from all the observations that should have been propagated to the query sense node. The computational complexity of this step is also $O(ods)$, since we must traverse the taxonomy up to its root, potentially gathering evidence from all the available observations. It should be noted that this is avoided in practice by setting a propagation threshold slightly above zero to speed up the computation at the cost of small rounding errors.

Finally, in the third step, all the gathered evidence is combined to compute the current degree of belief on both senses being connected by the specified semantic relation. This step is described in Algorithm 3. Function *weight(id)* returns the weight initially assigned to the relation with the *id* passed as argument. Function *sense_cnt(id)* returns the number of possible senses for the lemma that generated the annotation with the *id* passed as argument. All the evidence contained in the annotation lists for the source and the target senses are combined, taking their weight into consideration, as previously described. Both lists are transformed into a data structure containing the probability of that assignment for each

Algorithm 1 Upward evidence propagation process.

```
Input: Taxonomy $\mathcal{T}$, upward propagation factor $\alpha$, observed ambiguous set $O$, optional
    propagation threshold $\kappa$
Output: Annotated index $A$
 1: $A \leftarrow$ Create annotation index
 2: for $o = (o^s, type, o^t)$ in $O$ do
 3:     $id_o \leftarrow$ Create unique id
 4:     for $s$ in $senses(o^s)$ do
 5:         $id_u \leftarrow$ Create unique id
 6:         PROPAGATION($s$,$id_o$,$id_u$,$type$,$source$,1)
 7:     end for
 8:     for $s$ in $senses(o^t)$ do
 9:         $id_u \leftarrow$ Create unique id
10:         PROPAGATION($s$,$id_o$,$id_u$,$type$,$target$,1)
11:     end for
12: end for
13: return $A$
14: procedure PROPAGATION($s$,$id_o$,$id_u$,$t$,$d$,$v$)
15:     if $v \geq \kappa$ and $v \geq A[s][t][d][id_o][id_u]$ then
16:         $A[s][t][d][id_o][id_u] \leftarrow v$
17:         for $p$ in $parents(\mathcal{T}, s)$ do
18:             PROPAGATION($p$,$id_o$,$id_u$,$t$,$d$,$v * \alpha$)
19:         end for
20:     end if
21: end procedure
```

observation. This structure is iterated at the same time, combining and adding compatible evidence in order to obtain the final score. For symmetrical relations, the combination is also performed by swapping source and target, since observing the relation in one direction is also equivalent to observing the relation in the opposite direction. The computational complexity of this step is $O(os)$, although in practice we only work with a subset of all the observations.

The computational complexity of initializing the annotation index and making n predictions is bounded by $O(odsn)$. The amount of computation required in practice is much lower since, most of the time, only a small part of the total taxonomy is walked through and only a fraction of all observations are gathered.

Following the algorithms described in this section, the current belief on two specific senses being connected by a specific semantic relation given a set of observations can be updated. In order to disambiguate a specific instance, the algorithm must be applied to all the possible pairs of senses both concepts in the relation map to, obtaining a belief score for each one. Finally, sense assignments can be ranked according to these scores.

Algorithm 2 Evidence gathering query process.

```
Input: Taxonomy $\mathcal{T}$, annotation index $A$, downward propagation factor $\beta$, sense $sense$,
    type $type$, side $side$, optional propagation threshold $\kappa$
Output: Annotation list $L$
 1: $L \leftarrow$ Create annotation list
 2: GATHER($L$,$sense$,$type$,$side$,1)
 3: return $L$
 4: procedure GATHER($L$,$s$,$t$,$d$,$\gamma$)
 5:     for $id_o$ in $A[s][t][d]$ do
 6:         for $id_u$ in $A[s][t][d][id_o]$ do
 7:             $v \leftarrow A[s][t][d][id_o][id_u] * \gamma$
 8:             if $v \geq \kappa$ and $v \geq L[id_o][id_u]$ then
 9:                 $L[id_o][id_u] \leftarrow v$
10:             end if
11:         end for
12:     end for
13:     for $p$ in $parents(\mathcal{T}, s)$ do
14:         GATHER($L$,$p$,$t$,$d$,$\gamma * \beta$)
15:     end for
16: end procedure
```

3.4 An Illustrative Example

In this Section, we show a simple example of how our approach works. We use the example that we introduced in Section 3.1 to motivate our approach, where the goal is to disambiguate the relations *"a bus has a trunk"* and *"an omnibus has a boot"*. Let us assume that we have the simplified sense taxonomy shown in Figure 2. To represent an annotation, we use the notation $(unique\_id, relation\_id, side, score)$, where the side is denoted by s for source annotations and by t for target annotations.

We introduce the annotations corresponding to the first relation with id 1. This implies that $bus_{circuit}$ and $bus_{vehicle}$ are annotated with $(1, 1, s, 1)$ and $(2, 1, s, 1)$, respectively, and

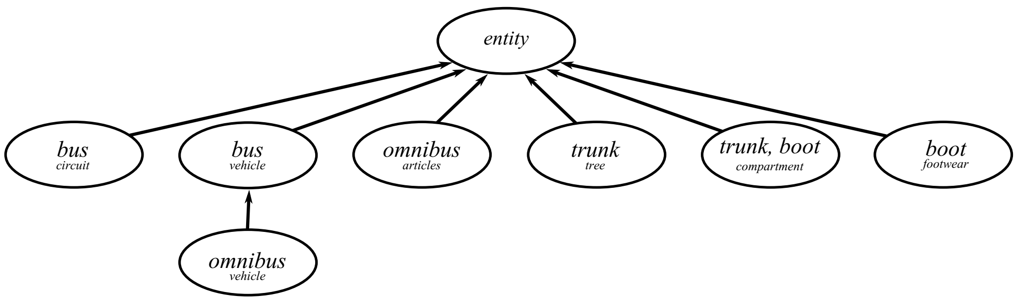


Figure 2: Simplified sense taxonomy for the example, where nodes represent senses and arrows is-a relations. For each node, the larger text label represents the sense lemma and the smaller text annotation indicates which specific sense is associated with the node.

Algorithm 3 Evidence aggregation.

```
Input: Source annotation list $L_s$, target annotation list $L_t$, type $type$
Output: Relative degree of belief $score$
 1: $M_s \leftarrow$ TRANSFORM($L_s$)
 2: $M_t \leftarrow$ TRANSFORM($L_t$)
 3: $invScore \leftarrow$ AGGREGATE($M_s$,$M_t$)
 4: if $type$ is symmetrical then
 5:     $invScore \leftarrow$ AGGREGATE($M_t$,$M_s$)
 6: end if
 7: $score \leftarrow 1 - invScore$
 8: return $score$
 9: procedure TRANSFORM($L$)
10:     $M \leftarrow$ Create hash table
11:     for $id_o$ in $L$ do
12:         for $id_u$ in $L[id_o]$ do
13:             $v \leftarrow weight(id_o) * L[id_o][id_u]/sense\_cnt(id_u)$
14:             $M[id_o] \leftarrow M[id_o] + v$
15:         end for
16:     end for
17:     return $M$
18: end procedure
19: procedure AGGREGATE($M_s$,$M_t$)
20:     $locInvScore \leftarrow 1$
21:     for $id_o$ in $M_s$ do
22:         $v \leftarrow 1 - M_s[id_o] * M_t[id_o]$
23:         $locInvScore \leftarrow locInvScore * v$
24:     end for
25:     return $locInvScore$
26: end procedure
```

trunk$_{tree}$ and *trunk*$_{compartment}$ are annotated with $(3, 1, t, 1)$ and $(4, 1, t, 1)$, respectively. These annotations are propagated upwards resulting in their storage in *entity* with scores α as the result of multiplying the upward propagation probability α with their own scores, which are initially equal to 1. We proceed with the second relation, with id 2. This step involves annotating *omnibus*$_{articles}$ and *omnibus*$_{vehicle}$ with source annotations $(5, 2, s, 1)$ and $(6, 2, s, 1)$, respectively, and *boot*$_{compartment}$ and *boot*$_{footwear}$ with target annotations $(7, 2, t, 1)$ and $(8, 2, t, 1)$, respectively. The annotations in *omnibus*$_{articles}$, *boot*$_{compartment}$, and *boot*$_{footwear}$ are propagated exactly as in the previous relation to *entity*. However, the annotation in *omnibus*$_{vehicle}$ is propagated to *bus*$_{vehicle}$ with score α and, finally, to *entity* with score α^2. To summarize, the annotations currently stored in the annotation index are:

- *entity*: $(1, 1, s, \alpha)$, $(2, 1, s, \alpha)$, $(3, 1, t, \alpha)$, $(4, 1, t, \alpha)$, $(5, 2, s, \alpha)$, $(6, 2, s, \alpha^2)$, $(7, 2, t, \alpha)$, $(8, 2, t, \alpha)$
- *bus*$_{circuit}$: $(1, 1, s, 1)$
- *bus*$_{vehicle}$: $(2, 1, s, 1)$, $(6, 2, s, \alpha)$

- $omnibus_{articles}$: $(5, 2, s, 1)$
- $omnibus_{vehicle}$: $(6, 2, s, 1)$
- $trunk_{tree}$: $(3, 1, t, 1)$
- $trunk, boot_{compartment}$: $(4, 1, t, 1)$, $(7, 2, t, 1)$
- $boot_{footwear}$: $(8, 2, t, 1)$

Once our taxonomy has been annotated, we can query which relations are more likely. For example, we want to compute the probability

$$P((omnibus_{vehicle}, HasA, boot_{compartment})|O)$$

and the probability

$$P((omnibus_{articles}, HasA, boot_{footwear})|O)$$

using the currently stored annotations.

To compute the first probability, we gather the annotations that should have been propagated to $omnibus_{vehicle}$, which includes its own annotations, annotations from $bus_{vehicle}$, and annotations from *entity*. The gathered source annotations are $(6, 2, s, 1)$ from itself, $(2, 1, s, \beta)$ from $bus_{vehicle}$, and $(1, 1, s, \alpha\beta)$ and $(5, 2, s, \alpha\beta)$ from *entity*. For $boot_{compartment}$ the gathered target annotations are $(4, 1, t, 1)$ and $(7, 2, t, 1)$ from itself, and $(3, 1, t, \alpha\beta)$ and $(8, 2, t, \alpha\beta)$ from entity. To compute the probability of the relation we must combine pairs of source and target annotations with the same *relation_id*, which are the following pairs: $(1, 3)$,$(1, 4)$,$(2, 3)$,$(2, 4)$,$(5, 7)$,$(5, 8)$,$(6, 7)$, and $(6, 8)$. Using the evidence aggregation expression, assuming $C \to 1$, $\alpha = 0.2$, and $\beta = 0.2$, we can compute the relation probability as

$$P((omnibus_{vehicle}, HasA, boot_{compartment})|O) =$$
$$1 - C(1 - \frac{\alpha^2\beta^2}{4})(1 - \frac{\alpha\beta}{4})(1 - \frac{\alpha\beta^2}{4})(1 - \frac{\beta}{4})(1 - \frac{\alpha\beta}{4})(1 - \frac{\alpha^2\beta^2}{4})(1 - \frac{1}{4})(1 - \frac{\alpha\beta}{4}) = 0.76$$

To compute the second probability, we proceed in the same way. First, we gather the source annotations for $omnibus_{articles}$: $(5, 2, s, 1)$ from itself, and $(1, 1, s, \alpha\beta)$, $(2, 1, s, \alpha\beta)$, and $(6, 2, s, \alpha^2\beta)$ from *entity*. Later, we gather the target annotations for $boot_{footwear}$: $(8, 2, t, 1)$ from itself and $(3, 1, t, \alpha\beta)$, $(4, 1, t, \alpha\beta)$, and $(7, 2, t, \alpha\beta)$ from *entity*. Finally, we combine compatible annotations using the evidence aggregation expression as

$$P((omnibus_{articles}, HasA, boot_{footwear})|O) =$$
$$1-C(1-\frac{\alpha^2\beta^2}{4})(1-\frac{\alpha^2\beta^2}{4})(1-\frac{\alpha^2\beta^2}{4})(1-\frac{\alpha^2\beta^2}{4})(1-\frac{\alpha\beta}{4})(1-\frac{1}{4})(1-\frac{\alpha^3\beta^2}{4})(1-\frac{\alpha^2\beta}{4}) = 0.26$$

Given these probabilities, we can see how $omnibus_{vehicle}$ and $boot_{compartment}$ is a more likely sense assignment for the relation *"an omnibus has a boot"* than $omnibus_{articles}$ and $boot_{footwear}$, given the current observations.

4 Evaluation Procedure

Due to the novel nature of the proposed technique, we did not find existing evaluation strategies to be suitable for the analysis of results for the problem of the disambiguation of semantic relations. We implemented different evaluation procedures, described in the following sections, to measure the performance of the compared techniques in different tasks related to the disambiguation of semantic relations.

We used WordNet 3.0 in our experiments, which has become a *de facto* standard in the field of WSD [31]. We obtained our taxonomy for nouns and verbs by considering *hypernym* and *instance* as is-a relations. Since verbs are in unconnected hierarchies within WordNet, we created a virtual root node as the parent of all the root nodes of these verb hierarchies. Adjectives and adverbs are omitted in this work, since WordNet does not provide taxonomic information for them.

In the following section, we describe in detail the performance metrics that we have used in this manuscript to compare the different disambiguation techniques evaluated in the proposed tasks. Next, we introduce the baseline and state-of-the-art methods included in our comparison.

4.1 Performance Metrics

We used two performance scores to evaluate the results obtained by each method in our experiments.

First, we used the Area Under the ROC Curve (AUC) [43], which is equivalent to the probability that a randomly-chosen correct sense assignment is ranked above a randomly-chosen incorrect sense assignment [44]. It is computed as

$$AUC = \frac{\sum_{p \in P} \sum_{n \in N} I(rank(p) > rank(n))}{|P||N|},$$

where P is the set of correct sense assignments, N is the set of incorrect sense assignments, and I is the indicator function, which equals 1 if the condition holds and 0 otherwise.

We also used Mean Reciprocal Rank (MRR) [45], commonly applied to evaluate information retrieval techniques, as a second performance measure. The reciprocal rank of an example is the multiplicative inverse of the rank of the first correct pair of senses. The rank is assigned by sorting pairs of senses according to the score computed by each one of the evaluated method. The MRR is then computed as

$$MRR = \frac{1}{|Q|} \sum_{i \in Q} \frac{1}{rank(p_i^{(1)})},$$

where Q is the set of examples in the test set and $p_i^{(1)}$ is the first correct sense assignment for the i-th example in the test set.

4.2 Alternative Methods

In order to evaluate the performance of our proposed method, we considered methods that could be used in our current evaluation settings. Since we do not require labeled data, supervised and semi-supervised methods were left out. Previously proposed unsupervised approaches could not be applied either, since they do not map senses to a reference inventory. Most knowledge-based techniques are omitted, since they require unambiguous expert-gathered information beyond the taxonomy of senses. Therefore, we compared our method with the following techniques:

- *Random (RAND):* We included a completely random approach to measure the gain each method provides over random sense assignment. Scores obtained by this method are the average of the score of each possible sense assignment.

- *Most frequent sense (MFS):* Most frequent sense assignment is a reasonable strategy that has shown to perform well in practice. WordNet provides frequency data for synsets and, despite our method ignores frequency data, we can easily devise a baseline method where

$$score = rank(source) + rank(target),$$

  with rank being the position in the list of senses sorted by frequency (highest ranks for the most frequent senses).

- *Closest senses (CS):* Most previous knowledge-based methods assume that proper senses are those that minimize a specific distance-related measure between senses in a given semantic graph. We consider the most basic distance-related score, which can be defined as

$$score = -dist(source, target),$$

  where *dist* computes the distance among source and target senses in the taxonomy.

- *Novel approach to a semantically-aware representation of items (NASARI):* A more sophisticated approach to compute how close two senses are, in comparison to the previously-mentioned strategy, is using vector-based embeddings for word senses. We use NASARI embedded vector representations, which consist of 300 dimensions. These vector embeddings are are the result of gathering information from different sources for BabelNet synsets and applying an iterative algorithm to obtain the final embeddings as vectors for each synset [46]. BabelNet synsets can be directly mapped to WordNet synsets, since BabelNet provides the mapping between both resources. As suggested by the authors of NASARI, we use the cosine distance to measure how far two synsets are. A major limitation of NASARI is that it was designed to compute embeddings for noun concepts, but not for verbs. Therefore, it can only be used in tests involving nouns. In the cases where the vector for a sense was not available, the vector of the closest hyperonym with an embedding available was used instead.

5 Experiments with WordNet

In this section, we evaluate the performance of the technique we propose for the disambiguation of semantic relations.

For these experiments, we use some annotated semantic relations among senses included in WordNet. We considered the following semantic relations in our experiments: part holonymy / meronymy (noun to noun), substance holonymy / meronymy (noun to noun), member holonymy / meronymy (noun to noun), antonym (noun to noun, symmetric), entailment (verb to verb), cause (verb to verb), and antonym (verb to verb, symmetric). Derivation and grouping relations were left out.

We created our training set by considering each possible combination of lemmas for each given semantic relation. We initially assume that all combinations of a semantic relation have the same probability of being observed. Thus, generated observations were weighted to sum 1 for each semantic relation. The test set was similarly created by considering each possible combination of lemmas for each given semantic relation, but in this case observations were filtered to include only those where the obtained lemmas in both sides of the relation mapped to more than one sense, thereby ensuring ambiguity in test instances. Instances in the test set were also weighted to sum 1 for each relation, so that no relation had more influence in the validation independently from the number of test instances derived from it. We performed a cross-validation over the generated data, using training instances as the ambiguous observed set of knowledge to predict senses for test instances.

For the optional threshold we chose $\kappa = 10^{-4}$ as a reasonable value. Results obtained by each method for each semantic relation in WordNet are shown in Table 1.

As it can be seen, our method clearly outperforms other methods in all relations, often by a large margin. Our technique is only beaten in the AUC score obtained by the closest sense for the antonymy relation for nouns, where this baseline technique obtains remarkable performance scores, specially when compared to its performance in other relations, where it is only slightly better than random sense assignment (or even worse in substance holonym). The reason why the closest distance works pretty well for antonyms is probably due to the tendency of antonymous senses to be very similar semantically and, thus, being close in the taxonomy. However, we can see how the closest senses method obtains much worse performance scores for other relations when elements do not need to be semantically similar according to the taxonomy (i.e. a Spaniard is a native or an inhabitant of Spain, but the first refers to a person and the second is a country, far away in the semantic taxonomy). Other limitation of distance-based techniques (CS and NASARI) is their inability to estimate the direction of the relation since they are symmetric scores. For this reason, both methods obtain specially poor results in the substance relation, where in many cases both sides of the relation share the same lemma. These examples highlight the limitations of distance-based methods. The most frequent sense also obtains slightly better results than the random method, yet clearly worse than the results obtained by our evidence propagation method.

In order to test if our approach is robust in the selection of α and β parameters, we performed a robustness evaluation experiment. This experiment consisted of fixing one of the two parameters to 0.2 and measuring the performance obtained by varying the other

Relation	#Test	#Training	Method	AUC	MRR
Part	5123	38510	**RAND**	0.5000	0.3151
			MFS	0.4991	0.3569
			CS	0.5887	0.4917
			NASARI	0.6998	0.5675
			EPROP	**0.9139**	**0.7997**
Substance	337	2810	**RAND**	0.5000	0.3769
			MFS	0.5001	0.4637
			CS	0.3728	0.3800
			NASARI	0.4536	0.4244
			EPROP	**0.9025**	**0.8217**
Member	633	60486	**RAND**	0.5000	0.3529
			MFS	0.4879	0.3964
			CS	0.4943	0.4363
			NASARI	0.7324	0.6262
			EPROP	**0.9139**	**0.8346**
Antonym (n)	1050	3851	**RAND**	0.5000	0.3253
			MFS	0.5011	0.3871
			CS	**0.8824**	**0.8013**
			NASARI	0.7747	0.6335
			EPROP	0.8721	0.7748
Entailment	1399	2352	**RAND**	0.5000	0.1870
			MFS	0.6170	0.2970
			CS	0.5937	0.3285
			NASARI	N/A	N/A
			EPROP	**0.9148**	**0.6530**
Cause	987	1254	**RAND**	0.5000	0.1958
			MFS	0.5501	0.2343
			CS	0.4237	0.1675
			NASARI	N/A	N/A
			EPROP	**0.8421**	**0.5409**
Antonym (v)	1968	3784	**RAND**	0.5000	0.2386
			MFS	0.5222	0.2887
			CS	0.6307	0.4493
			NASARI	N/A	N/A
			EPROP	**0.8524**	**0.6657**

Table 1: Statistics and performance in the WordNet disambiguation task.

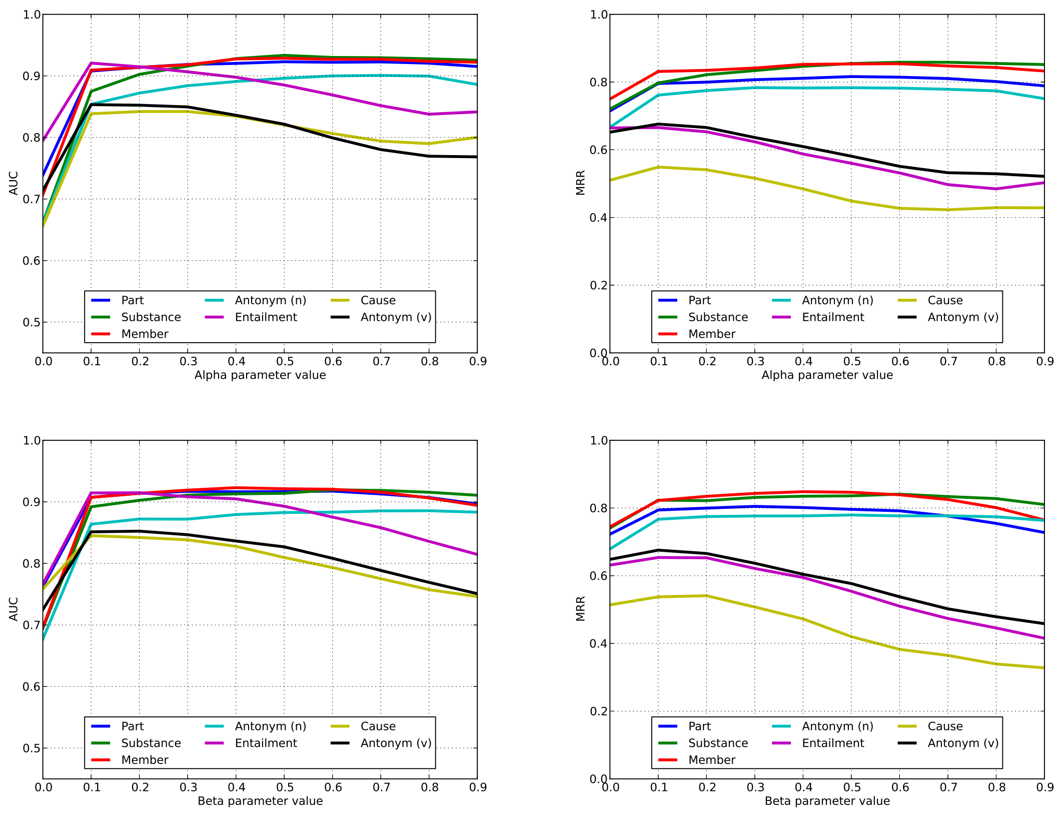


Figure 3: Robustness of EPROP for different semantic relations by fixing one of its propagation parameters to 0.2.

parameter. Robustness results are shown in Figure 3. It can be seen how the algorithm performance remains reasonably consistent with the variation of each parameter, specially for noun-related semantic relations.

In these experiments, we have shown that our method does not only obtains better results than a random sense assignment, but it also clearly outperforms other approaches by a large margin, obtaining remarkable AUC and MRR scores. These results show that our technique is not only reasonable in theory, but also obtains good results when applied in practice.

6 Disambiguating ConceptNet

In this experiment, we try to disambiguate a popular ambiguous semantic network relying only in the WordNet taxonomy and the ambiguous facts stored in the semantic network. The semantic network to disambiguate is ConceptNet 5 [36], whose knowledge has been collected from different sources, including crowdsourced resources, expert-created resources, and games with a purpose, among others. Except for instances gathered from

WordNet, most concepts in ConceptNet are not mapped to a sense inventory, which limits its applicability to NLP-related problems.

Relation	#Test	#Training	Method	AUC	MRR
AtLocation	50	21242	**RAND**	0.5000	0.2791
			MFS	0.7355	0.6558
			CS	0.6643	0.5904
			NASARI	0.8195	0.6933
			EPROP	**0.8725**	**0.7692**
PartOf	50	10799	**RAND**	0.5000	0.2458
			MFS	0.6296	0.4514
			CS	0.7240	0.5636
			NASARI	**0.9024**	**0.7511**
			EPROP	0.8160	0.6459
HasA	50	1307	**RAND**	0.5000	0.2026
			MFS	0.6571	0.4884
			CS	0.6957	0.4760
			NASARI	**0.9033**	**0.7525**
			EPROP	0.8693	0.5784
MadeOf	50	1478	**RAND**	0.5000	0.2784
			MFS	0.7387	0.6259
			CS	0.6792	0.5049
			NASARI	0.8689	0.6851
			EPROP	**0.8900**	**0.7268**

Table 2: Statistics and performance in the ConceptNet disambiguation task.

In order to give an estimate of the precision of the considered methods in the disambiguation of some relations in ConceptNet, we manually annotated 50 randomly-chosen instances for each of these semantic relations: AtLocation (noun to noun), PartOf (noun to noun), HasA (noun to noun), and MadeOf (noun to noun). Other types of relations were left out, including relations where few instances actually mapped to WordNet senses, relations where adjectives and adverbs were implied (WordNet does not provides a taxonomy for these), and relations where verbs were implied, since we found that properly annotating these instances is really hard and error-prone due to the extremely fine-grained definition of verb senses in WordNet.

Only instances where both concepts could be directly mapped to WordNet were considered for training and validation. Only instances with both concepts mapping to more than one sense were selected for validation. Some concepts were annotated to more than one sense due to the fine-grained nature of WordNet [47]. Despite the aforementioned limitations, we think that the included relations are representative enough to draw some initial conclusions, which was the goal of our experiment[1].

We set the same parameters used in our experiments with WordNet described in the previous section. We also compared the results obtained by our technique with those obtained by the methods used in our previous batch of experiments.

Table 2 summarizes our results. The ConceptNet experiment shows that NASARI distance-based approach and our method achieve better performance scores than the other methods in the disambiguation of the hand-annotated instances. It can be seen that the most frequent sense approach performs better in this disambiguation task with respect to the results it obtained in the WordNet disambiguation task. The closest senses heuristic also obtains slightly better results than in the previous task. However, the results obtained by our method are pretty consistent with those obtained in our WordNet experiments. As expected, NASARI improves the results it obtained in the WordNet test. In fact, NASARI and EPROP obtain the best results in our experiments with ConceptNet. NASARI obtains the best results for the PartOf and HasA relations, whereas EPROP obtains the best results for the AtLocation and MadeOf relations.

7 Application to WSD

In the previous experiments, we have shown how our technique can be satisfactorily applied to disambiguate ambiguous instances of relational knowledge bases using only the WordNet taxonomy. In this Section, we show how these disambiguated instances can be used to improve the results obtained by knowledge-based WSD techniques.

In this additional experiment, we show how relations extracted using our approach can improve the results obtained by Personalized PageRank (PPR, [23]), also known as UKB, with its default original configuration[2]. PPR uses a knowledge graph built from WordNet relations, which is comprised of 367576 semantic relations. In order to keep the experimentation simple, we augmented this graph by only disambiguating the ConceptNet relation *RelatedTo* between nouns, using the parameters and procedures described in the previous section, and choosing those relations with a probability equal or higher than 0.8, which has been chosen as a reasonable value, and omitting non-ambiguous relations. This led to the inclusion of 31624 new unique semantic relations using EPROP. The Personalized PageRank using our extended knowledge graph is referred as PPR+EPROP. We also consider the extended knowledge graph built by the original PPR authors based on the annotated glosses of senses, where a relation is added from a sense to all the senses that appear in its description. This version of the algorithm is called PPR+glosses. Finally, since NASARI obtained competitive results in the ConceptNet test, we have also applied NASARI to extend the knowledge graph of PPR, as we did with our proposed method. We called this approach PPR+NASARI.

In order to evaluate the performance of the compared approaches, we used the

[1]The hand-annotated instances we used in our experiment are available at `http://noesis.ikor.org/datasets/sense-disambiguation`.

[2]We used the version 3.0 of the original implementation of PPR, which can be downloaded from `http://ixa2.si.ehu.es/ukb/`.

validation framework provided by [48]. For a better illustration of the obtained performance, we include the state-of-the-art knowledge-based WSD techniques considered in that work, where more detailed descriptions of the different WSD methods can be found. The validation framework evaluates the current state-of-the-art WSD techniques according to senseval and semeval tasks of different years, datasets which are considered to be *de facto* standards in the evaluation of WSD techniques. We considered the results and scorer provided by this framework for all the included methods except for PPR-based techniques, which were recomputed to carry out our experimentation. Since we used EPROP just to add relations between nouns for simplicity, we restricted our experimentation to noun disambiguation.

Method	**senseval2**	**senseval3**	**semeval2007**	**semeval2013**	**semeval2015**	**Average**
Lesk$_{ext}$	60.37	49.85	40.70	53.64	53.99	51.71
Lesk$_{ext}$ + emb	74.58	**72.67**	**66.04**	66.24	67.80	69.46
WordNet 1st sense	72.05	72.00	65.41	62.96	66.29	67.74
Babelfy	74.02	66.67	61.01	**66.42**	**69.87**	67.60
PPR	77.39	71.22	65.41	64.31	69.11	69.49
PPR+glosses	75.05	71.56	65.41	64.48	69.30	69.16
PPR+NASARI	75.99	72.11	65.41	64.86	68.93	69.46
PPR+EPROP	**77.49**	71.00	**66.04**	64.37	69.68	**69.72**

Table 3: F1 scores of different state-of-the-art knowledge-based WSD techniques in the task of noun disambiguation on senseval and semeval datasets.

The obtained F1 scores for each method and dataset combination are shown in Table 3. We can see how, with the exception of Lesk$_{ext}$, all methods obtained close F1 scores despite the fact that they follow different algorithmic approaches and use different knowledge-based information. Once Lesk$_{ext}$ is kept out, which offers worse results despite its relatively widespread use in practice (probably due to its simplicity), a non-parametric Friedman's test with aligned ranks over the F1 scores for the different validation datasets shows no significant difference. Even by following very different approaches, different WSD techniques exhibit the typical behaviour of machine learning algorithms, a well-known phenomenon informally known as the 'no free lunch theorem' [49]. No significant difference, however, does not mean unimportant. PPR is the approach that obtains the best results. The addition of new semantic relations does not guarantee improved results, as shown by the inclusion of relations extracted from the annotation of senses glosses and the relations extracted by the NASARI technique. However, the semantic relations obtained using our approach led to the best overall results if the average F1 score is considered. These results suggest that EPROP has the potential to improve the performance of knowledge-based WSD techniques by allowing them to automatically expand their knowledge by the disambiguation of available large scale relational knowledge bases.

8 Conclusions

Our main contribution in this work is the proposal of a novel technique for WSD in ambiguous semantic relations. The proposed technique outperformed baseline techniques and a state-of-the-art method in most of our experiments, obtaining remarkable results in different experiments and hinting its potential for the WSD problem. Since our technique only requires a taxonomy of senses and non-annotated data to predict the right sense of the terms involved in a semantic relation, it can be used to automatically enhance the knowledge bases used by knowledge-based WSD techniques, with the potential to improve their performance as suggested by our experiments. The unsupervised nature of our approach alleviates the knowledge acquisition bottleneck, a major problem in the field of NLP.

This work serves as an introduction to the proposed evidence aggregation technique, which, as described, still contains several practical limitations. WordNet does not provide taxonomies for adjectives and adverbs, limiting the applicability of our technique to nouns and verbs unless alternative taxonomies are made widely available. Future work will include attempts to build these taxonomies from alternative data sources and will test how well they perform when used in conjunction with our technique.

In this work, we have shown, using different experiments, that the proposed technique is feasible for improving the performance of WSD techniques, with potential applications to real-world WSD problems. We plan to extend our WSD approach to make it widely available as part of a toolbox for NLP and text mining practitioners.

Acknowledgments

This work is partially supported by the Spanish Ministry of Economy and the European Regional Development Fund (FEDER), under grant TIN2012-36951 and the program "Ayudas para contratos predoctorales para la formación de doctores 2013" (grant BES-2013-064699).

References

[1] Mark Stevenson and Yorick Wilks. Word sense disambiguation. *The Oxford Handbook of Comp. Linguistics*, pages 249–265, 2003.

[2] Roberto Navigli. A quick tour of word sense disambiguation, induction and related approaches. In *Proceedings of the 38th Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2012*, pages 115–129. Springer, 2012.

[3] Jason C. Hung, Ching-Sheng Wang, Che-Yu Yang, Mao-Shuen Chiu, and George Yee. Applying word sense disambiguation to question answering system for e-Learning. In *Proceedings of the 19th International Conference on Advanced Information Networking and Applications, (AINA 2005)*, volume 1, pages 157–162. IEEE, 2005.

[4] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 1156–1165, 2014.

[5] Vassiliki Rentoumi, George Giannakopoulos, Vangelis Karkaletsis, and George A Vouros. Sentiment analysis of figurative language using a word sense disambiguation approach. In *Recent Advances in Natural Language Processing, RANLP 2009*, pages 370–375, 2009.

[6] Erik Cambria, Björn W. Schuller, Yunqing Xia, and Catherine Havasi. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, 28(2):15–21, 2013.

[7] Eneko Agirre and Philip Edmonds. *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media, 2007.

[8] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

[9] Yoong Keok Lee, Hwee Tou Ng, and Tee Kiah Chia. Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In *Senseval-3: third international workshop on the evaluation of systems for the semantic analysis of text*, pages 137–140, 2004.

[10] Yee Seng Chan, Hwee Tou Ng, and Zhi Zhong. NUS-PT: exploiting parallel texts for word sense disambiguation in the english all-words tasks. In *Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval ACL 2007*, pages 253–256. Association for Computational Linguistics, 2007.

[11] Douglas B. Lenat, Mayank Prakash, and Mary Shepherd. CYC: using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine*, 6(4):65–85, 1986.

[12] William A. Gale, Kenneth W. Church, and David Yarowsky. Using bilingual materials to develop word sense disambiguation methods. In *Proceedings of the International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 101–112. Citeseer, 1992.

[13] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics, 26-30 June 1995, MIT, Cambridge, Massachusetts, USA, Proceedings.*, pages 189–196, 1995.

[14] Simone Paolo Ponzetto and Roberto Navigli. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010*, pages 1522–1531. Association for Computational Linguistics, 2010.

[15] Christopher D Manning, Hinrich Schütze, et al. *Foundations of statistical natural language processing.* MIT Press, 2001.

[16] Ted Pedersen. Unsupervised corpus-based methods for WSD. *Word sense disambiguation: algorithms and applications*, pages 133–166, 2006.

[17] Rada Mihalcea. Knowledge-based methods for WSD. *Word Sense Disambiguation: Algorithms and Applications*, pages 107–131, 2006.

[18] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation, SIGDOC 1986*, pages 24–26. ACM, 1986.

[19] Satanjeev Banerjee and Ted Pedersen. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing, CICLing 2002*, pages 136–145. Springer, 2002.

[20] Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. An enhanced Lesk word sense disambiguation algorithm through a distributional semantic model. In *Proceedings of the 25th International Conference on Computational Linguistics, COLING 2014*, pages 1591–1600, 2014.

[21] Rada Mihalcea, Paul Tarau, and Elizabeth Figa. PageRank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20th International Conference on Computational Linguistics Proceedings of the Conference, COLING 2004*. Association for Computational Linguistics, 2004.

[22] Ravi Sinha and Rada Mihalcea. Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity. In *Proceedings of the First IEEE International Conference on Semantic, Computing (ICSC 2007)*, pages 363–369. IEEE, 2007.

[23] Eneko Agirre and Aitor Soroa. Personalizing PageRank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2009*, pages 33–41. Association for Computational Linguistics, 2009.

[24] Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84, 2014.

[25] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

[26] Eneko Agirre and David Martínez. Learning class-to-class selectional preferences. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning, CoNLL 2001*, page 3. Association for Computational Linguistics, 2001.

[27] Diana McCarthy and John Carroll. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654, 2003.

[28] William A. Gale, Kenneth W. Church, and David Yarowsky. One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*, pages 233–237. Association for Computational Linguistics, 1992.

[29] David Yarowsky. One sense per collocation. In *Proceedings of the workshop on Human Language Technology*, pages 266–271. Association for Computational Linguistics, 1993.

[30] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990.

[31] George A Miller. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[32] Roberto Navigli and Simone Paolo Ponzetto. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 216–225. Association for Computational Linguistics, 2010.

[33] Roberto Navigli and Simone Paolo Ponzetto. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[34] Myunggwon Hwang, Chang Choi, and Pan Koo Kim. Automatic enrichment of semantic relation network and its application to word sense disambiguation. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):845–858, 2011.

[35] P. Chen, C. Bowes, W. Ding, and M. Choly. Word sense disambiguation with automatically acquired knowledge. *IEEE Intelligent Systems*, 27(4):46–55, July 2012.

[36] Robert Speer and Catherine Havasi. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012*, pages 3679–3686, 2012.

[37] Robert Speer and Catherine Havasi. Conceptnet 5: A large semantic network for relational knowledge. In *The Peoples Web Meets NLP*, pages 161–176. Springer, 2013.

[38] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. YAGO3: A knowledge base from multilingual wikipedias. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*, 2015.

[39] Junpeng Chen and Juan Liu. Combining ConceptNet and WordNet for word sense disambiguation. In *Proceedings of the Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011*, pages 686–694, 2011.

[40] Junpeng Chen and Wei Yu. Enriching semantic knowledge for WSD. *IEICE TRANSACTIONS on Information and Systems*, 97(8):2212–2216, 2014.

[41] Edward H. Shortliffe and Bruce G. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23(3-4):351–379, 1975.

[42] David Heckerman. Probabilistic interpretations for MYCIN's certainty factors. *arXiv preprint arXiv:1304.3419*, 2013.

[43] James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

[44] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[45] Ellen M. Voorhees. The TREC-8 question answering track report. In *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19*, 1999.

[46] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64, 2016.

[47] Martha Palmer, Hoa Trang Dang, and Christiane Fellbaum. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, 13(2):137–163, 2007.

[48] Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*, pages 99–110, 2017.

[49] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

2.3 An automorphic distance metric for node role discovery

The technical report associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. An automorphic distance metric and its application to node embedding for role mining. ArXiv e-prints 1712.06979, December 2017.

An Automorphic Distance Metric and its Application to Node Embedding for Role Mining

Víctor Martínez   Fernando Berzal   Juan-Carlos Cubero

Abstract

Role is a fundamental concept in the analysis of the behavior and function of interacting entities represented by network data. Role discovery is the task of uncovering hidden roles. Node roles are commonly defined in terms of equivalence classes, where two nodes have the same role if they fall within the same equivalence class. Automorphic equivalence, where two nodes are equivalent when they can swap their labels to form an isomorphic graph, captures this common notion of role. The binary concept of equivalence is too restrictive and nodes in real-world networks rarely belong to the same equivalence class. Instead, a relaxed definition in terms of similarity or distance is commonly used to compute the degree to which two nodes are equivalent. In this paper, we propose a novel distance metric called automorphic distance, which measures how far two nodes are of being automorphically equivalent. We also study its application to node embedding, showing how our metric can be used to generate vector representations of nodes preserving their roles for data visualization and machine learning. Our experiments confirm that the proposed metric outperforms the RoleSim automorphic equivalence-based metric in the generation of node embeddings for different networks.

1 Introduction

Role discovery is defined as the process of finding sets of nodes following similar connectivity patterns or structural behaviors [1]. The role of a node can be understood as the function that node plays in the network. Different studies have shown the importance of roles in different domains, including predator-prey food webs [2], international relations [3], or the function of proteins in proteomes [4].

Unfortunately, this problem has received limited attention when compared to community detection [5, 6, 7], despite the fact that role discovery identifies complementary information and has found application in several useful network data mining tasks. For example, roles can be used to model and characterize the behaviors of entities in a network to predict structural changes and detect anomalies [8]. Since the same roles can be observed across different networks, this information has been successfully exploited for transfer learning [9]. Role information can also be used for enhanced visualization of interesting patterns in graphs [10]. Additional applications of role discovery are covered in more detail in [1].

Formally, two nodes have the same role if, given an equivalence relation, they belong to the same equivalence class [11, 12]. Different equivalence classes haven been studied for nodes in networks.

Structural equivalence, where two nodes play the same role if they are connected to exactly the same neighbor nodes, has been widely studied [13, 14]. These nodes will have exactly the same topological properties, such as degree, clustering coefficient, or centrality, since they are indistinguishable from a structural point of view. However, different authors have pointed out the limitations of structural equivalence for modeling roles or positions (the name that roles receive in sociology), since structural equivalence is more related to the concept of locality than the actual concept of role [15]. If the constraint of needing to be connected to exactly the same neighbors is relaxed to being connected to neighbors with exactly the same topological function, we obtain automorphic equivalence classes, where two nodes are equivalent if they can swap their labels to form an isomorphic graph [16, 17]. Automorphically equivalent nodes will also have exactly the same topological properties but, without the requirement of locality imposed by structural equivalence, pairs of nodes at distances larger than two can still have the same role. Therefore, automorphic equivalence is more closely related to the intuitive concept of role, understood as the function of a node within a network.

Other equivalence classes, less relevant than the ones previously mentioned, are not covered in this work. Regular equivalence deserves special mention due to its importance as a relaxation of automorphic equivalence that only requires being connected to nodes with the same function, omitting the actual count of connections [18]. Regular equivalence does not preserve topological properties and is more suited to hierarchically-organized networks [2].

These binary equivalences are strict mathematical abstractions that rarely occur in real-world networks, leading to all nodes being assigned a different role. In practice, these equivalences are relaxed to similarities, allowing two nodes to play the same role by partially satisfying the constraints imposed by the mathematical definition of structural, automorphic, or regular equivalence.

In this paper, we present a novel automorphic distance metric, capturing distances between nodes in terms of automorphic equivalence. According to the network structure, two nodes will be at a distance that is proportional to how far they are from being automorphically equivalent. This leads to a softer definition of automorphic roles, instead of forcing all nodes to fit in strict classes of roles. However, when needed, these distances can be used to discover and instantiate specific role classes. Our distance function satisfies metric axioms, as we prove below, does not require external parameters nor feature engineering, and is computable for nodes across different networks. We also present different applications of our proposal, with special emphasis on generating node embeddings that preserve node roles. Node embeddings are vector representations of nodes capturing relevant information in terms of pair-wise distances [19]. Much work has been done in embedding techniques that preserve neighborhoods or communities [20, 21], but, to the best of our knowledge, our work is the first one on role-preserving embeddings.

Our paper is structured as follows. In Section 2, we discuss the relevant related work. In Section 3, we describe our proposed automorphic distance metric and study its admissibility as a distance metric, as well as its computational complexity. In Section 4, we analyze its application to the generation of node embeddings that preserve roles and show how it outperforms previously proposed approaches. Finally, conclusions and suggestions for future research are presented in Section 5.

2 Related Work

Different metrics have been proposed to measure node similarity. One of the most popular metrics is SimRank [22], which iteratively computes similarity scores based on the hypothesis that two nodes are similar if they link to similar nodes. Different extensions of SimRank have been proposed [23]. SimRank recursively computes similarity of two nodes according to the average similarity of all their neighbor pairs, which can also be interpreted, as suggested by its original authors, as how soon two random walkers will meet if they start from these nodes. Thus, this definition is not suitable as a metric of similarity capturing automorphic equivalence because it requires the two nodes to be close to play the same role. Other similarity measures not based on SimRank have been proposed, such as PageSim [24] and Leicht's vertex similarity [25]. These similarities are formally rejected as valid metrics for capturing automorphic equivalence in [26].

Since automorphic equivalence ensures the same topological properties, some authors have tried to capture automorphic equivalence by defining a similarity function over a set of network topological properties [27]. The problem of these feature-based methods is that they require combining different complex hand-crafted features obtained by experts, which is far from trivial in practice. In addition, they cannot guarantee which set of features will correctly approximate automorphic equivalence, leading to a very limited approach to automorphic equivalence discovery.

As far as we know, RoleSim [28, 26] is the only proposed metric that tries to formally capture the concept of automorphic equivalence without using limited approximations based on hand-crafted topological features. Omitting the decay factor they introduce, by setting it to 0 in order to capture the global network topology, this similarity measure is iteratively computed until convergence as

$$s(x, y) = \max_{M(x,y)} \frac{\sum_{(u,v)\in M(x,y)} s(u, v)}{deg(x) + deg(v) - |M(x, y)|}$$

where $deg(n)$ is the degree of a node n and $M(x, y)$ is the optimal assignment of nodes in the neighborhood of x to nodes in the neighborhood of y maximizing the expression; that is, the pairs of neighbors of x and y with maximal similarity. In their work, Jin et al. prove that this function satisfies the axioms required to be considered a valid role similarity metric. RoleSim is a form of generalized Jaccard coefficient based on a recursive definition of the similarity of neighbor roles. Despite the admissibility of RoleSim, their approach presents several limitations. The RoleSim similarity can be considered an automorphic distance by taking its complementary or Jaccard distance: $d(x, y) = 1 - s(x, y)$. The problem is that the Jaccard coefficient is a normalized metric, which leads to a normalized distance. As will be shown in our experimentation, this normalization has a negative impact on the results obtained by RoleSim. In addition, this similarity function exhibits serious inconsistencies. For example, in the graph shown in Figure 1, where node d has a one-to-many relationship to x_i nodes, the node pair (a, c) has the same exact similarity than any pair (a, x_i), independently of the number of x_i nodes. This simple example shows the limitations of RoleSim when trying to capture automorphic similarity.

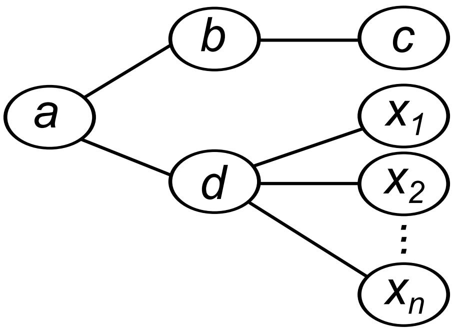


Figure 1: Example graph where RoleSim yields inconsistent values. Node d has a one to many relation to x_i nodes.

As far as we know, no distance metric has been proposed that is able to capture the concept of automorphic distance in a consistent way, without relying on approximations based on extracted topological properties nor forcing the normalization of the distance function.

3 A Novel Automorphic Distance Metric

An isomorphism is a bijection between the nodes of two graphs where two nodes are adjacent in one graph if and only if the nodes that result from applying the bijective function are also adjacent in the other graph. An automorphism is an isomorphism from one graph to itself. Therefore, two nodes are automorphically equivalent if there exists an automorphism creating a correspondence between them.

One form of testing for automorphic equivalence is computing the canonical form of graphs. Graph canonization is the task of computing a labeling for nodes in a graph such that every isomorphic graph yields the same canonical labeling. Given a canonized graph, two automorphically-equivalent nodes must have been assigned the same label. As previously stated, automorphic equivalence is too restrictive to appear in real-world networks, leading to most nodes having different canonical labels.

The solution that we propose to this problem is the definition of distances between labels, which ultimately allows the definition of distances between nodes based on the concept of automorphic equivalence. This distance will be proportional to the number of changes that need to be done in the network to transform one label or equivalence class into another. A zero distance implies that two nodes are automorphically equivalent and play exactly the same role. According to this distance d, we can say that nodes x and y are more automorphically similar or have a more similar role than u and v if $d(x, y) < d(u, v)$. In order to propose a valid distance metric, we must also prove that our metric satisfies the distance metric axioms.

Our work is based on the 1-dimensional Weisfeiler-Lehman test of isomorphism [29, 30], also known as color refinement, which is an algorithm to compute the canonical labeling of graphs. These canonical labels can be used to solve related problems, such as the computation of efficient graph kernels [31]. The Weisfeiler-Lehman algorithm works by initially assigning a label to each node according to its degree, so nodes with the same degree have the same initial label. Then, the algorithm iteratively updates these labels by the following procedure. First, it takes the labels from neighbor nodes, concatenates them according to certain arbitrary order (the same ordering must be applied for all nodes), and finally appends the label of the node at the beginning of the obtained list. Each different sequence is substituted by a newly generated unique label so nodes exhibiting exactly the same sequence are assigned the same label. This refinement process is repeated until labels stabilize, that is, when every pair of nodes with the same label in the previous iteration have the same label in the current iteration. Therefore, after m iterations, which depend on the network diameter, the canonical form is achieved and an additional iteration is required for testing the stabilization condition. These final labels are the canonical form of the graph and, therefore, two nodes with the same final label are automorphically equivalent. An example of running the algorithm in a simple graph is shown in Figure 2.

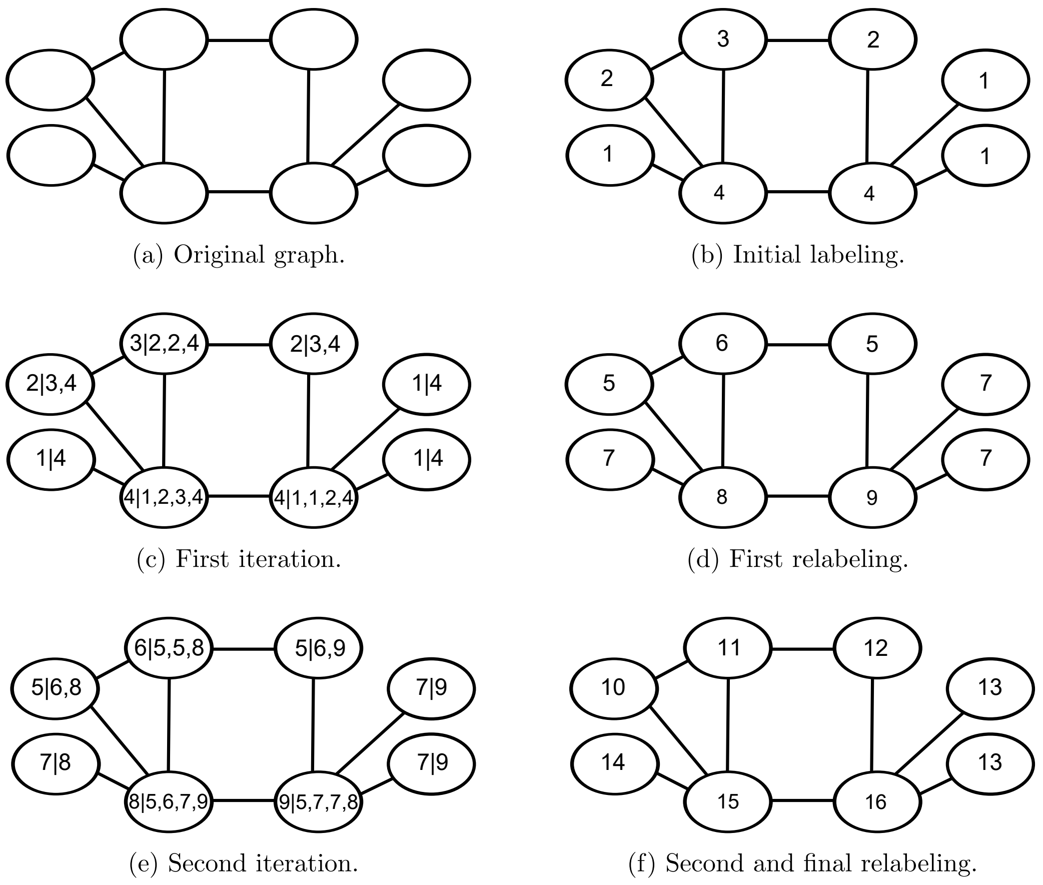
It can be noted that some pairs of the labels appearing in the same iteration of the Weisfeiler-Lehman algorithm are more similar than others. The automorphic distance between two nodes can be defined as the distance between their canonical labels. We propose a scheme to compute distances between the labels that are obtained by the Weisfeiler-Lehman algorithm. Since distances are only defined for labels appearing in the same iteration, a special label associated to nodes of degree 0, which we call the empty label $\ell_\emptyset$, is considered for convenience. Isolated nodes are directly assigned this label and left out of the iterative process.

Since labels created in the initial assignment are based on node degree, we define the distance of labels of nodes x and y as the number of links that must be added to or removed from node x to transform it into node y. This can be easily computed as their absolute degree difference:

$$d(\ell_0(x), \ell_0(y)) = |deg(x) - deg(y)|, \tag{1}$$

where $\ell_0(n)$ is the initially-assigned label to node n. This definition of distance for initial labels is also valid for isolated nodes, with degree 0, which have been assigned the empty label.

Given these distances for initial labels, the distance between labels for the subsequent iterations can be computed as the distance of the optimally-matched pairs of labels of their neighbors from the previous iteration. The distance of labels from the i-th iteration can be computed as



(a) Original graph. (b) Initial labeling.

(c) First iteration. (d) First relabeling.

(e) Second iteration. (f) Second and final relabeling.

Figure 2: The Weisfeiler-Lehman canonization algorithm applied to a simple graph.

$$d(\ell_i(x), \ell_i(y)) = \min_{M_{i-1}(x,y)} \sum_{(u,v)\in M_{i-1}(x,y)} d(\ell_{i-1}(u), \ell_{i-1}(v)), \tag{2}$$

where $M_{i-1}(x, y)$ is the optimal assignment of neighbors of x to neighbors of y that minimizes the expression and, therefore, it is just the sum of distances between neighbors of x and y. If the neighborhood of one node is larger than the neighborhood of the other, leading to unmatched nodes, these nodes are directly matched with virtual nodes, which are labeled with $\ell_\emptyset$. The distances of unmatched nodes to the empty label can be seen as the cost, in terms of distance, of inserting and transforming a virtual isolated node to obtain a node with the label of the unmatched node. The optimal assignment, which would consider $O(n!)$ alternatives using a naive brute force approach, can be computed in polynomial time using the Hungarian algorithm [32].

Initially, Equation 1 and, in subsequent iterations, Equation 2 are used to compute a distance table. At any given time, only distances from two iterations need to be maintained: the distances currently being computed and the distances from the most recent previous iteration.

The described iterative process is carried out for each iteration of the 1-dimensional Weisfeiler-Lehman algorithm until label stabilization. The automorphic distance between a pair of nodes is defined as the distance between their canonical labels. It should be noted that labels can be represented using any set of symbols. However, for simplicity, we represent labels as positive integers.

Algorithm 1 Automorphic Distance Algorithm

```
procedure AUTOMORPHIC DISTANCE
Input: Set of nodes $N$ of an undirected graph.
    Initialize $i = 0, stabilized = false, r = HashTable()$.
    Set $\ell_i(x) = deg(x)\ \forall x \in N$.
    Set $d(\ell_i(x), \ell_i(y)) = |deg(x) - deg(y)|\ \forall x, y \in N \times N$.
    while not $stabilized$ do
        Set $m = i$. Iterate $i = i + 1$.
        Set $h_i(x) = concatenate(sort(neighbors(x)))$.
        Set $c_i(x) = concatenate(\ell_{i-1}(x), h_i(x))$.
        Set $\ell_i(x) = unique(c_i(x))$.
        Use Hungarian algorithm to compute $M_{i-1}(x, y)\ \forall x, y \in N \times N$.
        Set $d(\ell_i(x), \ell_i(y)) = \min_{M_{i-1}(x,y)} \sum_{(u,v) \in M_{i-1}(x,y)} d(\ell_{i-1}(u), \ell_{i-1}(v))\ \forall x, y \in N \times N$.
        Set $stabilized = true$.
        Store in $r$ the label of $x$ in iteration $i - 1\ \forall x \in N$ as $r[\ell_i(x)] = \ell_{i-1}(x)$.
        If the entry $r[\ell_i(x)]$ was already set with a different label, set $stabilized = false$.
    end while
    Set $d(x, y) = d(\ell_m(x), \ell_m(y))$ for each pair of nodes $x, y \in N \times N$.
Output: Pairwise automorphic distances $d(x, y)$ for each pair of nodes $x, y \in N \times N$.
end procedure
```

The complete algorithm is shown in Algorithm 1. The function $neighbors(x)$ returns the set of neighbor nodes of node x. The function $sort(s)$ sorts a set of elements. The ordering among elements is not relevant for the algorithm, but the same ordering must always be applied. The function $concatenate(x_1, \ldots, x_n)$ returns the concatenation of elements $x_1, \ldots, x_n$. Finally, the function $unique(s)$ generates and returns a unique symbol, such an integer, for each observed unique string s, where $unique(s) = unique(s')$ if and only if $s = s'$.

3.1 Case Example

In this section, we show an illustrative example applying the proposed automorphic distance metric to the network shown in Figure 2a.

The proposed algorithm for computing the automorphic distance initially assigns a label to each node according to its degree, as shown in Figure 2b. Therefore, two nodes will have the same label if and only if they have the same degree. The initial distance table, represented as an upper triangular matrix due to the symmetry of distances, as will be proved in Section 3.2.3, is shown in Table 1. This distance table is computed using Equation 1 according to the initial label assignations. For example, the distance between the labels

1 and 4 is 3, since this value is the absolute degree difference of the corresponding nodes.

$\boldsymbol{\ell/\ell}$	$\boldsymbol{\ell_\emptyset}$	**1**	**2**	**3**	**4**
$\boldsymbol{\ell_\emptyset}$	0	1	2	3	4
1		0	1	2	3
2			0	1	2
3				0	1
4					0

Table 1: Initialization of the distance table.

After the initialization, the algorithm enters into its main loop and performs its first iteration. For each node, the labels of its neighbors are ordered and concatenated with its own label, as shown in Figure 2c. For example, the only node with label 3 has the associated string $3|2,2,4$, since its neighbors have labels 2, 4, and 2. These concatenated strings are replaced by a new label, chosen so that two nodes are assigned the same new label if and only if they had the same concatenated string. This process generates a new labeling as shown in Figure 2d. Given these new labels, the algorithm computes their pairwise distances using Equation 2, which are shown in Table 2.

$\boldsymbol{\ell/\ell}$	$\boldsymbol{\ell_\emptyset}$	**5**	**6**	**7**	**8**	**9**
$\boldsymbol{\ell_\emptyset}$	0	9	11	5	14	12
5		0	3	3	3	3
6			0	4	2	2
7				0	6	4
8					0	2
9						0

Table 2: Distance table after the first relabeling.

For example, in order to compute the distance between labels 5 and 9, the optimal assignment between their neighbors minimizing the summation of distances, according to the previous iteration, must be obtained. In the previous iteration, 3 and 4 were the labels of the neighbors of nodes with label 5. Likewise, 1, 1, 2, and 4 were the labels of the neighbors of nodes with label 9. The Hungarian algorithm matches these neighbor labels to minimize their sum of distances: $(3,2)$ and $(4,4)$ according to Table 1. Since the two neighbor labels 1 of label 9 were left unmatched, they are both matched with the empty label as $(1,\ell_\emptyset)$. Given this optimal assignment, we can compute the distance between labels 5 and 9 as:

$$d_1(5,9) = d_0(3,2) + d_0(4,4) + d_0(1,\ell_\emptyset) + d_0(1,\ell_\emptyset) = 1 + 0 + 1 + 1 = 3$$

Following this iterative process, the algorithm performs the second iteration. Concatenated strings are computed as shown in Figure 2e and labels are updated as shown in Figure 2f. It can be easily seen that labels have stabilized, obtaining the canonical labeling of this graph. The stabilization condition can be tested by performing an additional iteration and observing that nodes with label 13 are assigned the same label, while the other nodes are assigned an unique new label. This new labeling would be equivalent to the labeling obtained in this iteration, the condition required to achieve stabilization. The pairwise distances computed in this iteration are shown in Table 3.

$\boldsymbol{\ell/\ell}$	$\boldsymbol{\ell_\emptyset}$	**10**	**11**	**12**	**13**	**14**	**15**	**16**
$\boldsymbol{\ell_\emptyset}$	0	34	43	32	17	19	51	45
10		0	12	2	16	11	16	13
11			0	14	20	18	10	8
12				0	11	14	14	15
13					0	2	25	21
14						0	28	19
15							0	6
16								0

Table 3: Distance table after the second and final relabeling.

For instance, let us compute the distance between labels 11 and 16. We start by finding the optimal assignment that minimizes the pairwise distances of their neighbor labels in the previous iteration, which are $5, 5$, and 8 for label 11 and $5, 7, 7$, and 8 for label 16. The Hungarian algorithm obtains the optimal matching $(5, 5), (5, 7)$, and $(8, 8)$, with an additional $(\ell_\emptyset, 7)$, due to the difference of the node degrees associated to labels 11 and 16. Once this optimal matching has been obtained, we can easily compute the distance between labels 11 and 16 using Equation 2 as:

$$d_1(11, 16) = d_1(5, 5) + d_1(5, 7) + d_1(8, 8) + d_1(\ell_\emptyset, 7) = 0 + 3 + 0 + 5 = 8$$

The automorphic distance between a pair of nodes is defined as the distance between their canonical labels, which are the final labels assigned in the iteration where stabilization is achieved. Therefore, in our example, the distance between nodes is given by Table 3. For example, we can see how nodes with canonical labels 13 and 14 are close to being automorphically equivalent, since their automorphic distance is only 2. In contrast, nodes with canonical labels 14 and 15 have a large automorphic distance, equal to 28, which indicates that they are far from being automorphically equivalent.

3.2 Metric Admissibility

In this section, we prove that the distance function that we have defined is a valid metric or distance function. In order to assert this statement, we must prove the following four conditions: non-negativity, identity of indiscernibles, symmetry, and triangle inequality.

To prove these conditions, we note that Equation 2 can be recursively decomposed as

$$\begin{aligned}
d(\ell_m(x), \ell_m(y)) &= \min_{M_{m-1}(x,y)} \sum_{(u,v)\in M_{m-1}(x,y)} d(\ell_{m-1}(u), \ell_{m-1}(v)) \\
&= \min_{\substack{M_{m-1}(x,y) \\ M_{m-2}(u,v)}} \sum_{\substack{(u,v)\in \\ M_{m-1}(x,y)}} \sum_{\substack{(u',v')\in \\ M_{m-2}(u,v)}} d(\ell_{m-2}(u'), \ell_{m-2}(v')) \\
&= \quad \dots \\
&= \sum_{(x',y')\in M'(x,y)} d(\ell_0(x'), \ell_0(y')) \\
&= \sum_{(x',y')\in M'(x,y)} |deg(x') - deg(y')|
\end{aligned} \tag{3}$$

where $M'(x, y)$ is the set of pairs of nodes that appear in the recursive summation at the deepest level of recursion as a result of choosing the optimal assignment in each iteration.

3.2.1 Proof of Non-Negativity

Non-negativity requires that the distance function satisfies $d(x, y) \geq 0$ for any possible pair of nodes x and y. Given the decomposition of our metric as shown in Equation 3, it is straightforward to see that the summation of absolute values is guaranteed to be always equal or greater than 0.

3.2.2 Proof of the Identity of Indiscernibles

The identity of indiscernibles implies that the distance function satisfies $d(x, y) = 0$ if and only if $x \equiv y$. The Weisfeiler-Lehman algorithm guarantees that automorphically equivalent pairs of nodes are assigned the same canonical label, and non-automorphically equivalent pairs of nodes are assigned different canonical labels.

Equation 3 is only equal to zero when $deg(x') = deg(y')$ for every pair of nodes in $M'(x, y)$. Two nodes can only have the same canonical label if they are automorphically equivalent, as guaranteed by the Weisfeiler-Lehman algorithm. If two nodes are assigned the same canonical label, their neighbors must have been assigned equivalent labels in all the iterations. Since the distance of a label to itself is 0, we can see that the summation yields 0, leading to a zero distance for nodes when $x \equiv y$.

On the other side, when two nodes have different canonical labels, their neighbors must have been assigned different labels during the execution of the algorithm. This implies that, in the recursive decomposition shown in Equation 3, at least one pair of nodes will not match nodes with the same initial labels, leading to a distance greater than 0 for nodes $x \not\equiv y$.

3.2.3 Proof of Symmetry

The condition of symmetry requires that the proposed distance function must satisfy the property $d(x, y) = d(y, x)$. We can prove that Equation 3 is symmetric, as

$$\begin{aligned} d(\ell_m(x), \ell_m(y)) &= \sum_{(x',y') \in M'(x,y)} |deg(x') - deg(y')| \\ &= \sum_{(y',x') \in M'(y,x)} |deg(y') - deg(x')| \\ &= d(\ell_m(y), \ell_m(x)) \end{aligned}$$

since $M'(x, y)$ is equal to $M'(y, x)$ when we swap the nodes. The order of the nodes in each pair does not affect the result of our distance function.

3.2.4 Proof of the Triangle Inequality

The triangle inequality requires that the inequality $d(x, y) \leq d(x, z) + d(z, y)$ is satisfied by the proposed automorphic distance function.

By the definition of the proposed distance, we know that

$$\begin{gathered} d(x, y) \leq d(x, z) + d(z, y) \implies \\ d(\ell_m(x), \ell_m(y)) \leq d(\ell_m(x), \ell_m(z)) + d(\ell_m(z), \ell_m(y)) \implies \\ \sum_{(a,b) \in M'(x,y)} |deg(a) - deg(b)| \leq \sum_{(a,c) \in M'(x,z)} |deg(a) - deg(c)| + \sum_{(c,d) \in M'(z,y)} |deg(c) - deg(d)|. \end{gathered}$$

We know that the absolute value satisfies the triangle inequality, thus the lower value that the right side can take is

$$\sum_{(a,b) \in M'(x,y)} |deg(a) - deg(b)| \leq \sum_{(a,d) \in M''(x,y,z)} |deg(a) - deg(d)|$$

where $M''(x, y, z)$ is the set of pairs resulting from chaining or combining $M'(x, z)$ and $M'(z, y)$ so that $(a, d) \in M''(x, y, z)$ if and only if, $(a, c) \in M'(x, z)$ and $(c, d) \in M'(z, y)$.

For this inequality to hold, it requires the non-existence of a pairing of nodes better than the matching done in the left side. Since the Hungarian algorithm ensures that matchings are optimal, minimizing their sum of distances, the matching at the right side cannot be better than optimal and, therefore, the value of the right side can only be equal to or greater than the value on the left side, satisfying the triangle inequality condition.

3.3 Metric Computational Complexity

In this section, we analyze the temporal and spatial computational complexity of our proposed metric.

The initialization of labels based on the degree of nodes has $O(n)$ temporal and spatial complexity, where n is the number of nodes in the network. The computation of the table for the initial distances has $O(n^2)$ temporal and spatial complexity, since the distance is computed for every pair of nodes.

Each iteration of the algorithm requires computing the sorted list of labels of the neighbors for each node. This task can be accomplished for each node with computational and spatial complexity $O(k)$, where k is the degree of the node, when using radix, bucket, or counting sort. Thus, computing these strings for all nodes has $O(nk)$ temporal and spatial complexity. Renaming these labels can be done in $O(n)$ time using hash-based data structures. To compute the pairwise distances between labels in the current iteration, the Hungarian algorithm, with computational complexity $O(k^3)$, must be computed for each pair of nodes, leading to $O(n^2k^3)$ temporal complexity and $O(n^2)$ spatial complexity. Finally, checking if the labels have stabilized can be done in $O(n)$ using a hash-based index.

The number of iterations, m, required for 1-dimensional Weisfeiler-Lehman algorithm to converge is closely related to the diameter of the network [30]. Even though the number of iterations is theoretically bounded by n, it has been widely observed that real-world networks tend to exhibit the small-world phenomenon, presenting a small diameter [33, 34] and leading to a small number of iterations required for convergence.

Therefore, by combining these partial results, the total temporal complexity is

$$O(n + n^2 + mn^2k^3) \approx O(mn^2k^3),$$

where n is the number of nodes in the graph, k is the degree of nodes, and m is the number of iterations required for convergence. The spatial complexity of the algorithm is $O(n^2)$, since only the distances and labels from the previous and the current iteration must be maintained at any given time. In practice, the algorithm can handle large networks, given that m and k tend to remain small in real-world networks due to their sparse and small-world nature. In addition, most of these steps can be easily parallelized, since most of them are independent for each node and are only based on the results from the previous iteration. For example, the initial labels for each node can be assigned independently. Once we have assigned these labels, the computation of their pairwise distances can be split among the available processors, since they are independent. The iterative assignment of labels in each loop iteration can be also parallelized using a concurrent data structure to ensure that the new labels are properly generated. Furthermore, the pairwise distances between these new labels can be easily computed in a parallel way, since they are completely independent as they only rely on the distance table computed in the previous iteration.

4 Experimental Evaluation

The experimental evaluation of role discovery techniques is a complicated task due to the lack of available evaluation datasets. Most role mining research projects use private datasets, which are not released and made publicly available to the scientific community. To overcome this limitation and perform an illustrative comparison of automorphic distances, we propose a novel experimental evaluation, which is presented in this section. As a collateral result of this experimentation, a new method for representing node roles using feature vectors is presented.

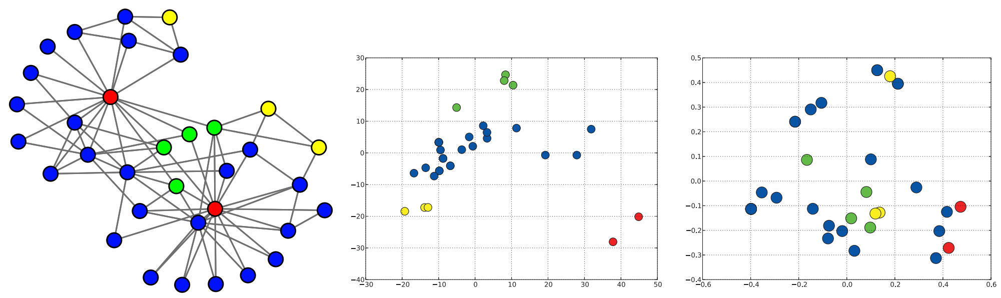
A central problem in machine learning is finding representations that ease the visualization or extraction of useful information from data [35]. A common solution is the computation of embeddings that represent complex objects in a vector space preserving certain properties [36]. Node embedding, also known as graph embedding, is the task of mapping each node in a graph to a dimensional space trying to preserve the similarity or distance between pairs of nodes. Therefore, similar nodes will be located in similar regions of the space. Node embeddings have lately gained attention since they have achieved good results in different machine learning tasks [36]. Several models have been proposed for node embedding. However, these techniques try to preserve the connectivity of the network by obtaining embeddings that preserve structural equivalence, the neighborhood, or the community of nodes [37, 20, 21]. This information has proven to be useful due to the presence of homophily, also known as assortativity, in real-world networks [38], where entities tend to be connected to similar ones, a feature that allows us to explain certain features of the nodes. Even though the connectivity information captured by these techniques is relevant, these techniques fail to capture information related to the role or function of the nodes in the network, which is a highly-valuable information that is complementary to the information obtained by locality-based embedding techniques.

We propose a new kind of embedding by exploiting this complementary information. Our node embeddings capture the roles of nodes by placing nodes that play a similar function in the network close in the resulting vector space. To compute these embeddings, we apply the classical multidimensional scaling (MDS) [39, 40] to the distance matrices computed by the different approaches. In the following section, we show how 2-dimensional embeddings can capture relevant information in different real-world networks. We compare the results obtained by our distance metric with the results obtained by RoleSim, which can be interpreted as a distance function as previously described.

4.1 Zachary's Karate Club Network

Zachary's karate club network is a popular social network representing the 34 members (as nodes) of a university karate club and their interactions (as edges) outside the club [41]. During the study carried out by Wayne W. Zachary, a conflict arose between the two club administrators, leading to the split of the club into two groups according to the leader each member decided to follow. For this reason, this network has served as a prototypical case study for community detection algorithms and some network analysis techniques.

If each member is assigned to a binary class according to the leader it decided to follow, Zachary observed that this property is highly homogeneous and assortative. Therefore, nodes tend to be connected to nodes that took the same decision. In this context, previously proposed embedding techniques generate embeddings that clearly separate nodes according to the leader they decided to follow [37]. This information is crucial when the task is related to community detection. However, these embeddings fail, by nature, to reveal the role of each node in the network.



(a) Kamada-Kawai layout. (b) Automorphic embedding. (c) RoleSim embedding.

Figure 3: Zachary's karate club network and its node embeddings, with node role coloring.

To illustrate how our technique captures the roles of nodes, we assigned a class to each node according to objective role-related properties. The two leaders are colored in red. Nodes interacting with the two leaders are colored in green. Nodes not interacting with any of the two leaders are colored in yellow. Finally, the remaining nodes, which interact with only one of the leaders, are colored in blue. Figure 3 shows the network drawn using the Kamada-Kaway layout algorithm [42] and the embeddings obtained by applying our automorphic distance and RoleSim. It can be seen the four classes of roles are linearly separated in the embedding generated using our distance metric. In addition, it can be seen how the two leaders are clearly mapped as outliers, placed significantly apart from the other nodes representing normal members of the club. However, RoleSim fails to clearly separate the different role classes. The two leaders are placed pretty close to normal club members, without capturing their unique function in the network.

4.2 World Trade Network

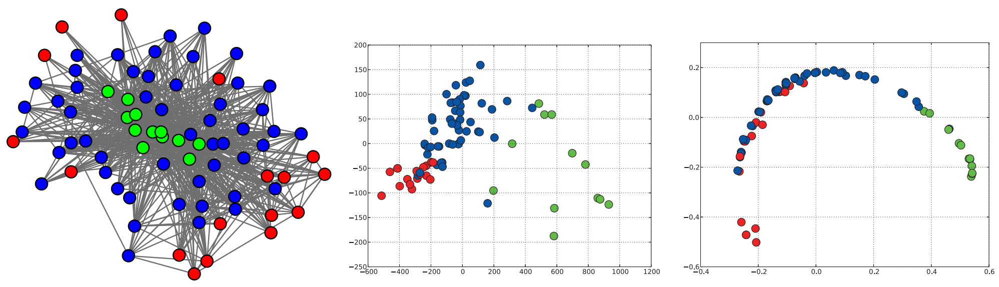
In network data mining, homophily is commonly exploited in node classification tasks, since nodes tend to exhibit the same class than their neighbors [43]. Even though this situation occurs in a large number of networks from very different domains, homophily-based classification techniques fail when the classes of the nodes are defined by the role they play in the network, instead of by the community they belong to.

An illustrative example of this situation is a network containing data on trade of miscellaneous manufactures of metal among 80 countries [1], according to data gathered in 1993 and 1994 from the Commodity Trade Statistics published by the United Nations [44].

[1]The dataset can be downloaded from `http://vlado.fmf.uni-lj.si/pub/networks/data/esna/metalWT.htm`

Each country is represented by a node in the network. Each commercial relationship is represented by an arc, which we consider an undirected edge in practice. In this case, arcs correspond to trading high technology products or heavy manufactures between countries.

In addition, the authors of this dataset annotated countries in the network with their structural world economic position in 1994. World economic positions are a classification of countries in the context of the world-system theory that explains some complex dynamics observed in the real world [45, 46]. This classification splits countries into three possible categories: core countries (colored in green), semi-periphery countries (colored in blue), and periphery countries (colored in red). In short, core countries have a high economical, military, and political power, which allows them to control the world economic system. The periphery is composed of less developed countries, owning a disproportionately small share of global wealth. Finally, the semi-periphery is conformed by countries that do not clearly fall in the previous two categories and exhibit a more intermediate status.



(a) Kamada-Kawai layout. (b) Automorphic embedding. (c) RoleSim embedding.

Figure 4: World trade network and its corresponding node embeddings.

Figure 4 shows the trade network drawn using the Kamada-Kaway layout algorithm and the embeddings obtained by applying our automorphic distance and the RoleSim-based distance. It can be seen that both metrics achieve a good separation of core and semi-periphery countries. However, our distance metric is clearly superior in the separation of semi-periphery and periphery countries. In addition, the embedding generated using distances computed with RoleSim exhibits an artificial curved-line shape, which indicates that only a dimension would be required to represent the information captured by RoleSim. In contrast, the embedding generated using our proposed automorphic distance exploits the two dimensions to represent role-related node properties and achieves a better separation of the different classes.

5 Conclusions

In this paper, we have proposed a novel distance metric for nodes that relaxes the strict concept of automorphic equivalence. To the best of our knowledge, this is the first work to propose a consistent non-normalized distance metric that captures the concept of automorphic equivalence without approximating it using feature engineering. In addition, we have shown that the proposed distance function is a valid distance metric by proving the required conditions. Finally, we have shown how our metric can be exploited to generate node embeddings that capture role information in contrast to previously proposed embedding techniques, which capture locality-based information. We have also shown how our metric is superior to RoleSim in the generation of automorphic node embeddings, leading to a better separation of nodes according to their roles.

Our proposal creates new opportunities in problems related to role discovery. Future work includes exploiting our distance metric in problems related to anomaly detection in networks and transfer learning based on roles shared by nodes across different networks.

Acknowledgments

This work is partially supported by the Spanish Ministry of Economy and the European Regional Development Fund (FEDER), under grant TIN2012-36951 and the program "Ayudas para contratos predoctorales para la formación de doctores 2013" (grant BES-2013-064699).

References

[1] Ryan A Rossi and Nesreen K Ahmed. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1112–1131, 2015.

[2] Joseph J Luczkovich, Stephen P Borgatti, Jeffrey C Johnson, and Martin G Everett. Defining and measuring trophic role similarity in food webs using regular equivalence. *Journal of Theoretical Biology*, 220(3):303–321, 2003.

[3] Emilie M Hafner-Burton, Miles Kahler, and Alexander H Montgomery. Network analysis for international relations. *International Organization*, 63(3):559–592, 2009.

[4] Petter Holme and Mikael Huss. Role-similarity based functional prediction in networked systems: Application to the yeast proteome. *Journal of the Royal Society Interface*, 2(4):327–333, 2005.

[5] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[6] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[7] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3):515–554, 2012.

[8] Ryan A Rossi, Brian Gallagher, Jennifer Neville, and Keith Henderson. Modeling dynamic behavior in large evolving graphs. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pages 667–676. ACM, 2013.

[9] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: Structural role extraction & mining in large graphs. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 1231–1239, 2012.

[10] Eric P Xing, Wenjie Fu, Le Song, et al. A state-space mixed membership blockmodel for dynamic network tomography. *The Annals of Applied Statistics*, 4(2):535–566, 2010.

[11] Harrison C White, Scott A Boorman, and Ronald L Breiger. Social structure from multiple networks. I. Blockmodels of roles and positions. *American Journal of Sociology*, 81(4):730–780, 1976.

[12] Ronald S Burt. Detecting role equivalence. *Social Networks*, 12(1):83–97, 1990.

[13] Francois Lorrain and Harrison C White. Structural equivalence of individuals in social networks. *The Journal of Mathematical Sociology*, 1(1):49–80, 1971.

[14] Lee Douglas Sailer. Structural equivalence: Meaning and definition, computation and application. *Social Networks*, 1(1):73–90, 1978.

[15] Stephen P Borgatti and Martin G Everett. Notions of position in social network analysis. *Sociological Methodology*, pages 1–35, 1992.

[16] Philippa E Pattison. Network models: Some comments on papers in this special issue. *Social Networks*, 10(4):383–411, 1988.

[17] Noah E Friedkin and Eugene C Johnsen. Social positions in influence networks. *Social Networks*, 19(3):209–222, 1997.

[18] Martin G Everett and Stephen P Borgatti. Regular equivalence: General theory. *Journal of Mathematical Sociology*, 19(1):29–52, 1994.

[19] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *arXiv preprint arXiv:1705.02801*, 2017.

[20] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.

[21] Vincent W Zheng, Sandro Cavallari, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. From node embedding to community embedding. *arXiv preprint arXiv:1610.09950*, 2016.

[22] Glen Jeh and Jennifer Widom. SimRank: A measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[23] Masoud Reyhani Hamedani and Sang-Wook Kim. SimRank and its variants in academic literature data: Measures and evaluation. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 1102–1107. ACM, 2016.

[24] Zhenjiang Lin, Michael R Lyu, and Irwin King. PageSim: A novel link-based measure of web page aimilarity. In *Proceedings of the 15th International Conference on the World Wide Web*, pages 1019–1020. ACM, 2006.

[25] Elizabeth A Leicht, Petter Holme, and Mark EJ Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[26] Ruoming Jin, Victor E Lee, and Longjie Li. Scalable and axiomatic ranking of network role similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):3, 2014.

[27] Longjie Li, Lvjian Qian, Victor E. Lee, Mingwei Leng, Mei Chen, and Xiaoyun Chen. *Fast and accurate computation of role similarity via vertex centrality*, pages 123–134. Springer International Publishing, 2015.

[28] Ruoming Jin, Victor E Lee, and Hui Hong. Axiomatic ranking of network role similarity. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 922–930. ACM, 2011.

[29] Boris Weisfeiler and AA Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

[30] Martin Fürer. Weisfeiler-Lehman refinement requires at least a linear number of iterations. In *International Colloquium on Automata, Languages, and Programming*, pages 322–333. Springer, 2001.

[31] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

[32] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

[33] Jeffrey Travers and Stanley Milgram. The small–world problem. *Phychology Today*, 1:61–67, 1967.

[34] Duncan J Watts and Steven H Strogatz. Collective dynamics of small–world networks. *Nature*, 393(6684):440–442, 1998.

[35] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[36] Yoav Goldberg and Omer Levy. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[37] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710. ACM, 2014.

[38] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[39] Florian Wickelmaier. An introduction to MDS. *Sound Quality Research Unit, Aalborg University, Denmark*, 46, 2003.

[40] Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[41] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[42] Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[43] Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. *Node classification in social networks*, pages 115–148. Springer US, Boston, MA, 2011.

[44] Wouter De Nooy, Andrej Mrvar, and Vladimir Batagelj. *Exploratory social network analysis with Pajek*. Cambridge University Press, New York, NY, USA, 2011.

[45] Daniel Chirot and Thomas D Hall. World-system theory. *Annual Review of Sociology*, 8(1):81–106, 1982.

[46] David A Smith and Douglas R White. Structure and dynamics of the global economy: Network analysis of international trade 1965–1980. *Social Forces*, 70(4):857–893, 1992.

3 A network data mining framework

This Section contains the publication related to the software developed in this thesis for network data mining.

The journal paper associated to this part of the dissertation is:

V. Martínez, F. Berzal, J.C. Cubero. The NOESIS network-oriented exploration, simulation, and induction system. ArXiv e-prints 1611.04810, June 2017. *Submitted to Complexity (ISSN: 1099-0526).*

- Status: **Submitted**
- Impact Factor (JCR 2016): 4.621
- Subject Category:
  - Mathematics, Interdisciplinary Applications. Ranking 2/100.
  - Multidisciplinary Sciences. Ranking 9/64.

NOESIS: A Framework for Complex Network Data Analysis

Víctor Martínez Fernando Berzal Juan-Carlos Cubero

Abstract

Network data mining has attracted a lot of attention since a large number of real-world problems have to deal with complex network data. In this paper, we present NOESIS, an open source framework for network-based data mining. NOESIS features a large number of techniques and methods for the analysis of structural network properties, network visualization, community detection, link scoring, and link prediction. The proposed framework has been designed following solid design principles and exploits parallel computing using structured parallel programming. NOESIS also provides a stand–alone graphical user interface allowing the use of advanced software analysis techniques to users without prior programming experience. The framework is available under a BSD open source software license.

1 Introduction

Data mining, an interdisciplinary subfield of Computer Science, studies the process of extracting valuable information from data by discovering patterns or relationships. Data mining includes classification, regression, clustering, or anomaly detection, among others tasks [1]. A large number of tools and techniques are available for tabular data, where all data examples can be represented as tuples in a relation and they share the same set of attributes. However, many problems involve dealing with relational data, where instances are explicitly related through semantic relations. Classic data mining techniques designed for tabular data have severe limitations when we try to fully exploit the relationships available in relational data. The scientific community has focused in the development of data mining techniques for relational data, leading to the availability of a large collection of techniques for the analysis of the structural properties of networks [2], their pleasant visualization [3], the detection of existing communities [4], and scoring existing or potential links to rank existing ones or predict their existence, respectively [5, 6]. Network data mining involves very different problems. Some examples include the prediction of previously unknown protein interactions in protein-protein interaction networks [7], the prediction of collaborations and tendencies in co-authorship networks [8], or ranking the most relevant web sites according to a user query [9].

Different software tools for analyzing relational data have been developed, according to their main goal and the type of user they are directed to. Several tools provide their functionality to end users through closed graphical user interfaces, leading to improved usability but also neglecting the possibility of using the provided techniques in ways unforeseen by their software developers and integrating them in other software projects.

Other tools were designed as software libraries that can be used in different software projects, providing more functionality at the cost of limiting their usage to users with prior programming experience.

Most existing frameworks are focused towards a specific user community or task, as shown in Table 1. For example, Graphviz [10] and Cytoscape [11] are two popular alternatives for network visualization. On the other hand, igraph [12], NetworkX [13], and SNAP [14] are mainly focused on providing reusable collections of algorithms for network manipulation and analysis. Finally, Pajek [15], NodeXL [16], Gephi [17], and UCINET [18] are some of the most widely used tools for social network analysis (SNA) [1].

Tool name	**User interface**	**Programming library**	**Parallelization support**	**Extensibility**
UCINET	✓			
Pajek	✓			
Cytoscape	✓			✓
NodeXL	✓			
Gephi	✓	✓		
igraph		✓		✓
NetworkX		✓		✓
Graphviz		✓		
SNAP		✓	✓	✓
NOESIS	✓	✓	✓	✓

Table 1: Feature comparison of some popular network analysis software packages.

In this paper, we introduce NOESIS (Network–Oriented Exploration, Simulation, and Induction System), a software framework released under a permissive BSD open source license for analyzing and mining complex networks. NOESIS is built on top of a structured parallel programming suite of design patterns, providing a large number of network mining techniques that are able to exploit the multiple processing cores available in current microprocessors for a more efficient computation. Our framework is fully written in Java, which means it is portable across different hardware and software platforms, provided they include a Java virtual machine. In addition, we provide an API binding for Python, which enables the use of NOESIS from the Python scripting language, often used by data scientists.

Our paper is structured as follows. In Section 2, we describe and discuss the NOESIS architecture and design. In Sections 3 and 4, the network analysis and network mining techniques available in NOESIS are presented. In Section 5, NOESIS performance is compared to other popular network analysis tools. Finally, our project current status and future directions are described in Section 6.

[1]A more comprehensive and up–to–date list of available software tools for network analysis can be found at Wikipedia: `https://en.wikipedia.org/wiki/Social_network_analysis_software`.

2 The design of the NOESIS framework

NOESIS has been designed to be an easily–extensible framework whose architecture provides the basis for the implementation of network data mining techniques. In order to achieve this, NOESIS is designed around abstract interfaces and a set of core classes that provide essential functions, which allows the implementation of different features as independent components with strong cohesion and loose coupling. NOESIS components are designed to be maintainable and reusable, yet highly efficient.

2.1 System architecture

The NOESIS framework architecture and its core subsystems are displayed in Figure 1. These subsystems are described below.

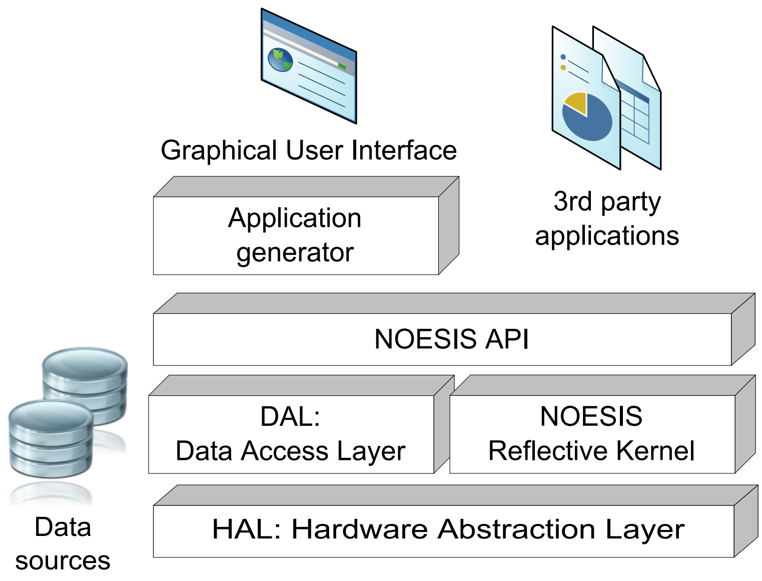


Figure 1: The NOESIS framework architecture and its core subsystems.

The lowest-level component is the hardware abstraction layer (HAL), which provides support for the execution of algorithms in a parallel environment and hides implementation details and much of the underlying technical complexity. This component provides different building blocks for implementing well-studied parallel programming design patterns, such as MapReduce [19]. For example, we would just write *result = (double) Parallel.reduce(index ->x[index] * y[index], ADD, 0, SIZE-1)* to compute the dot product of two vectors in parallel. The HAL does not only implement structured parallel programming design patterns, but it is also responsible for task scheduling and parallel execution. It allows the adjustment of parallel execution parameters, including the task scheduling algorithm.

The reflective kernel is at the core of NOESIS and provides its main features. The reflective kernel provides the base models (data structures) and tasks (algorithms) needed to perform network data mining, as well as the corresponding meta-objects and meta-models, which can be manipulated at run time. It is the underlying layer that supports a large collection of network analysis algorithms and data mining techniques, which are described in the following section. Different types of networks are dealt with using an unified interface, allowing us to choose the particular implementation that is the most adequate for the spatial and computational requirements of each application. Algorithms provided by this subsystem are built on top of the HAL building blocks, allowing the parallelized execution of algorithms whenever possible.

The data access layer (DAL) provides an unified interface to access external data sources. This subsystem allows reading and writing networks in different file formats, providing implementations for some of the most important standardized network file formats. This module also enables the development of data access components for other kinds of data sources, such as network streaming.

Finally, an application generator is used to build a complete graphical user interface following a model driven software development (MDSD) approach. This component provides a friendly user interface that allows users without programming skills to use most of the NOESIS framework features.

2.2 Core classes

The core classes and interfaces shown in Figure 2 provide the foundation for the implementation of different types of networks with specific spatial and computational requirements. Basic network operations include adding and removing nodes, adding and removing links, or querying a node neighborhood. More complex operations are provided through specialized components.

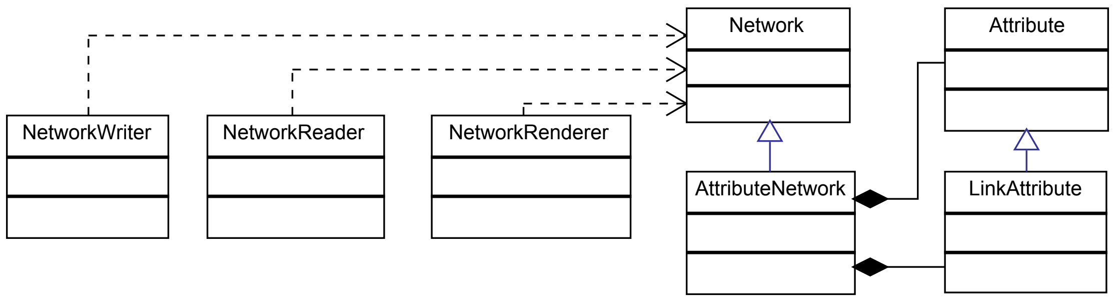


Figure 2: UML class diagram depicting some of the NOESIS core classes and interfaces.

NOESIS supports networks with attributes both in their nodes and their links. These attributes are defined according to predefined data models, including categorical and numerical values, among others.

2.3 Supported data formats

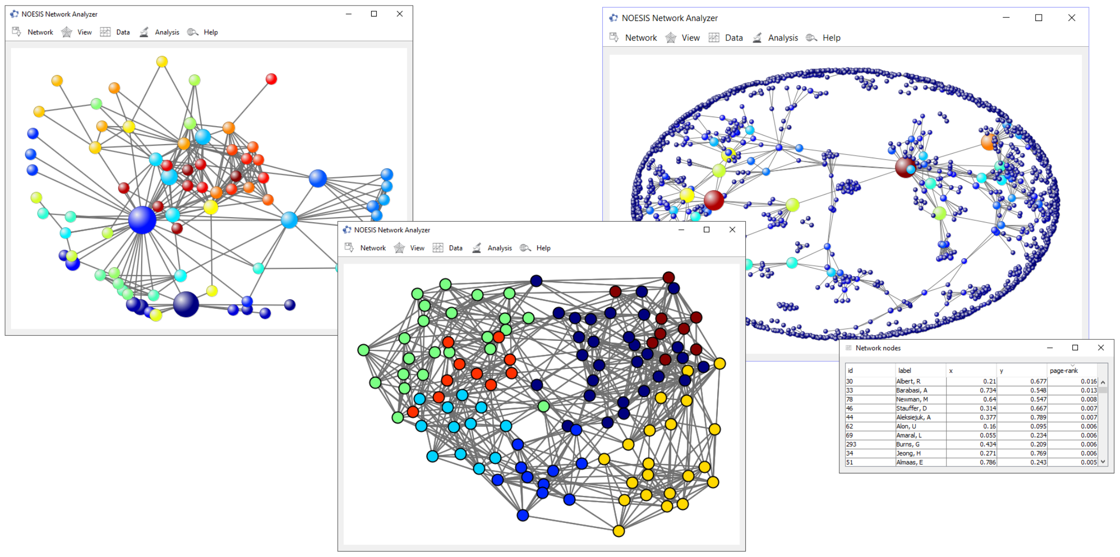
Different file formats have been proposed for network datasets. Some data formats are more space efficient, whereas others are more easily parseable.

NOESIS supports reading and writing network data sets using the most common data formats. For example, the GDF file format is a CSV-like format used by some software tools such as GUESS and Gephi. It supports attributes in both nodes and links. Another supported file format is GML, which stands for Graph Modeling Language. GML is a hierarchical ASCII-based file format. GraphML is another hierarchical file format based on XML, the ubiquitous eXtensible Markup Language developed by the W3C.

Other file formats are supported by NOESIS, such as the Pajek file format, which is similar to GDF, or the file format of the datasets from the Stanford Network Analysis Platform (SNAP) [20].

2.4 Graphical user interface

In order to allow users without programming knowledge to use most of the NOESIS features, a lightweight easy–to–use graphical user interface is included with the standard NOESIS framework distribution. The NOESIS GUI allows non–technical end users loading, visualizing, and analyzing their own network data sets by applying all the techniques provided with NOESIS.



id	label	x	y	page-rank
30	Albert, R	0.21	0.677	0.016
33	Barabasi, A	0.734	0.548	0.013
78	Newman, M	0.64	0.547	0.008
46	Stauffer, D	0.314	0.667	0.007
44	Aleksiejuk, A	0.377	0.789	0.007
62	Alon, U	0.16	0.095	0.006
69	Amaral, L	0.055	0.234	0.006
293	Burns, G	0.434	0.209	0.006
34	Jeong, H	0.271	0.769	0.006
51	Almaas, E	0.786	0.243	0.005

Figure 3: Different screenshots of the NOESIS graphical user interface.

Some screenshots of this GUI are shown in Figure 3. A canvas is used to display the network in every moment. The network can be manipulated by clicking or dragging nodes. At the top of the window, a menu gives access to different options and data mining algorithms. The *Network* menu allows loading a network from an external source and exporting the results using different file formats, as well as creating images of the current network visualization both as raster and vector graphics image. The *View* menu allows the customization of the network appearance by setting specific layout algorithms and custom visualization styles. In addition, this menu allows binding the visual properties of nodes and links to their attributes. The *Data* menu allows the exploration of attributes for each node and link. Finally, the *Analysis* menu gives access to most of the techniques that will be described in the following sections.

3 Network analysis tools

NOESIS is designed to ease the implementation of network analysis tools. It also includes reusable implementations of a large collection of popular network–related techniques, from graph visualization [3] and common graph algorithms, to network structural properties [21] and network formation models [22]. The network analysis tools included in NOESIS and the modules that implement them are introduced in this section.

3.1 Network models

NOESIS implements a number of popular random network generation models, which are described by probability distributions or random processes. Such models have been found to be useful in the study and understanding of certain properties or behaviors observed in real-world networks. Some examples of these models are shown in Figure 4.

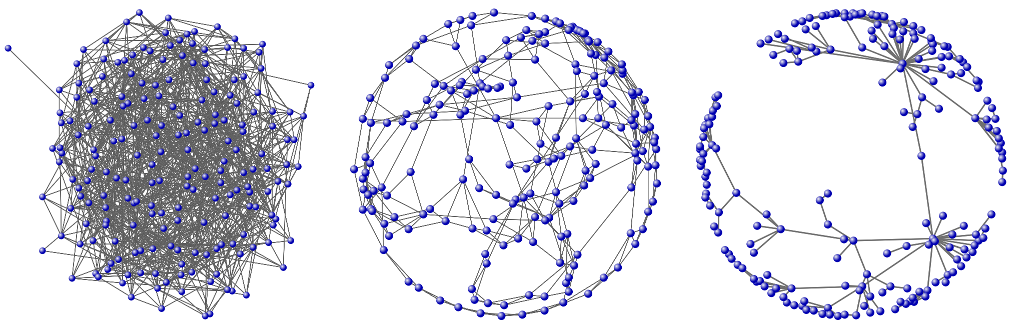


Figure 4: Random networks generated using the Erdös-Rényi model (left), the Watts–Strogatz model (center), and the Barabási–Albert model (right).

Among the models included in NOESIS, the Erdös-Rényi model [23] is one of the simplest ones. The Gilbert model [24] is similar but a probability of existence is given for links instead. The anchored network model is also similar to the two previous models, with the advantage of reducing the occurrence of isolated nodes, but at the cost of being less than perfectly random. Finally, the connected random model is a variation of the anchored model that avoids isolated nodes.

Other models included in NOESIS exhibit specific properties often found in real-world networks. For example, the Watts–Strogatz model [25] generates networks with small-world properties, that is, low diameter and high clustering. This model starts by creating a ring lattice with a given number of nodes and a given mean degree, where each node is connected to its nearest neighbors on both sides. In the following steps, each link is rewired to a new target node with a given probability, avoiding self-loops and link duplication.

Despite the small-world properties exhibited by networks generated by the Watts–Strogatz model are closer to real world networks than those generated by models based on the Erdös-Rényi approach, they still lack some important properties observed in real networks. The Barabási–Albert model [26] is another well-known model that generates networks whose node degree distribution follows a power law, which leads to scale-free networks. This model is driven by a preferential attachment process, where new nodes are added and connected to existing nodes with a probability proportional to their current degree. Another model with very similar properties to Barabási–Albert's model is the Price's citation model [27].

In addition to random network models, a number of regular network models are included in NOESIS. These models generate synthetic networks that are useful in the process of testing new algorithms. The networks regular models include complete networks, where all nodes are interconnected; star networks, where all nodes are connected to a single hub node; ring networks, where each node is connected to its closest two neighbors along a ring; tandem networks, like ring model but without closing the loop; mesh network, where nodes are arranged in rows and columns, and connected only to their adjacent nodes; toruses, meshes where nodes in the extremes of the mesh are connected; hypercubes; binary trees; and isolates, a network without links.

3.2 Network structural properties

Network structural properties allow the quantification of features or behaviors present in the network. They can be used, for instance, to measure network robustness or reveal important nodes and links. NOESIS considers three types of structural properties: node properties, node pair properties (for pairs both with and without links among them), and global properties.

NOESIS provides a large number of techniques for analyzing network structural properties. Many structural properties can be computed for nodes. For example, in-degree and out-degree, indicate the number of incoming and outgoing links, respectively. Related to node degree, two techniques to measure node degree assortativity have been included: biased [28] and unbiased [29] node degree assortativity. Node assortativity is a score between -1 and 1 that measures the degree correlation between pairs of connected nodes. The clustering coefficient can also be computed for nodes. The clustering coefficient of a node is the fraction of its neighbors that are also connected among them.

Reachability scores are centrality measures that allow the analysis of how easy it is to reach a node from other nodes. The eccentricity of a node is defined as the maximum distance to any other node [30]. The closeness, however, is the inverse of the sum of the distance from a given node to all others [31]. An adjusted closeness value that normalizes the closeness according to the number of reachable nodes can also be used. Inversely to closeness, average path length is defined as the mean distance of all shortest paths to any other node. Decay is yet another reachability score, computed as the summation of a delta factor powered by the path length to any other node [22]. It is interesting to note that with a delta factor close to 0, the measure becomes the degree of the node, whereas with a delta close to 1, the measure becomes the component size of the component the node is located at. A normalized decay score is also available.

Betweenness, as reachability, is another way to measure node centrality. Betweenness, also known as Freeman's betweenness, is a score computed as the count of shortest paths the node is involved in [32]. Since this score ranges from $2n-1$ to $n^2-(n-1)$ for n the number of nodes in strongly-connected networks, a normalized variant is typically used.

Finally influence algorithms provide a different perspective on node centrality. These techniques measure the power of each node to affect others. The most popular influence algorithm is PageRank [9], since it is used by the Google search engine. PageRank computes a probability distribution based on the likelihood of reaching a node starting from any other node. The algorithm works by iteratively updating node probability based on direct neighbors probabilities, which leads to convergence if the network satisfies certain properties. A similar algorithm is HITS [33], which stands for hyperlink-induced topic search. It follows an iterative approach, as PageRank, but computes two scores per node: the hub, which is a score related to how many nodes a particular node links, and the authority, which is a score related to how many hubs link a particular node. Both scores are connected by an iterative updating process: authority is updated according to the hub scores of nodes connected by incoming links and hub is updated according to authority scores of nodes connected by outgoing links. Eigenvector centrality is another iterative method closely related to PageRank, where nodes are assigned a centrality score based on the summation of the centrality of their neighbors nodes. Katz centrality considers all possible paths, but penalizes long ones using a given damping factor [34]. Finally, diffusion centrality [35] is another influence algorithm based on Katz centrality. The main difference is that, while Katz considers infinite length paths, diffusion centrality considers only paths of a given limited length.

In the following Java example, we show how to load a network from a data file and compute its structural properties using NOESIS, its PageRank scores in particular:

```
FileReader fileReader = new FileReader("karate.gml");
NetworkReader reader = new GMLNetworkReader(fileReader);
Network network = reader.read();
PageRank task = new PageRank(network);
NodeScore score = task.call();
```

We also show how to implement this example using the NOESIS API for Python:

```
ns = Noesis()
network_reader = ns.create_network_reader("GML")
network = network_reader.read("karate.gml")
```

```
pagerank_scorer = ns.create_node_scorer("PageRank")
scores = pagerank_scorer.compute(network)
ns.end()
```

Different structural properties for links can also be computed by NOESIS. For example, link betweenness, which is the count of shortest paths the link is involved in, or link rays, which is the number of possible paths between two nodes that cross a given link. Some of these properties are used by different network data mining algorithms.

3.3 Network visualization techniques

Humans are still better than machines at the recognition of certain patterns when analyzing data in a visual way. Network visualization is a complex task since networks tend to be huge, with thousands nodes and links. NOESIS enables the visualization of networks by providing the functionality needed to render the network and export the resulting visualization using different image file formats.

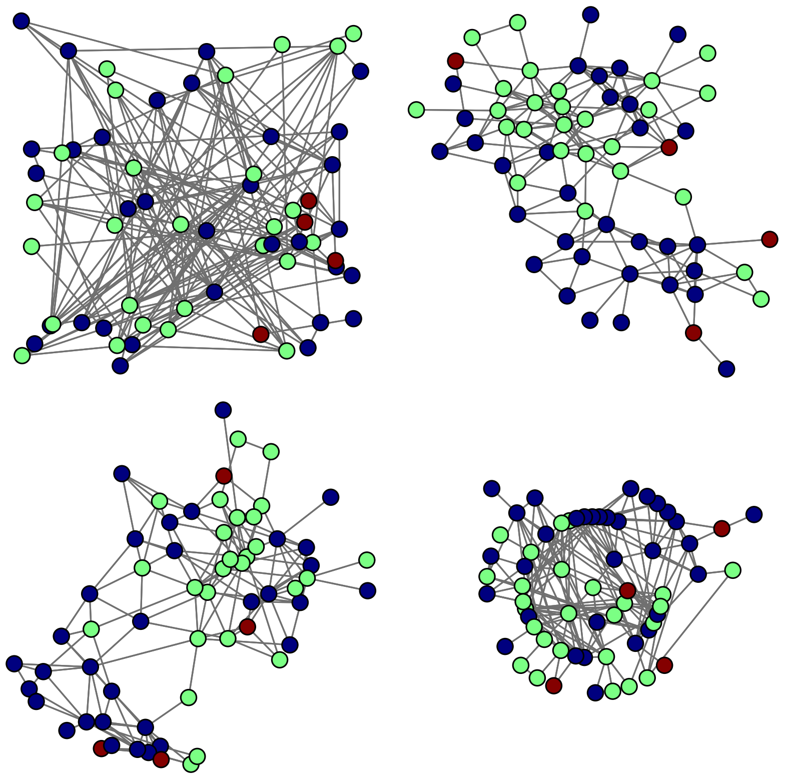


Figure 5: The same dolphin social network [36] represented using different network visualization algorithms: random layout (top left), Kamada–Kawai layout (top right), Fruchterman–Reingold layout (bottom left), and circular layout using average path length (bottom right).

NOESIS provides different automatic graph layout techniques, such as the well–known

Fruchterman–Reingold [37] and Kamada–Kawai [38] force–based layout algorithms (see Figure 5). Force–based layout algorithms assign forces among pairs of nodes and solve the system to reach an equilibrium point, which usually leads to an aesthetic visualization.

Hierarchical layouts [3], which arrange nodes in layers trying to minimize edge crossing, are also included. Different radial layout algorithms are included as well [39]. These layouts are similar to the hierarchical ones, but arrange nodes in concentric circles. Finally, several regular layouts are included. These layouts are common for visualizing regular networks, such as meshes or stars.

NOESIS allows tuning the network visualization look and feel. The visual properties of nodes and links can be customized, including color, size, borders, and so on. In addition, visual properties can be bound to static or dynamic properties of the network. For example, node sizes can be bound to a specific centrality score, allowing the visual display of quantitative information.

4 Network data mining techniques

Network data mining techniques exist for both unsupervised and supervised settings. NOESIS includes a wide array of community detection methods [4] and link prediction techniques [40]. These algorithms are briefly described below.

4.1 Community detection

Community detection can be defined as the task of finding groups of densely connected nodes. A wide range of community detection algorithms have been proposed, exhibiting different pros and cons. NOESIS features different families of community detection techniques and implements more than ten popular community detection algorithms. Some examples of these community detection techniques are shown in Figure 6. The included algorithms, their time complexity, and their bibliographic references are shown in Table 2.

NOESIS provides hierarchical clustering algorithms. Agglomerative hierarchical clustering treats each node as a cluster, and then iteratively merges clusters until all nodes are in the same cluster [53]. Different strategies for the selection of clusters to merge have been implemented, including single-link [42], which selects the two clusters with the smallest minimum pairwise distance; complete-link [43], which selects the two clusters with the smallest maximum pairwise distance; and average-link [44], which selects the two clusters with the smallest average pairwise distance.

Modularity-based techniques are also available in our framework. Modularity is a score that measures the strength of particular division into modules of a given network. Modularity–based techniques search for communities by attempting to maximize their modularity score [54]. Different greedy strategies, including fast greedy [45] and multi-step greedy [46], are available. These greedy algorithms merge pairs of clusters that maximize the resulting modularity, until all possible merges would reduce the network modularity.

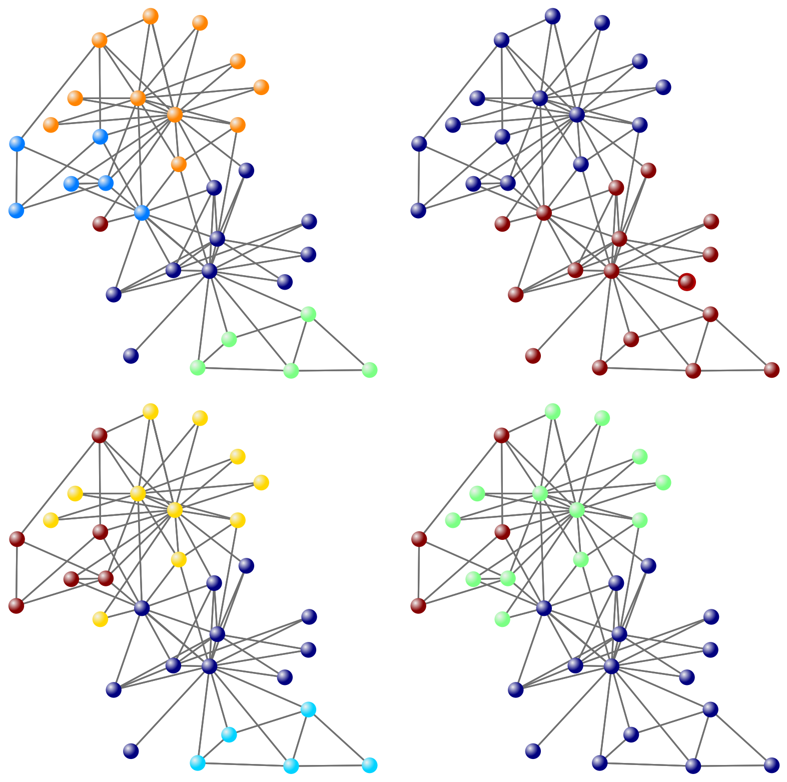


Figure 6: Different community detection methods applied to Zachary's karate club network [41]: Fast greedy partitioning (top left), Kernighan-Lin bi-partitioning (top right), average-link hierarchical partitioning (bottom left), and complete-link hierarchical partitioning (bottom right).

Partitional clustering is another common approach. Partitioning clustering decomposes the network and performs an iterative relocation of nodes between clusters. For example, Kernighan-Lin bi-partitioning [47] starts with an arbitrary partition in two clusters. Then, iteratively exchanges nodes between both clusters to minimize the number of links between them. This approach can be applied multiple times to subdivide the obtained clusters. K-means community detection [48] is an application of the traditional k-means clustering algorithm to networks and another prominent example of partitioning community detection.

Spectral community detection [53] is another family of community detection techniques included in NOESIS. These techniques use the Laplacian representation of the network, which is a network representation computed by subtracting the adjacency matrix of the network to a diagonal matrix where each diagonal element is equal to the degree of the corresponding node. Then, the eigenvectors of the Laplacian representation of the network are computed. NOESIS includes the ratio cut algorithm (EIG1) [49], the Jordan and Weiss NG algorithm (KNSC1) [50], and spectral k-means [51].

Finally, the BigClam overlapping community detector is also available in NOESIS [52]. In this algorithm, each node has a profile, which consists in a score between 0 and 1 for each

Type	**Name**	**Complexity**	**Reference**
Hierarchical	Single-link (SLINK)	$O(v^2)$	[42]
	Complete-link (CLINK)	$O(v^2 \log v)$	[43]
	Average-link (UMPGA)	$O(v^2 \log v)$	[44]
Modularity	Fast greedy	$O(kvd \log v)$	[45]
	Multi-step greedy	$O(kvd \log v)$	[46]
Partitional	Kernighan-Lin bi-partitioning	$O(v^2 \log v)$	[47]
	K-means	$O(kvd)$	[48]
Spectral	Ratio cut algorithm (EIG1)	$O(v^3)$	[49]
	Jordan-Weiss NG (KNSC1)	$O(v^3)$	[50]
	Spectral k-means	$O(v^3)$	[51]
Overlapping	BigClam	$O(v^2)$	[52]

Table 2: Computational time complexity and bibliographic references for the community detection techniques provided by NOESIS. In the time complexity analysis, v is the number of nodes in the network, d is the maximum node degree, and k is the desired number of clusters.

cluster that is proportional to the likelihood of the node belonging to that cluster. Also, a score between pairs of nodes is defined yielding values proportional to their clustering assignment overlap. The algorithm iteratively optimizes each node profile to maximize the value between connected nodes and minimize the value among unconnected nodes.

In the following example, we show how to load a network from a data file and detect communities with the KNSC1 algorithm using NOESIS:

```
FileReader fileReader = new FileReader("mynetwork.net");
NetworkReader reader = new PajekNetworkReader(fileReader);
Network network = reader.read();
CommunityDetector task = new NJWCommunityDetector(network);
Matrix results = task.call();
```

The same example can also be coded in Python using the NOESIS API for Python:

```
ns = Noesis()
network_reader = ns.create_network_reader("Pajek")
network = network_reader.read("mynetwork.net")
community_detector = ns.create_community_detector("NJW")
communities = community_detector.compute(network)
ns.end()
```

4.2 Link scoring and prediction

Link scoring and link prediction are two closely related tasks. On the one hand, link scoring aims to compute a value or weight for a link according to a specific criteria. Most link scoring techniques obtain this value by considering the overlap or relationship between the neighborhood of the nodes at both ends of the link. On the other hand, link prediction computes a value, weight, or probability proportional to the likelihood of the existence of a certain link according to a given model of link formation.

The NOESIS framework provides a large collection of methods for link scoring and link prediction, from local methods, which only consider the direct neighborhood of nodes, to global methods, which consider the whole network topology. Some examples are shown in Figure 7. As the amount of information considered is increased, the computational and spatial complexity of the techniques also increases. The link scoring and prediction methods available in NOESIS are shown in Table 3.

Type	Name	Complexity	Reference
Local	Common Neighbors count	$O(vd^3)$	[55]
	Adamic–Adar score	$O(vd^3)$	[56]
	Resource–allocation index	$O(vd^3)$	[57]
	Adaptive degree penalization score	$O(vd^3)$	[58]
	Jaccard score	$O(vd^3)$	[59]
	Leicht-Holme-Newman score	$O(vd^3)$	[60]
	Salton score	$O(vd^3)$	[61]
	Sorensen score	$O(vd^3)$	[62]
	Hub promoted index	$O(vd^3)$	[63]
	hub depressed index	$O(vd^3)$	[63]
	Preferential attachment score	$O(vd^2)$	[64]
Global	Katz index	$O(v^3)$	[34]
	Leicht-Holme-Newman score	$O(cv^2d)$	[60]
	Random walk	$O(cv^2d)$	[65]
	Random walk with restart	$O(cv^2d)$	[66]
	Flow propagation	$O(cv^2d)$	[67]
	Pseudoinverse Laplacian score	$O(v^3)$	[68]
	Average commute time score	$O(v^3)$	[68]
	Random forest kernel index	$O(v^3)$	[69]

Table 3: Computational time complexity and bibliographic references for the link scoring and prediction methods provided by NOESIS. In the time complexity analysis, v is the number of nodes in the network, d is the maximum node degree, and c refers to the number of iterations required by iterative global link prediction methods.

Among local methods, the most basic technique is the common neighbors score [55], which is equal to the number of shared neighbors between a pair of nodes. Most techniques are variations of the common neighbors score. For example, the Adamic–Adar score [56] is the sum of one divided by the logarithm of the degree of each shared node. The resource–allocation index [57] follows the same expression, but directly considers the degree instead of the logarithm of the degree. The adaptive degree penalization score [58] also follows the same approach, yet automatically determines an adequate degree penalization by considering properties of the network topology. Other local measures consider the number of shared neighbors, but normalize their value according to certain criteria. For example, the Jaccard score [59] normalizes the number of shared neighbors by the total number of neighbors. The local Leicht-Holme-Newman score [60] normalizes the count of shared neighbors by the product of both neighborhoods sizes. Similarly, the Salton score [61] also normalizes, this time using the square root of the product of both node degrees. The Sorensen score [62] considers the double of the count of shared neighbors normalized by the sum of both neighbors size. The hub promoted and hub depressed scores [63] normalize the count of shared neighbors by the minimum and the maximum of both nodes degree respectively. Finally, the preferential attachment score [64] only considers the product of both node degrees.

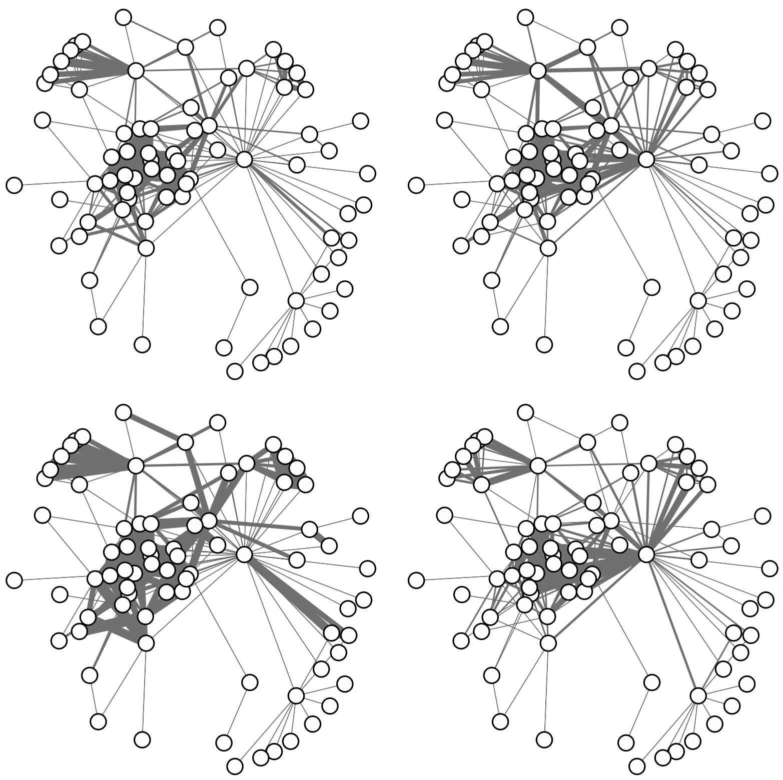


Figure 7: Different link scoring techniques applied to Les Miserables coappearance network [70]: common neighbors (top left), preferential attachment score (top right), Sorensen score (bottom left), and Katz index (bottom right). Link width in the figure is proportional to the link score.

Global link scoring and prediction methods are more complex than local methods. For example, the Katz score [34] sums the influence of all possible paths between two nodes, incrementally penalizing paths by their length according to a given damping factor. The global Leicht-Holme-Newman score [60] is quite similar to the Katz score, but resorts to the dominant eigenvalue to compute the final result.

Random walk techniques simulate a Markov chain of randomly-selected nodes [65]. The idea is that, starting from a seed node and randomly moving through links, we can obtain a probability vector where each element corresponds to the probability of reaching each node. The classical random walk iteratively multiplies the probability vector by the transition matrix, which is the row-normalized version of the adjacency matrix, until convergence. An interesting variant is the random walk with restart [66], which models the possibility of returning to the seed node with a given probability. Flow propagation is another variant of random walk [67], where the transition matrix is computed by performing both row and column normalization of the adjacency matrix.

Some spectral techniques are also available in NOESIS. Spectral techniques, as we mentioned when discussing community detection methods, are based on the Laplacian matrix. The pseudoinverse Laplacian score [68] is the inner product of the rows of the corresponding pair of nodes from the Laplacian matrix. Other spectral technique is the average commute time [68], which is defined as the average number of steps that a random walker starting from a particular node takes to reach another node for the first time and go back to the initial node. Despite it models a random walk process, it is considered to be a spectral technique because it is usually computed in terms of the Laplacian matrix. Given the Laplacian matrix, it can be computed as the diagonal element of the starting node plus the diagonal element of the ending node, minus two times the element located in the row of the first node and the column of the second one.

Finally, the random forest kernel score [69] is a global technique based on the concept of spanning tree, i.e. a connected undirected sub-network with no cycles that includes all the nodes and some or all the links of the network. The matrix-tree theorem states that the number of spanning trees in the network is equal to any cofactor, which is a determinant obtained by removing the row and column of the given node, of an entry of its Laplacian representation. As a result of this, the inverse of the sum of the identity matrix and the Laplacian matrix gives us a matrix that can be interpreted as a measure of accessibility between pairs of nodes.

Using network data mining algorithms in NOESIS is simple. In the following code snippet, we show how to generate a Barabsi-Albert preferential attachment network with 100 nodes and 10 links per node, and then compute the Resource Allocation score for each pair of nodes using NOESIS:

```
Network network = new BarabasiAlbertNetwork(100, 10);
LinkPredictionScore method = new ResourceAllocationScore(network);
Matrix result = method.call();
```

In Python, the previous example would be implemented as follows:

```
ns = Noesis()
network = ns.create_network_from_model("BarabasiAlbert", 100, 10)
predictor = ns.create_link_predictor("ResourceAllocation")
result = predictor.compute(network)
ns.end()
```

5 Performance comparison

NOESIS is designed to manage large complex networks comprised of thousands or even millions of nodes. It provides structures to efficiently represent and deal with relational data. While these features are often offered by most other graph-related frameworks, they lack of parallel computing features. In this Section, we perform an empirical comparison of different popular graph-related frameworks for some common data analysis tasks. The igraph, SNAP, and NetworkX frameworks have been chosen for their comparison with NOESIS. For NOESIS, different CPU scheduling techniques were tested: a work-stealing scheduler (WSS), a future scheduler (FS), and a conventional thread pool scheduler (TPS). Our experiments were performed in a Intel Core i7-3630QM (8 cores at 2.40GHz) with 16GB of RAM under Windows 10 (64 bits).

Three datasets were used in our experiments: a network extracted from Wikipedia with 27475 nodes and 85729 links [2], a peer-to-peer Gnutella file sharing network from August 2002 with 6301 nodes and 20777 edges [71], and a Facebook ego network with 4039 nodes and 88234 edges [72].

The experimentation carried out in this Section consists of computing different computationally-expensive network metrics that are commonly used in network mining: betweenness centrality (BC), link betweenness (LB), closeness (CN), and all shortest paths from every node using Dijkstra's algorithm (APSP). The APSP task could not be implemented using SNAP due to support limitations. We performed 5 runs for each measure and reported the average execution time required to complete the task for each software tool and network data set.

The average execution time for each task/dataset/tool triplet are shown in Table 4. It should be noted that NOESIS is consistently faster that well-known existing frameworks. The superior performance of the NOESIS framework can be attributed to different factors. First of all, NOESIS is written in pure Java, which can be orders of magnitude faster than the Python implementation of igraph and NetworkX. The NOESIS Java code is highly-optimized taking into account the peculiarities of the Java virtual machine (e.g. trying to minimize of memory fragmentation and using a properly-tuned cost model for different operations). As a consequence, the NOESIS Java implementation is surprisingly faster than the C++ implementation of SNAP, which should be expected to be more efficient since highly-optimized C++ applications are usually more efficient than their Java counterparts. Finally, NOESIS parallel programming design patterns let algorithm designers easily exploit the parallelism in multicore processors and multiprocessor systems. The combination of

[2]The Wikipedia dataset can be downloaded from: `http://spark-public.s3.amazonaws.com/sna/other/wikipedia.gml`.

Dataset	Framework	BC	LB	CN	APSP
Wikipedia	NOESIS-WSS	14485	32748	23581	54374
	NOESIS-FS	13837	33082	23657	57299
	NOESIS-TPS	**10854**	**23424**	**23233**	**48005**
	igraph	43328	110681	73181	198568
	SNAP	464055	579435	184254	-
	NetworkX	2246487	2812940	409698	1598431
p2p-Gnutella	NOESIS-WSS	576	1025	879	1971
	NOESIS-FS	499	1097	868	1804
	NOESIS-TPS	**331**	**717**	**865**	**1288**
	igraph	1576	2875	4046	8947
	SNAP	18506	21854	8949	-
	NetworkX	67631	81315	16719	60983
Facebook	NOESIS-WSS	2267	6778	5087	1778
	NOESIS-FS	2500	7937	4977	2140
	NOESIS-TPS	**1841**	**5339**	**4874**	**1443**
	igraph	4677	26962	5302	15640
	SNAP	15206	17604	15295	-
	NetworkX	244842	291605	120558	481848

Table 4: The performance of different network analysis tools in different tasks over several network data sets (time shown in milliseconds). Best times for each dataset are shown in bold.

these features allows NOESIS to complete tasks associated to computing important network-related structural properties faster than some popular software packages, even orders of magnitude faster.

In our parallelization experiments, we implemented different CPU scheduling techniques. NOESIS is faster than existing tools no matter which scheduler is employed. The NOESIS conventional thread pool scheduler, which implements a common solution to the parallelization of software with the help of a single thread pool, consistently obtains the best results in the batch of experiments we have performed. An alternative implementation, using Java futures is somewhat slower. Similar results were obtained using a work-stealing CPU scheduler [73]. In a work stealing scheduler, each processor in a computer system has a queue of work items (computational tasks, threads) to perform. When a processor runs out of work, it looks at the queues of other processors and "steals" their work items. In effect, work stealing distributes the scheduling work over idle processors, and as long as all processors have work to do, no scheduling overhead occurs. It is employed in the scheduler for the Cilk programming language [74], the Java fork/join framework [75], and the .NET Task Parallel Library [76]. For the tests we performed, which involved a heavy work load for a single multicore processor, the simplest strategy seemed to work better, yet typical workloads might be different and alternative CPU schedulers might be preferable.

6 Conclusion

In this paper, we have presented the NOESIS framework for complex network data mining. NOESIS provides a large collection of data analysis techniques for networks, in a much more efficient way than competing alternatives, since NOESIS exploits the parallelism available in existing hardware using structured parallel programming design patterns [77]. As shown in our experimentation, NOESIS is significantly faster than other libraries, even orders of magnitude faster, which is extremely important when dealing with massive networks.

Currently, the NOESIS project has achieved a mature status with more than thirty five thousand lines of Java code, hundreds of classes, and dozens of packages. In addition, production code is covered by automated unit tests to ensure the correctness of the implemented algorithms. NOESIS includes a custom library of reusable components with more than forty thousand lines, providing different general functionalities, such as customizable collections, structured parallel programming building blocks, mathematical routines, and a model-driven application generator on top of which the NOESIS graphical user interface is built. Since Python has become a very popular programming language for scientific computing, a complete API binding for Python is provided, so that NOESIS can be used from Python.

NOESIS is one of the most complete and efficient "out of the box" frameworks for analyzing and mining relational data. Our framework can accelerate the development of software that needs to analyze networks by providing efficient and scalable techniques that cover different aspects of network data mining. Our framework is open source and freely available from its official web site, `http://noesis.ikor.org`, under a permissive Berkeley Software Distribution license.

Data availability

The network datasets used in our performance evaluation are from publicly available studies, which have been cited in our paper. Network datasets can be obtained from the Stanford Large Network Dataset Collection, `http://snap.stanford.edu/data`.

Conflicts of interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The NOESIS project is partially supported by the Spanish Ministry of Economy and the European Regional Development Fund (FEDER), under grant TIN2012–36951, and the Spanish Ministry of Education under the program "Ayudas para contratos predoctorales para la formación de doctores 2013" (grant BES–2013–064699). We are grateful to Aarón Rosas, Francisco–Javier Gijón, and Julio–Omar Palacio for their contributions to the implementation of community detection methods in NOESIS.

References

[1] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, et al. *Introduction to data mining.* Pearson, Addison Wesley, Boston, 2006.

[2] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4):175–308, 2006.

[3] Roberto Tamassia. *Handbook of Graph Drawing and Visualization.* CRC Press, 2013.

[4] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[5] Lise Getoor and Christopher P Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[6] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[7] Víctor Martínez, Carlos Cano, and Armando Blanco. Prophnet: A generic prioritization method through propagation of information. *BMC Bioinformatics*, 15(Suppl 1):S5, 2014.

[8] Milen Pavlov and Ryutaro Ichise. Finding experts by link prediction in co-authorship networks. *Finding Experts on the Web with Semantics Workshop*, 290:42–55, 2007.

[9] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[10] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. Graphviz - open source graph drawing tools. In *Graph Drawing*, pages 483–484. Springer, 2002.

[11] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[12] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006.

[13] Daniel A Schult and P Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, volume 2008, pages 11–16, 2008.

[14] Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[15] Vladimir Batagelj and Andrej Mrvar. Pajek-program for large network analysis. *Connections*, 21(2):47–57, 1998.

[16] Marc A Smith, Ben Shneiderman, Natasa Milic-Frayling, Eduarda Mendes Rodrigues, Vladimir Barash, Cody Dunne, Tony Capone, Adam Perer, and Eric Gleave. Analyzing (social media) networks with NodeXL. In *Proceedings of the fourth International Conference on Communities and Technologies*, pages 255–264. ACM, 2009.

[17] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *International AAAI Conference on Weblogs and Social Media*, 8:361–362, 2009.

[18] Stephen P Borgatti, Martin G Everett, and Linton C Freeman. UCINET for Windows: Software for social network analysis. Technical report, Analytic Technologies, 2002.

[19] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[20] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, jun 2014.

[21] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[22] Matthew O. Jackson. *Social and Economic Networks*. Princeton University Press, Princeton, NJ, USA, 2008.

[23] Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[24] Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

[25] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *Nature*, 393(6684):440–442, 1998.

[26] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

[27] Mark EJ Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[28] M Piraveenan, M Prokopenko, and AY Zomaya. Local assortativeness in scale-free networks. *Europhysics Letters*, 84(2):28002, 2008.

[29] Mahendra Piraveenan, Mikhail Prokopenko, and Albert Y Zomaya. Classifying complex networks using unbiased local assortativity. In *ALIFE*, pages 329–336, 2010.

[30] Per Hage and Frank Harary. Eccentricity and centrality in networks. *Social Networks*, 17(1):57–63, 1995.

[31] Alex Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950.

[32] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[33] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[34] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[35] Chanhyun Kang, Cristian Molinaro, Sarit Kraus, Yuval Shavitt, and VS Subrahmanian. Diffusion centrality in social networks. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 558–564. IEEE Computer Society, 2012.

[36] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[37] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[38] Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[39] Graham J Wills. Nicheworksinteractive visualization of very large graphs. *Journal of computational and Graphical Statistics*, 8(2):190–212, 1999.

[40] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[41] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, pages 452–473, 1977.

[42] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[43] Daniel Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

[44] Bing Liu. *Web Data Mining, 2 ed.* Berlin, Germany: Springer Berlin Heidelberg, 2011.

[45] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[46] Philipp Schuetz and Amedeo Caflisch. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E*, 77(4):046112, 2008.

[47] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[48] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, number 14 in 1, pages 281–297. Oakland, CA, USA., 1967.

[49] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[50] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2:849–856, 2002.

[51] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[52] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 587–596. ACM, 2013.

[53] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[54] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[55] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, 2001.

[56] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[57] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

[58] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. Adaptive degree penalization for link prediction. *Journal of Computational Science*, 13:1–9, 2016.

[59] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[60] Elizabeth A Leicht, Petter Holme, and Mark EJ Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[61] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[62] Thorvald Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biologiske Skrifter*, 5:1–34, 1948.

[63] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, 2002.

[64] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[65] Karl Pearson. The problem of the random walk. *Nature*, 72:342, 1905.

[66] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 613–622. IEEE Computer Society, 2006.

[67] Oron Vanunu and Roded Sharan. A propagation-based algorithm for inferring gene-disease assocations. In *German Conference on Bioinformatics*, pages 54–52, 2008.

[68] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[69] Pavel Chebotarev and Elena Shamis. Matrix-forest theorems. *arXiv preprint math/0602575*, 2006.

[70] Donald E. Knuth. *Stanford GraphBase: A Platform for Combinatorial Computing, The.* Addison-Wesley Professional, 1st edition, 2009.

[71] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *arXiv preprint cs/0209028*, 2002.

[72] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems*, pages 539–547, 2012.

[73] Robert D Blumofe and Charles E Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM (JACM)*, 46(5):720–748, 1999.

[74] Robert D Blumofe, Christopher F Joerg, Bradley C Kuszmaul, Charles E Leiserson, Keith H Randall, and Yuli Zhou. Cilk: An efficient multithreaded runtime system. *Journal of Parallel and Distributed Computing*, 37(1):55–69, 1996.

[75] Doug Lea. A java fork/join framework. In *Proceedings of the ACM 2000 conference on Java Grande*, pages 36–43. ACM, 2000.

[76] Daan Leijen, Wolfram Schulte, and Sebastian Burckhardt. The design of a task parallel library. *ACM SIGPLAN Notices*, 44(10):227–242, 2009.

[77] Michael D McCool, Arch D Robison, and James Reinders. *Structured parallel programming: patterns for efficient computation.* Elsevier, 2012.