Machine Learning (2021) 110:37–87
https://doi.org/10.1007/s10994-020-05898-0

Node classification over bipartite graphs through projection

Marija Stankova[1] · **Stiene Praet**[1] · **David Martens**[1] · **Foster Provost**[2]

Received: 15 July 2014 / Revised: 13 March 2020 / Accepted: 14 July 2020 / Published online: 28 July 2020

Abstract
Many real-world large datasets correspond to bipartite graph data settings—think for example of users rating movies or people visiting locations. Although there has been some prior work on data analysis with such bigraphs, no general network-oriented methodology has been proposed yet to perform node classification. In this paper we propose a three-stage classification framework that effectively deals with the typical very large size of such datasets. The stages are: (1) top node weighting, (2) projection to a weighted unigraph, and (3) application of a relational classifier. This paper has two major contributions. Firstly, this general framework allows us to explore the design space, by applying different choices at the three stages, introducing new alternatives and mixing-and-matching to create new techniques. We present an empirical study of the predictive and run-time performances for different combinations of functions in the three stages over a large collection of bipartite datasets with sizes of up to 20 million × 30 million nodes. Secondly, thinking of classification on bigraph data in terms of the three-stage framework opens up the design space of possible solutions, where existing and novel functions can be mixed and matched, and tailored to the problem at hand. Indeed, in this work a novel, fast, accurate *and* comprehensible method emerges, called the SW-transformation, as one of the best-performing combinations in the empirical study.

Keywords Bipartite graphs · Two-mode networks · Node classification · Behavioral data

Editor: Luc De Raedt.

✉ Stiene Praet
stiene.praet@uantwerpen.be

Marija Stankova
marija.stankova@uantwerpen.be

David Martens
david.martens@uantwerpen.be

Foster Provost
fprovost@stern.nyu.edu

[1] University of Antwerp, Prinsstraat 13, Antwerp, Belgium

[2] New York University, 44 West Fourth Street, 8-86, New York, USA

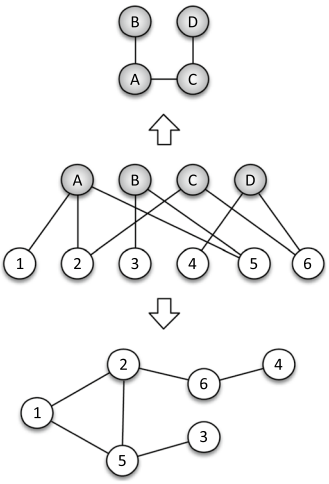
1 Introduction

Many relational, behavioral and transactional datasets can be modeled as bipartite graphs (bigraphs, sometimes also referred to as 2-mode or affiliation networks), which are defined by having (1) two types of nodes and (2) edges that exist only between nodes of different types. Think for example of relationships based on companies sharing board members (Seierstad and Opsahl 2011), users meeting at locations or events (Eagle and Pentland 2006), users rating different products (Ziegler et al. 2005), consumers making payments to merchants (Martens and Provost 2011), mobile devices visiting locations (Provost et al. 2012, 2015), authors collaborating on scientific papers (Newman 2001b), people communicating on online forums (Opsahl 2011), actors playing in the same movies (Guillaume and Latapy 2006), words occurring in the same sentence/search query (Guillaume and Latapy 2006; Cancho and Solé 2001) or even proteins involved in the same metabolic processes (Guillaume and Latapy 2006). These datasets are typically high-dimensional and sparse: although the number of possible top nodes is large (e.g., the possible number of webpages to like or products to rate), due to limited 'behavioral capital' any individual can only take a very small fraction of all the possible actions (Junqué de Fortuny et al. 2013).

The analysis of bigraph data has been mainly limited to measuring descriptive statistics, link prediction for recommender systems, and clustering. In this study, we take a different approach and focus on the task of node classification within bigraphs, where nodes for which the class is known are related to nodes for which the class must be estimated (Macskassy and Provost 2007). As an example of node classification, we consider a bigraph of users and locations, where the users are connected to the locations they have visited (e.g., logged into a WiFi IP address (Provost et al. 2012, 2015) or checked in using a social network app (Cho et al. 2011). An interesting node classification task would be to predict the brand interest of the users, in order to target them with mobile ads. For this example, brand interest is defined as whether a user would demonstrate brand affinity actions, like visiting a brand loyalty club page or a purchase page. Based on the brand interest of other users visiting the same locations we can infer the (likelihood of the) class of the unknown user (Provost et al. 2012, 2015). The rationale behind this idea is the concept of cross-domain similarity (Martens and Provost 2011; Provost et al. 2009, 2012, 2015): users that have similar preferences for some locations, like specific bars or restaurants, are likely interested in the same brands.

For this task, there has been no general network-based methodology proposed yet in prior work. Most of the previous studies that have looked at node classification for this type of data formulate it simply as a standard classification problem, which results in massive, sparse feature data. Some examples include predicting personality traits from datasets of Facebook users liking pages (Kosinski et al. 2013), predicting demographic attributes (Goel et al. 2012; Hu et al. 2007) and brand interest (Raeder et al. 2012) from people's browsing history, predicting political views from history of videos watched on YouTube (Weber et al. 2013), etc. In many of these applications, a linear Support Vector Machine (SVM) is used because of its efficiency for classifying large sparse datasets (Li et al. 2015). In this paper, we examine an alternative, and more concise, network-based formulation. We compare this method on a collection of large bigraph datasets to the frequently used linear SVM and show that it outperforms the standard formulation both in terms of predictive performance and scalability to large datasets.

Generally, two main approaches to analyse bigraphs exist with the aim of obtaining summary metrics and summary graphs. The first one is using techniques and metrics that are



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Fig. 1 Bigraph, top node projection and bottom node projection (left), adjacency matrix representation of the bigraph (right)

specially designed for bipartite graphs (Latapy et al. 2008). This direct approach takes into account the bipartite nature of this particular type of graph, but unfortunately there are only few techniques that can be applied directly on the bigraph. Therefore a second, indirect approach is often used, which is the basis of the methodology that we propose. Let us separate the two subsets of nodes in the bigraph into a set of *top* nodes and a set of *bottom* nodes. Choosing the nodes to focus on (i.e., to classify) as the bottom nodes, a bigraph can be analyzed by transforming it to a homogeneous unigraph of the bottom nodes, called a projection, where nodes are linked if they share a common top node (see Fig. 1, left) (Latapy et al. 2008). This projection approach allows the application of existing network analysis techniques for unigraphs to the bipartite case. It is very convenient for the problem of node classification, as numerous relational classifiers for network data exist for homogeneous graphs.

To the best of our knowledge, this paper presents a first, general study of node classification within bigraphs by transforming the bigraph into a unigraph projection. The main contributions of this work are: (1) we provide a general framework for performing node classification within bipartite data via projection, which allows us to explore the design space and mix-and-match components to create new techniques; and (2) in doing so, we introduce a fast, comprehensible model, called the SW-transformation, that calculates the label scores directly on the bigraph. This method allows easy scaling to big datasets of up to millions of nodes and it is convenient for most of today's big datasets that are very sparse, with nodes being connected to only few other nodes in the projection.

The rest of the paper is structured as follows. Section 2 summarizes the related literature on bigraph data analysis and node classification. Next, Sect. 3 presents a range of functions that can be employed in the different framework stages. Section 4 describes the datasets used and Sect. 5 presents our findings. Limitations and future research are discussed in Sect. 6 and finally, Sect. 7 concludes.

2 Related work

To the best of our knowledge, this paper presents the first systematic study of node classification within bigraphs by transforming the bigraph into a unigraph projection. In the following sections we will first present prior literature on bigraph analysis and unigraph projections. Next, we will briefly discuss existing node classification techniques.

2.1 Bigraph data analysis

The literature regarding bigraphs has so far been focused on measuring descriptive statistics, link prediction for recommender systems, and clustering. There has been some initial research that explores the properties of the bigraphs and that extended several *global network metrics* for unigraphs to the bipartite case. Centrality measures, which determine the varying importance of nodes within the graph, like betweenness, degree, closeness and eigenvector centralities (Borgatti and Halgin 2011; Borgatti and Everett 1997; Faust 1997), as well as the clustering coefficient (Latapy et al. 2008; Lind et al. 2005; Opsahl 2011; Robins and Alexander 2004) have been adapted for bigraphs. The second research area, link prediction, has been applied in prior literature to recommender systems for online music shops (Benchettara et al. 2010), online book stores (Huang et al. 2005), movies recommendation (Zhou et al. 2007), hypotheses generation (Kim 2017) etc. Finally, clustering has been used for discovering community structures in bigraphs of companies and board directors (Barber 2007), women attending events (Barber 2007; Doreian et al. 2004), supreme court voting (Doreian et al. 2004), finding similar users or genres of music (Lambiotte and Ausloos 2005), clustering documents based on the occurring terms (Zha et al. 2001), looking for similar actors based on the movies they have played in Sun et al. (2005), similar authors based on the papers they collaborated (Sun et al. 2005), conferences based on the authors that published, etc.

In addition, there exist many studies that essentially use unigraph projections of bipartite data. For instance, the datasets used to create networks based on scientific collaborations (Liben-Nowell and Kleinberg 2007), co-occurrence of companies in text documents (Macskassy and Provost 2007), web page co-citation (Lu and Getoor 2003), movies linked if they share the same production company or crew (Macskassy and Provost 2003, 2007), book co-purchase (Gallagher et al. 2008) and so on, in the unigraph literature are in fact bigraph projections. Projecting bigraphs into unigraphs results in loss of information (Latapy et al. 2008). Studies exists that explore the problem of how to most accurately represent the bigraph with a transformation to unipartite graph. For example, Zweig and Kaufmann (2011) take the approach of connecting nodes in the projection if they have a much higher number of occurrences of motifs (recurrent and statistically significant subgraphs or patterns) compared to the random graph model of the given bigraph. Furthermore, Zhou et al. (2007) propose a method for projecting bigraphs into asymmetrical unigraphs, where the weight from one node to another in the projection is not necessarily the same as in the opposite direction. They calculate the weights in the projection by first assigning an initial weight to the bottom nodes in the bigraph and then equally distributing them over the neighboring top nodes. In the next phase, the weights are once more distributed, this time from the top to the bottom nodes. This results in a linear equation for each bottom node, where the coefficients signify the link weight in the projection with direction from the specific bottom node. Gupte and Eliassi-Rad (2012) consider a wide range of

measures for weighting unigraph projections. They define a set of axioms which approximate intuition and examine how well the weighting measures in previous literature satisfy this characterization.

This paper looks at bigraphs from a different angle: we compare different projection methods based on their performance on a particular task—node classification. We propose a range of measures for determining the weights of the unigraph and assess how well they represent the relevant underlying structure by comparing the predictive performance of relational classifiers on the unigraph projections. As such, we have an objective function that determines to what extent the predictive information present in a bigraph is also contained in the projected unigraph.

2.2 Node classification techniques

In this paper, we use bigraph data for node classification. Specifically, given a bipartite network with labels on some nodes, how do we classify the other nodes. A traditional approach for node classification in bigraphs is to extract features from the network structure and apply a standard classifier. The bigraph data is represented by the corresponding adjacency matrix, with as many rows as there are bottom nodes and as many columns as there are top nodes. This typically results in a high-dimensional and very sparse matrix as most elements in this matrix will be zero (Junqué de Fortuny et al. 2013). To this dataset, standard classification techniques can be applied. Linear SVMs are often used to classify such large sparse datasets efficiently (Li et al. 2015). Moreover, according to a large benchmark study of de Cnudde et al. (2017), comparing 11 classification techniques on 43 fine-grained behavioral data sets, linear SVM with L2 regularization is one of the best performing techniques in terms of predictive performance. However, scalability issues impede the easy application of this technique on very large datasets. To train an SVM on such a huge dataset can require sampling and dimensionality reduction, which also scale badly to these settings (Martens et al. 2013). In our empirical study, below, we further discuss this scalability requirement and include SVM as a benchmark for comparison. A full comparison among all techniques that could be applied to the adjacency matrix is outside the scope of this paper, which focuses on bigraph projections.

In this paper, we present a different approach for node classification in bigraph data, based on the unigraph projection. Seeing that typical bigraph datasets often are very large transactional datasets, our proposed method is designed to scale up easily to millions and even billions of nodes. First, the bigraph is transformed into a unigraph using a projection approach, as was discussed in the previous section. Next, a relational classifier is applied to the unigraph. In a graph where nodes with known class labels are connected to nodes with unknown class labels, relational classifiers make use of the graph structure to estimate the unknown labels. Unlike traditional, non-relational models, which make use only of the local information about a node, univariate relational classifiers use information about the target variable for the related nodes (their labels or predictions thereof) (Macskassy and Provost 2007). Macskassy and Provost (2007) compared various relational classifiers for univariate, unipartite node classification and found that the network-only Link-Based classifier (nLB) dominates when many labels are known, whereas the weighted-vote Relational Neighbor (wvRN) classifier and class-distribution Relational Neighbor (cdRN) classifier dominate when fewer labels are known. The network-only Bayes Classifier (nBC) was almost always significantly worse than the other three relational classifiers. These relational classifiers will be used in our study, although other techniques might fit into our

framework as well. For example, other methods for Statistical Relational Learning (SRL) go beyond just using the univariate, unipartite network, combining statistical learning to address uncertainty in the data and relational learning to deal with relational structures (Getoor and Taskar 2007). Khosravi and Bina (2010) review four well-known SRL models: Probabilistic Relational Models (PRM), Relational Dependency Network (RDN), Bayesian Logic Programming (BLP) and Markov Logic Networks (MLN). They argue that the biggest limitation shared between SRL methods is their computational complexity, which is proportional to the size of the graph and limits the scalability for many realistic datasets. Thus, these techniques are not suitable for many applications, including our benchmark study that consists of very large datasets of up to millions of nodes.

Recently, network embeddings increasingly have been used to extract node features based on network structure. Network embeddings represent nodes by a vector in a low-dimensional space, while incorporating information about the original structure of the network (Liu et al. 2018). Classifiers built on these embeddings have shown high precision for node classification (Goyal and Ferrara 2018). The most commonly used methods are based on matrix factorization (e.g., singular value decomposition (SVD) and multiple dimensional scaling (MDS)), random walks [e.g., DeepWalk (Perozzi et al. 2014) and node2vec (Grover and Leskovec 2016)], edge modeling [e.g., LINE (Tang et al. 2015) and Graphgan (Wang et al. 2018)] or deep neural networks [e.g., structural deep network embedding (SDNE) (Cui et al. 2018; Wang et al. 2016) and graph convolutional networks (GCN) (Kipf and Welling 2016)] and can be performed unsupervised or semi-supervised (Cui et al. 2018; Zhang et al. 2018). Random walk and edge modeling methods adopt stochastic gradient descent optimization and the time complexity of these algorithms is often linear with respect to the number of vertices or edges. This makes them much more efficient than matrix factorization based methods that are solved by eigenvector decomposition and involve quadratic time complexity in the number of vertices, or higher. However, random walks and edge modeling only capture the local structure of the network. Deep learning based methods can capture non-linearity in networks, but their computational cost is usually high (Zhang et al. 2018).

Most network representation studies have proposed methods for homogeneous network embeddings, where vertices are of the same type. However, more recent work has also focused on bipartite network embeddings for the task of node classification. Gao et al. (2018) developed Bipartite Network Embedding (BiNE), that accounts for both the explicit relations (observed links) and implicit relations (transitive links) in learning the node representations. Since it is based on random walks and can be applied to the bipartite network structure directly, it has the potential to be scalable to large networks. Also methods that are developed for heterogeneous networks; such as Metapath2vec (Dong et al. 2017), IGE (Zhang et al. 2017), GPSP (Du et al. 2018), activeHNE (Chen et al. 2019) and FeatWalk (Huang et al. 2019) are applicable to bipartite networks.

Next to the expensive storage and computation costs for most network embedding techniques, these techniques suffer mainly from another important drawback: the results of network embeddings are difficult to understand. The goal of network embedding is to represent nodes by a vector in a low-dimensional space, so that the embeddings implicitly preserve certain structural and content information of the original networks. Nodes that are similar to each other in the network are mapped closely together in the embedding space. Network embeddings have been shown to generate feature representations that can be effective for particular tasks, but they have not been shown to be comprehensible regarding how the constructed dimensions in the embedding space related to the important properties for a particular task. For example, when these representations are used as input

to a classification algorithm it is difficult to explain why a certain node (represented by its embedding) is classified as positive or negative (Liu et al. 2018). Unfortunately, interpretability is often crucial for real-world applications where a model needs to be validated and trusted before implementation (Martens et al. 2011; Martens and Provost 2014).

The goal of this paper is to develop a framework that allows the systematic exploration of the design space of bigraph projection and classification methods. We propose various options that can be mixed and matched and show how these design choices can be compared easily and on equal footing using our flexible framework. Of course, our list of proposed options contains just a small fraction of all possible techniques that could be examined. We encourage practitioners and researchers to experiment with different design choices for a particular problem and apply our framework for empirical evaluation.

3 Methods

Let us now describe the technical details of the framework, defining bigraphs and projections more formally, defining the framework based on these definitions, presenting the various functions used within the framework, and finally discussing scalability in more detail.

3.1 Bigraphs and projections

A bigraph can formally be defined as the triplet $G = (\top, \bot, E)$, where $\top$ denotes a set of top nodes, $\bot$ is a set of bottom nodes and $E \subseteq \top \times \bot$ is a set of links (edges). In this study, we use several basic metrics for describing the bigraphs that were introduced in the work of Latapy et al. (2008). For each bigraph dataset $G = (\top, \bot, E)$, $n_\top$ denotes the number of top nodes $n_\top = |\top|$, $n_\bot$ the number of bottom nodes $n_\bot = |\bot|$ and m the total number of edges. The average degree of the top and the bottom nodes can be calculated as $k_\top = \frac{m}{n_\top}$ and $k_\bot = \frac{m}{n_\bot}$ respectively, and the total average degree k over the whole bigraph as $k = \frac{2m}{n_\top + n_\bot}$. The density of the graph, which represents the probability that two randomly chosen nodes from the distinct node sets are connected, is equal to $\delta(G) = \frac{m}{n_\top \cdot n_\bot}$.

In order to make use of the existing relational classifiers, we can transform a bigraph into a unigraph using the projection approach. A projection is created by interconnecting the nodes of one of the two sets of the bigraph, if they share at least one neighboring node from the other set of nodes. This means that the projection of the bottom nodes ($\bot$ projection), defined as $G' = (\bot, E')$ with a set of edges $E' \subseteq \bot \times \bot$, can be obtained by connecting the nodes in $\bot$ that share at least one common neighbor in $\top$. The projection of the top nodes can be defined similarly, but for consistency in what follows we will only consider the bottom node projection. Figure 1 (left) depicts a bigraph, along with its $\top$-projection and its $\bot$-projection. The adjacency matrix A of the same bigraph can be seen on Fig. 1 (right), with rows representing the bottom nodes and columns representing the top nodes. An element x_{ij} in the adjacency matrix has value of 1 if the corresponding bottom node i and top node j are connected and otherwise 0. Since every top node with degree d creates a clique in the $\bot$ projection with $d(d-1)/2$ links (analogously for the bottom nodes in the $\top$ projection), the process of projecting the bigraph can result in very dense projections, even in cases where the bigraph itself is not very dense (Latapy et al. 2008). Guillaume and Latapy (2006) have looked at the projections of random bipartite graphs with prescribed degree distributions in order to analyse the properties that are induced by the underlying

structure of the bigraphs. They observed that the projections have a low average distance between the nodes ("small world effect"), with a diameter on the order of $\theta(log|\perp|)$ (or $\theta(log|\top|)$ for the top node projections). Moreover, if the bottom (or top) node set follows a power-law distribution, the projection as well follows a power law distribution with the same exponent. The authors also observed high clustering coefficients in the projections, which suggests that this property can be seen as a consequence of the projecting step, rather than a property of the particular network under study.

Projecting the bigraph gives the advantage of using the powerful methods for unipartite graph analysis, but is also an irreversible process that results in loss of information. For instance, in the projections of Fig. 1 we lose information associated to the opposite node set, like the degree distributions, numbers of shared nodes and their identities, etc. By intelligently assigning weights to the edges in the projection graph, we can incorporate some information about the top nodes and better reflect the underlying structure of the bigraph. In light of this, we propose a general three-step framework for projecting and classifying bigraphs aimed at dealing flexibly with the incorporation of the appropriate information for node classification:

1. First we calculate a weight for each of the top nodes in the bigraph. This weight represents the importance of the top node and the distinctiveness it has for the target variable. All the top node weights are a function of the node degree and thus retain information about the degree distributions in the projections.
2. Next, we determine the weights of the edges in the projection by combining the weights of the top nodes shared by the bottom nodes. This additionally includes information about the number of shared nodes in the projection's weights.
3. Finally, we use relational (unipartite network) classifiers on the weighted unigraphs in order to predict the values for the target variables. For this paper, the relational classifiers use only the graph structure to make predictions.[1]

We continue this section by proposing specific functions for each of the steps in the framework and explaining the rationale behind the choices. Of course, these are not the only possible choices—the list can be extended to other functions as well. This generality and flexibility is a key advantage of the framework.

3.2 Determining importance of top nodes

Functions for calculating the weights of the top nodes, s_k, are listed in Table 1 and visualised in Fig. 2. This list is not meant to be exhaustive—various other ways may exist to weight top nodes. An advantage of the component-based framework is that other methods could be introduced easily (e.g., one might have an application where it would make sense to weight based on PageRank or centrality).

Clearly, the simplest weighting scheme would be to assign equal importance, $s_k = 1$, to all the top nodes. Although this is an easy and basic method to use, it does not make any distinction between the top nodes. Other, more complex weighting methods can be proposed based on some property of the top node k, like the number of connections (degree)

[1] This is not a fundamental limitation of the framework—additional features could be constructed and more sophisticated relational classifiers could be used. We leave that for future work.

Table 1 Overview of the functions for determining top nodes weight

Top node weight function	Formula	References
Simple weight assignment	$s_k = 1$	Allali et al. (2011), Gupte and Eliassi-Rad (2012), Provost et al. (2012, (2015), Macskassy and Provost (2007)
Inverse degree	$s_k = \frac{1}{d_k}$	Gupte and Eliassi-Rad (2012), Newman (2001b)
Inverse frequency	$s_k = log_{10}\left(\frac{N}{d_k}\right)$	Martens and Provost (2011), Martens et al. (2013), Provost et al. (2012, (2015)
Hyperbolic tangent	$s_k = tanh\left(\frac{1}{(d_k)}\right)$	
Adamic and Adar	$s_k = \frac{1}{log_{10}(d_k)}$	Adamic and Adar (2003), Gupte and Eliassi-Rad (2012)
Delta	$s_k = \frac{2}{d_k(d_k-1)}$	Allali et al. (2011), Gupte and Eliassi-Rad (2012)
Beta distribution	$s_k = Beta\left(\alpha, \beta, \left(\frac{max(d_k)-(d_k)}{max(d_k)-min(d_k)}\right)\right)$	Martens et al. (2013)
Likelihood ratio	$s_k = \frac{d_k^c}{d_k}$	Martens and Provost (2011), Martens et al. (2013)

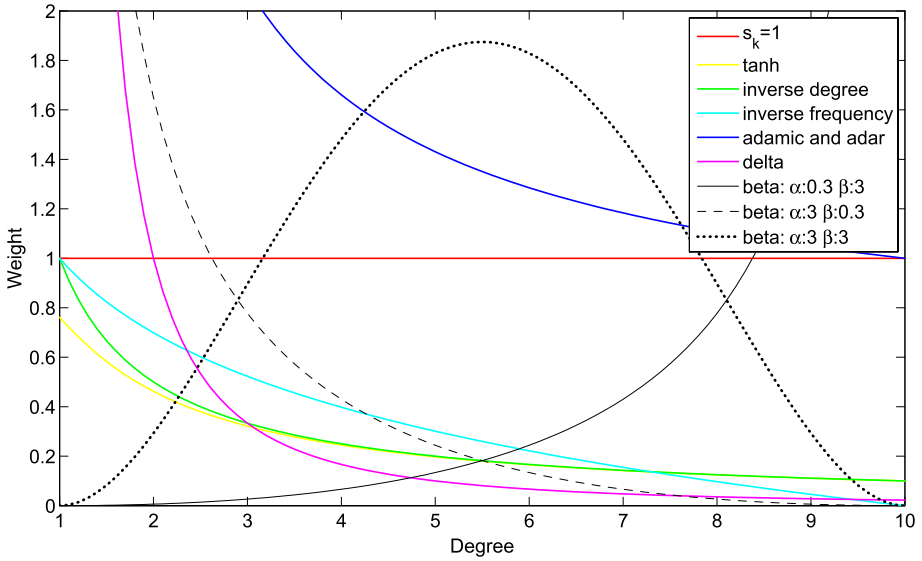


Fig. 2 Functions for determining the top nodes weight. Most of the functions follow the intuition that the lower-degree nodes are more discriminable for the target variable and therefore assign them higher weights

d_k—e.g., inverse degree, referred to as "linear" by Gupte and Eliassi-Rad (2012). Adding complexity, consider inverse degree frequency (Martens and Provost 2011), defined in analogy to a commonly used measure in information retrieval (inverse document frequency or IDF) (Jones 1972) and closely related to measures of entropy (Provost and Fawcett 2013). With IDF, very common terms that occur in many documents are assigned lower weights since they are less likely to be good discriminators. Inverse degree frequency defines the weight of a top node as a logarithmic function of the ratio between the total number of bottom nodes $n_\perp$ and the number of bottom nodes that are connected to that particular top node d_k. In the context of, for example, the users-movies network, the movies connecting fewer users provide more information for the target variable than those linking many. Users rating films noirs are more likely to have preferences in common than users rating a current blockbuster. An alternative method for weighting the top nodes is the hyperbolic tangent function. As an input to the function, we use the inverse degree of the node, based on the intuition that lower-degree nodes tend to provide higher discriminability. To our knowledge, this weighting method has not been used in prior literature and this is a first study that experiments with it. A different approach to determine the importance of the top nodes is the use of the delta function as defined in Allali et al. (2011). This function takes into account that each top node has influence on the similarity between all pairs of bottom nodes that are connected to it. Therefore, a top node with a degree d, has an equal impact on $\frac{d(d-1)}{2}$ pairs of bottom nodes. The Adamic–Adar measure (Adamic and Adar 2003) can be decomposed into a combination of the aggregation function sum of shared nodes, discussed in the next section and the associated Adamic–Adar top-node function (Table 1).

As one can observe from Fig. 2, all the functions discussed so far with the exception of the simple $s_k = 1$ assignment, follow the intuition that a top node with fewer connections creates stronger ties between the connected bottom nodes (Gupte and Eliassi-Rad 2012). In contrast, one might argue that the top nodes with very few edges are nothing more than

noise in the data and hence should not receive a high weight. Inaccuracies in data collection or the way the data are sampled could lead to a top node having a misleadingly high weight. A more flexible weighting scheme could automatically fit a function to choose an appropriate trade-off between specificity and noise tolerance (Martens et al. 2013). To this end, we employ the beta distribution density function, defined by Eq. 1, over the interval $x \in [0, 1]$, where x is the normalized top node degree (Eq. 2). Here, α and β are two parameters of the density function, which are positive numbers and define the shape of the density curve (see examples for different values in Fig. 2). The beta distribution is commonly used in Bayesian analysis as a prior distribution for binominal proportions (Forbes et al. 2011). For our purpose, the beta function provides a method for tuning the "rarity" weight to fit each dataset individually. This is done by applying a grid search to find the optimal α and β parameters for the specific dataset that provide the best predictive performance (e.g., the area under the ROC curve) on a held-out validation set.

$$Beta(\alpha, \beta, x) = \begin{cases} (1/B(\alpha, \beta))x^{\alpha-1}(1-x)^{\beta-1}, & \text{if } 0 \le x \le 1 \\ 0, & \text{otherwise} \end{cases}$$
$$B(\alpha, \beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx \tag{1}$$

$$x = \frac{d_k - min(d_k)}{max(d_k) - min(d_k)} \tag{2}$$

The likelihood ratio function, finally, takes a different approach to weighting the top nodes. It introduces supervised weighting in the projection, by taking into account how the top nodes are connected to the different classes, rather than just how they are connected in general (Martens and Provost 2011; Martens et al. 2013). The weight of a top node presents a ratio between the number of connected bottom nodes with positive class d_k^c and the total degree of the top node d_k.

3.3 Determining the link weights in the projection

Once we have determined the weights s_k of the top nodes, we continue with the second stage of the framework which calculates the link weights w_{ij} between the bottom nodes i and j in the unipartite projection. In Table 2, we present several methods for calculating w_{ij} as an aggregation of the weights from the shared top nodes s_k. The most straightforward way to combine the common top nodes is to simply sum their weights (Adamic and Adar 2003; Allali et al. 2011; Gupte and Eliassi-Rad 2012; Macskassy and Provost 2007; Martens and Provost 2011; Martens et al. 2013; Newman 2001b; Provost et al. 2009, 2012, 2015). Another approach is to select the maximum weight of the shared top nodes as a weight for the projection edges (Gupte and Eliassi-Rad 2012). We can also use an extended, weighted version of the Jaccard index, that is defined as the sum of the weights of the top nodes that are shared by both the bottom nodes, divided by the sum of the weights of the top nodes that are connected to at least one of the bottom nodes (Allali et al. 2011; Gupte and Eliassi-Rad 2012; Provost et al. 2012, 2015). A problem can arise with the Jaccard index in the case when one of the bottom nodes is connected to many top nodes and the other node is connected to only few. In that case, even when all the neighbors of one of the nodes are also neighbors of the other node, the similarity will be low. Another option for aggregating the top node weights is by employing the cosine similarity function,

Table 2 Overview of the aggregation functions

Aggregation function	Formula	References
Sum of shared nodes	$w_{ij} = \sum_{k \in N(i) \cap N(j)} s_k$	Adamic and Adar (2003), Allali et al. (2011), Gupte and Eliassi-Rad (2012), Macskassy and Provost (2007), Provost et al. (2009), Martens and Provost (2011), Martens et al. (2013), Newman (2001b), Provost et al. (2012, (2015)
Max of shared nodes	$w_{ij} = \max_{k \in N(i) \cap N(j)} s_k$	Gupte and Eliassi-Rad (2012)
Jaccard similarity	$w_{ij} = \frac{\sum_{k \in N(i) \cap N(j)} s_k}{\sum_{k \in N(i) \cup N(j)} s_k}$	Allali et al. (2011) ,Gupte and Eliassi-Rad (2012), Provost et al. (2012, (2015)
Cosine similarity	$w_{ij} = \frac{\sum_{k \in N(i) \cap N(j)} s_k^2}{\sqrt{\sum_{k \in N(i)} s_k^2} \cdot \sqrt{\sum_{k \in N(j)} s_k^2}}$	Provost et al. (2009, (2012, (2015)
Zero-one	$w_{ij} = \begin{cases} 1 & \text{if } \sum_{k \in N(i) \cap N(j)} s_k > 0 \\ 0 & \text{if } \sum_{k \in N(i) \cap N(j)} s_k = 0 \end{cases}$	Guillaume and Latapy (2006), Newman (2010a), Provost et al. (2009)

which calculates the similarity between pairs of vectors as the cosine value of the angle between them (Provost et al. 2009, 2012, 2015). Using this measure, the similarity between two bottom nodes will be the highest and equal to one when they share exactly the same top nodes and equal to zero when they don't have any neighbors in common. Finally, a very simple weighting measure assigns the value of 0 or 1 to the links in the projection, depending on whether the bottom nodes have at least one shared top node or not (Guillaume and Latapy 2006; Newman 2010a; Provost et al. 2009). This corresponds to an unweighted version of the projection graph, so it loses all the information related to the strength of the bonds between pairs of bottom nodes.

3.4 Relational classifiers

The third step of the framework for node classification within bigraphs is to use a relational (network) classifier over the unigraph projection. We consider methods for within-network classification in univariate networks, as defined by Macskassy and Provost (2007) (see Sect. 2.2). Based on the performance of the classifiers in their analysis, we will consider the following relational classifiers which dominated in the study.

The *weighted-vote Relational Neighbor (wvRN)* classifier (Macskassy and Provost 2003) is a straightforward classifier that uses the known class labels of the related nodes (or predictions thereof) to make a probability estimation (score) of the node's own class label (see Eq. 3). It is based on the assumption of assortativity, that the linked nodes in the graph are likely to be of the same class. The classifier calculates the node's score $P(l_i = c|N(i))$ as a weighted average of the neighbors' scores.

$$P(l_i = c|N(i)) = \frac{1}{Z} \sum_{j \in N(i)} w_{ij} \cdot P(l_j = c) \tag{3}$$

In this equation, Z is the normalizing factor and is equal to the sum over the link weights to all neighboring nodes ($\sum_{j \in N_{(i)}} w_{ij}$).

The second relational classifier used in this study is the *class-distribution Relational Neighbor (cdRN)* classifier (Macskassy and Provost 2007). Unlike the previous classifier, it takes into account the class distribution linkages of the whole training set, and not only the neighborhood of the focal node, through class-specific "reference vectors". First, a class vector $CV(i)$ is created for each node as a sum of the links' weights to other nodes with each known class (l_j) (Eq. 4). The class vectors of the training nodes are then aggregated into reference vectors for the different classes $RV(c)$ and represent an average of the $CV(i)$ for nodes known to be of class c (Eq. 5).

$$CV(i)_c = \sum_{j \in N(i), l_j = c} w_{ij} \tag{4}$$

$$RV(c) = \frac{1}{| \perp^c |} \sum_{i \in \perp^c} CV(i) \tag{5}$$

In this equation, $\perp^c$ denotes the bottom nodes in the bigraph known to have label $l_i = c$.

The probability of a node i having class c can then be estimated as the normalized vector similarity between the class vector of node i ($CV(i)$) and the reference vector RV (Eq. 6).

The vector similarity function we use is cosine similarity, but other functions such as L1 or L2 similarity, scaled to the range of [0,1], can also be applied.

$$P(l_i = c|N(i)) = sim(CV(i), RV(c)) \tag{6}$$

A more complex relational classifier is the *network-only Link-Based classifier (nLB)* (Macskassy and Provost 2007; Lu and Getoor 2003). This classifier builds a class vector $CV(i)$ for every training node i in the network, that contains scores for each label class c. Since we only consider binary bigraphs, the class vector for a training node is a vector with two elements, that are the scores for both classes c_0 and c_1. The class-specific scores are calculated in the same way as for the wvRN classifier, also known as the count model in the study of Lu and Getoor (2003) (Eq. 7). In the next step, the nLB classifier builds a logistic regression model based on these class vectors (Eq. 8).

$$CV(i)_c = \frac{\sum_{j \in N(i)} w_{ij} \cdot P(l_j = c)}{\sum_{j \in N(i)} w_{ij}} \tag{7}$$

$$P(l_i = c|N(i)) = \frac{1}{1 + e^{-\beta_0 - \beta CV(i)}} \tag{8}$$

3.5 Decomposition of metrics

In this study, we consider a wide range of functions for creating the weights of the projection. To the best of our knowledge, we apply all the methods that were previously used in the literature for defining the link weights in bigraph projections and that can be decomposed within our framework. In Table 3, we present a summary of the measures used in prior literature, divided in the three stages: top nodes weighting function, aggregation function and relational classifier. The formula for decomposition is given in Eq. 10, where g represents the aggregation function and f the top node weighting function. As an example, the Adamic–Adar coefficient (Adamic and Adar 2003) (see Eq. 9), can be decomposed into the Adamic–Adar top node weighting function (Table 1) and the sum of shared nodes aggregation function (Table 2).

$$w_{ij} = \sum_{k \in N(i) \cap N(j)} \frac{1}{\log_{10}(d_k)} \tag{9}$$

$$w_{ij} = g(s_{k1}, s_{k2}, ..., s_{kn}) = g(f(d_{k1}), f(d_{k2}), ..., f(d_{kn})) \tag{10}$$

This decomposability creates opportunities for combining the existing weighting functions in new ways, resulting in completely new measures. Note that some of the combinations from prior literature do not include a relational classifier, since these studies use the unigraph projection for other tasks rather than classification, like link prediction (Allali et al. 2011; Gupte and Eliassi-Rad 2012), measuring descriptive statistics (Guillaume and Latapy 2006; Newman 2010a, 2001b), etc. Moreover, some studies consider the unweighed unigraph projections (Guillaume and Latapy 2006; Newman 2010a). Since this is independent of the top nodes weight, we can apply any top-node function in the first step of the framework.

Table 3 Overview of measures for defining links' weights in bigraph projections used in previous literature

Top node weight	Aggregation function	Rel. classifier	References
Any	Zero-one	–	Guillaume and Latapy (2006), Newman (2010a)
Simple weight	Sum of shared nodes	–	Allali et al. (2011), Gupte and Eliassi-Rad (2012)
Inverse degree	Sum of shared nodes	–	Gupte and Eliassi-Rad (2012), Newman (2001b)
Adamic and Adar	Sum of shared nodes	–	Adamic and Adar (2003), Gupte and Eliassi-Rad (2012)
Delta	Sum of shared nodes	–	Allali et al. (2011), Gupte and Eliassi-Rad (2012)
Simple weight	Jaccard similarity	–	Allali et al. (2011), Gupte and Eliassi-Rad (2012)
Inverse degree	Max of shared nodes	–	Gupte and Eliassi-Rad (2012)
Simple weight	Sum of shared nodes	wvRN	Macskassy and Provost (2007), Provost et al. (2012, (2015)
Inverse frequency	Sum of shared nodes	wvRN	Provost et al. (2012, (2015)
Inverse frequency, likelihood ratio	Sum of shared nodes	wvRN	Martens and Provost (2011), Martens et al. (2013)
Beta distribution, likelihood ratio	Sum of shared nodes	wvRN	Martens et al. (2013)
Simple weight	Jaccard similarity	wvRN	Provost et al. (2012, (2015)
Inverse frequency	Jaccard similarity	wvRN	Provost et al. (2012, (2015)
Simple weight	Cosine similarity	wvRN	Provost et al. (2012, (2015)
Inverse frequency	Cosine similarity	wvRN	Provost et al. (2012, (2015)
Simple weight	Sum of shared nodes	cdRN	Macskassy and Provost (2007)
Simple weight	Sum of shared nodes	nLB	Macskassy and Provost (2007)

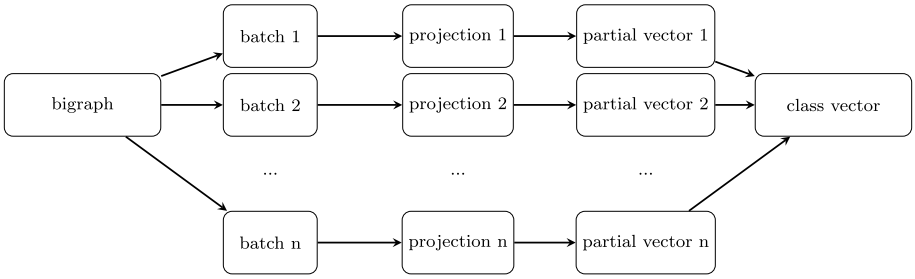


Fig. 3 Overview of the batch process for the nLB classifier. With random sampling a subset of the bigraph is selected. The projection is created for each batch, to calculate a part of the class vector. Subsequently, we assemble all the partial vectors into a whole class vector that is used by the logistic regression

3.6 Scalability

Above we discussed how bigraphs are a natural and efficient representation for sparse feature data, and indeed the sparse representation commonly used by machine learning methods is in fact a (possibly weighted) bigraph adjacency list. The projection itself can reduce the data size, but only sometimes (e.g., when the connections are sparse and the number of top nodes is much larger than the number of bottom nodes). This notwithstanding, the scalability of the algorithms nonetheless deserves special attention since bigraph data and their corresponding unigraph projections often are very large; methods are needed that can deal with massive data. In this section we propose several techniques that enable the algorithms to scale up to very large datasets and/or improve their run-time performance.

3.6.1 Batch processing

Large datasets that can not be processed in memory can be divided into smaller, processable subsets called batches. In this paper, *batch processing* means that the label scores will be produced by processing the batches one at a time, either sequentially or in parallel, instead of processing the whole dataset at once (Provost and Kolluri 1999). For instance, in the case of the network-only Link-Based (nLB) classifier (see Eq. 7), we create a partial projection for each batch from the training dataset and then use it to calculate a part of the class vector. Subsequently, we assemble all the partial vectors into a whole class vector that is used by the logistic regression (see Fig. 3). In the same manner, we create partial label scores from each batch of the test set and then we aggregate all the scores into a final solution. The sizes of the batches in this study were determined experimentally, by testing different sizes for each dataset. We need to be careful when choosing the size of the batch; on one hand if the size is too large it will not be possible to process the dataset in main memory. Instead, the CPU will thrash, wasting substantial time swapping memory blocks between RAM and disc, which will degrade runtime performance substantially. On the other hand, choosing a size of the batch that is too small will add time to the process, since each fragment introduces additional calculation overhead. Therefore, the size should be a balance between the two. Batch processing would also allow for easy scaling up using parallel and distributed computing systems (Provost and Kolluri 1999).

3.6.2 Sampling

Another technique that enables the network-only Link-Based (nLB) classifier to scale to larger datasets and improve the run-time performance is *sampling*. Since the number of features used by nLB is usually very small, instead of building class vectors for all the training nodes, we can use only a subset of the training nodes to train the classifier. In our experiments, we observed that a sample of around 100 training instances was usually sufficient for training the classifier. Therefore, in the experimental set-up we run nLB with and without sampling and compare the results.

3.6.3 Grid search

Fine tuning the parameters of the top nodes beta function (Eq. 1) can require many iterations. In order to reduce the number of iterations, a *grid search* with multiple levels[2] can be employed. The idea is that every level of the search performs a more fine-grained lookup around the best selected parameters' values from the previous level. If, for instance, our search on level i with step s determined that x is the best value for a parameter, then the following level $i+1$ will look for a better value in the range $[x-s, x+s]$. The size of the step, i.e. the coarseness of the search, decreases in each new level.[3] This yields significant runtime improvements in comparison to extensive search of the space, while giving the same solution granularity. As we discuss further in Sect. 5.2, one can also perform a grid search with fewer levels that results in limited performance degradation, or incorporate knowledge about the most suitable beta shapes in the search procedure.

3.6.4 SW transformation

Finally, we introduce a fast method, called *SW-transformation*, to calculate the label scores for the case where wvRN (Eq. 3) is combined with an aggregation function that sums the weights of the top nodes. Hence the name SW-transformation, which is an acronym of the two methods involved: the sum of shared nodes and the wvRN.

To calculate the projection using the sum of shared nodes we find for each bottom node ($n_\perp$) all the top nodes it is connected to (on average $k_\perp$) and then look at all the nodes connected by the top node (on average $k_\top$). This results in a time complexity of $O(n_\perp \cdot k_\perp \cdot k_\top) = O(m \cdot k_\top)$, with m the number of edges. For sparsely connected graphs, or any graph where a node has no more than some constant maximum number of connections (i.e. $k_\perp$ and $k_\top$ are bounded by a constant), this scales linearly with the number of bottom nodes ($O(n_\perp)$). In the case of fully connected graphs ($k_\perp = n_\top$ and $k_\top = n_\perp$) however, the complexity is $O(n_\perp^2 \cdot n_\top)$. Consequently, the wvRN is applied to this projection, which calculates the node's score as a weighted average of the neighbors' scores. The time complexity of the wvRN classifier scales linearly with the number of edges in the projection ($O(n_\perp \cdot k_p) = O(m_p)$) and is no larger than the time complexity of the projection. Again, when the average degree in the projection (k_p) is bounded by a constant this becomes $O(n_\perp)$ and when the graph is fully connected it is $O(n_\perp^2)$.

[2] In our case the beta function is tuned in three levels.

[3] The grid we use searches for the optimal α and β in the range between 0.1 and 12.1 with steps of 3 in the first level. We decrease the step size in each successive level by 3 times.

This specific combination can be rewritten as a fast linear model over the top nodes. We rewrite the wvRN formula as follows:

$$Z \cdot P(l_i = c|N(i)) = \sum_{j \in N(i)} w_{ij} \cdot P(l_j = c) \tag{11}$$

$$= \sum_{j|a_{ij} \neq 0} w_{ij} \cdot y_j \tag{12}$$

$$= \sum_{j|a_{ij} \neq 0} \left(\left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot y_j + \left(\sum_{k|x_{ik} = 0} s_k \right) \cdot y_j \right) \tag{13}$$

$$= \sum_{j|a_{ij} \neq 0} \left(\left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot y_j + 0 \right) \tag{14}$$

$$= \sum_{\substack{j|a_{ij} \neq 0 \\ y_j = 0}} \left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot y_j + \sum_{\substack{j|a_{ij} \neq 0 \\ y_j = 1}} \left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot y_j \tag{15}$$

$$= 0 + \sum_{\substack{j|a_{ij} \neq 0 \\ y_j = 1}} \left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot 1 \tag{16}$$

$$= \sum_{k|x_{ik} \neq 0} \left(s_k \sum_{\substack{j|a_{ij} \neq 0 \\ y_j = 1}} 1 \right) \tag{17}$$

$$= \sum_{k|x_{ik} \neq 0} \left(s_k \sum_{\substack{j|x_{ik} \neq 0, x_{jk} \neq 0 \\ y_j = 1}} 1 \right) \tag{18}$$

$$= \sum_{k|x_{ik} \neq 0} s_k \cdot ns_k \tag{19}$$

In this formula we first substitute the projection weights w_{ij} with the corresponding aggregation function (Eq. 11). Since wvRN takes into account only the labels of the neighboring nodes, in Eq. 12 we consider only the bottom nodes j that have an element $a_{ij} = 1$ in

$$\begin{array}{c} \\ b_1 \\ b_2 \\ \\ b_i \\ \\ b_m \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & \dots & t_k & \dots & t_n \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1k} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2k} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ik} & \dots & x_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mk} & \dots & x_{mn} \end{pmatrix} \end{array}$$

(a) $m \times n$ matrix representation of a bigraph.

$$\begin{array}{c} \\ b_1 \\ b_2 \\ \\ b_i \\ \\ b_m \end{array} \begin{array}{c} \begin{array}{cccccc} b_1 & b_2 & \dots & b_j & \dots & b_m \end{array} \\ \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{im} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mm} \end{pmatrix} \end{array}$$

(b) $m \times m$ matrix representation of the projected unigraph.

Fig. 4 Adjacency matrices of the bigraph and the projected unigraph

the unigraph adjacency matrix (see Fig. 4, right). The link weight w_{ij} in the projection is calculated by summing the weights of the top nodes that are shared by both nodes i and j. This means that the non-neighboring top nodes of node i (that have elements $x_{ik} = 0$ in the bigraph adjacency matrix from Fig. 4, left) can be discarded in Eq. 14. When predicting an attribute of a node, the wvRN takes into account only the neighboring nodes that have the label class of interest ($y_{ij} = 1$). In our study, since we consider a binary target variable, this leads to eliminating the neighboring nodes with label $y_{ij} = 0$ in Eq. 16.[4] Note that the neighboring nodes with label $y_{ij} = 0$ are still counted when calculating the normalization factor Z in Eq. 11. The result is the SW-transformation (Eq. 19), a linear model that computes the label scores directly on the bigraph and avoids the costly step of calculating the projection unigraph. In terms of implementation, the transformation corresponds to a linear model where the "coefficients" of each feature can easily be calculated by multiplying the score s_k of a feature/top node and the number of datapoints ns_k that have a positive value for that feature and a non-zero value for the target variable. Doing so for each of the $n_\top$ top nodes, we only need to consider the (on average $k_\top$) non-zero elements and the time complexity becomes $O(n_\top \cdot k_\top) = O(m)$. For bounded graphs this becomes $O(n_\top)$, for fully connected graphs it is $O(n_\top \cdot n_\bot)$. When working with sparse graphs (which is the focus of our study) both approaches scale linearly (either with the number of bottom nodes or the number of top nodes) but the SW transformation always runs faster than the combination of the projection and wvRN classifier (as we also show empirically, see Fig. 8).

In a similar manner, the normalization factor Z can be transformed into $Z = \sum_{k|x_{ik}\neq 0} s_k \cdot Zs_k$, where $Zs_k = |x_{jk} = 1|$. Finally, we can write:

$$P(l_i = c|N(i)) = \frac{\sum_{k|x_{ik}\neq 0} s_k \cdot ns_k}{\sum_{k|x_{ik}\neq 0} s_k \cdot Zs_k} \tag{20}$$

In this manner, we directly calculate the influence of the top node, in the form of the coefficient of the corresponding linear model[5]. The SW-transformation yields substantially faster run times (compared to calculating the whole projection) and allows easy scaling of the method to big data sets of millions of nodes, as we discuss further in Sect. 5. Large, sparse

[4] The transformation can also be applied to cases where the node labels are score probabilities.

[5] A Python implementation of the SW-transformation is available online on https://github.com/SPraet/SW-transformation.

data sets with a bipartite structure have become common, largely due to the recording of behaviors of individuals (or things); these data sets often are very sparse (Junqué de Fortuny et al. 2013), with nodes being connected to only a few other nodes. If this sparsity extends to the projection, the SW-transformation may be quite advantageous.

4 Data and experimental setup

For this study, we collected bipartite datasets from various sources: the Koblenz Network Collection (KONECT),[6] the MIT Reality Mining Project,[7] the social networks collection of The Max Plank Institute for Software Systems,[8] and more. We selected all datasets where a bipartite structure is clearly present and a target variable is available to predict. Note that in some cases we discard a dataset because the class variable is related to the links in the bigraph. For instance, predicting the number of books that the users have read from a bigraph of users rating books is clearly not suitable. The datasets are summarized in Table 4 and to our knowledge comprise the first large collection of benchmark datasets for node classification over bigraphs.

The *MovieLens* dataset contains information about movie ratings from users of the MovieLens website, collected from September 1997 through April 1998.[9] The bigraph is defined between users and movies, where links are present if a user rated a movie. We focus on the task of predicting the genre of the movie, as well as the gender and the age of the user. In the first case, the movies are considered as bottom nodes and the users as top nodes, for the latter it is vice versa. For multiclass problems (as genre), we use a one-versus-all formulation and as such define as many additional datasets as there are classes (19 in this case).[10] The *Yahoo Movies* dataset[11] has a similar setting, where we again are predicting the gender and the age of users who rated movies. Likewise, *Book-Crossing* contains book ratings from the web site Bookcrossing.com (Ziegler et al. 2005) and our aim is to predict the age of the readers. The dataset collected by Seierstad and Opsahl (2011) is used for defining a bigraph between *Norwegian companies* and their board members, and the target variable is the gender of the board members. Furthermore, we use the information about mobile phone usage collected by the MIT Human Dynamics Lab (*Reality Mining* project) (Eagle and Pentland 2006) to define a bigraph of users connected to the locations (cell towers) they visited. The target variable is the affiliation of the user, being student, laboratory staff, professor, etc. Another bigraph is defined from the *LibimSeTi* (Brozovsky and Petricek 2007) dataset, that contains data about profile ratings from users of the Czech social network LibimSeTi.cz. The prediction task in this case is the gender of the users. *Ta-Feng* is a dataset of supermarket transactions, where we predict the age of the customers, based on the products they bought (Huang et al. 2005). A dataset from the Max Plank Institute for Software Systems contains data about users and several million

[6] http://konect.uni-koblenz.de.
[7] http://realitycommons.media.mit.edu.
[8] http://socialnetworks.mpi-sws.org.
[9] http://www.grouplens.org.
[10] For this paper, we only consider binary classification, where multiclass problems are cast to several one-versus-all binary classification problems. Other approaches to multiclass problems can easily be incorporated within our proposed framework. For more details, see Sect. 6.
[11] http://webscope.sandbox.yahoo.com/.

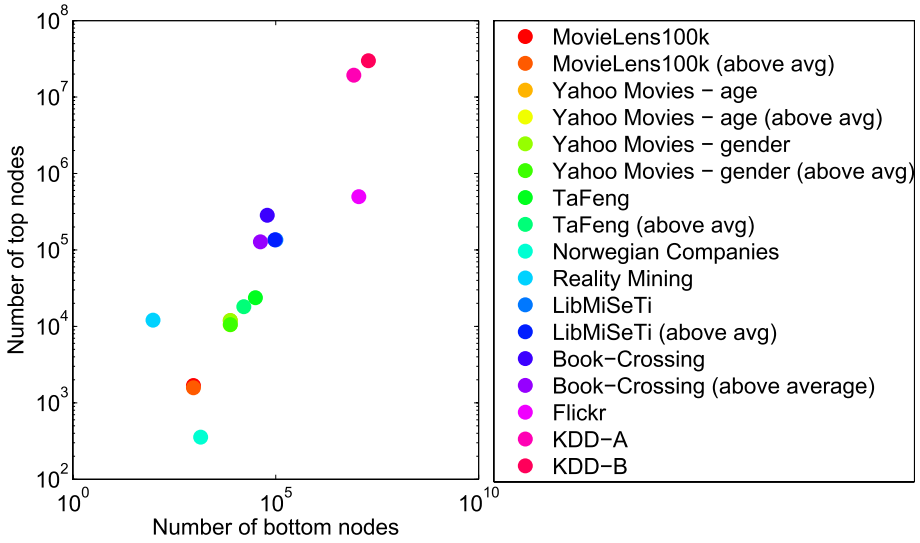


Fig. 5 Size of the datasets under study

Flickr pictures, creating a bigraph of pictures and the users that mark them as favorites. The target class variable is the number of comments on the pictures. The largest datasets used in this study are from the KDD Cup 2010, where the participants were asked to predict student performance on tests. The winner of the Cup, the National Taiwan University, expanded the original dataset by converting the categorical features into sets of binary features (Yu et al. 2010) and this version of the data can be downloaded from the LibSVM website.[12] In addition to the previously described datasets, we also created new bigraphs for the rating data, where a connection exists between the nodes only if the rating was positive (defined as higher than the average rating). We annotate these bigraphs as "above average" in Fig. 5 and Table 4.

Figure 5 gives an overview of the number of nodes present in the bigraphs under study. As shown in the plot, the sizes of the bigraphs differ, with some datasets having fewer than 100 bottom nodes (Reality Mining, number of people involved) and a few hundred top nodes (Norwegian companies, number of companies involved) and others with up to a few million top and bottom nodes (KDDa and KDDb datasets). In the "Appendix" (Figs. 9, 10, 11 and 12), we also examine the distributions of the probability $P(k)$ that a node has a degree k (also known as degree distributions) for the bottom and top nodes of each dataset. As one can observe, most of the datasets show a heavy-tailed degree distribution, resembling the typical power-law with different exponents. In such distributions, many nodes in the bigraph are connected only to few nodes from the opposite set, but a non-negligible number of nodes are connected to many other nodes. The top nodes of the LibMiSeTi dataset do not follow the power law shape: most of the profiles in the social network get an average number of rankings from the other users, similarly for the Yahoo Movies dataset. The bottom nodes of the KDDa dataset are an exception as well, which might be due to the fact that it is an artificially created dataset.

[12] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Table 4 Descriptive statistics of the bipartite datasets: class distribution (l_0, l_1), number of top ($n_\top$) and bottom ($n_\bot$) nodes, number of edges (m), average degree for top ($k_\top$) and bottom ($k_\bot$) nodes, average combined degree (k) and density ($\delta(G)$). The basic bipartite statistics used in this table are defined in Sect. 3

Dataset	Target label	l_0	l_1	$n_\top$	$n_\bot$	m	$k_\top$	$k_\bot$	k	δ
MovieLens100k	Gender	273	670	1682	943	100,000	59.45	106.04	76.19	0.063
MovieLens100k	Age	448	495	1682	943	100,000	59.45	106.04	76.19	0.063
MovieLens100k	Genre	–	–	943	1682	100,000	106.04	59.45	76.19	0.063
MovieLens100k (above average)	Gender	273	670	1574	943	82,520	52.43	87.51	65.57	0.0556
MovieLens100k (above average)	Age	448	495	1574	943	82,520	52.43	87.51	65.57	0.0556
Yahoo Movies	Gender	2206	5436	11,915	7642	221,330	18.57	28.96	22.63	0.0024
Yahoo Movies (above average)	Gender	2206	5431	10,547	7637	181,470	17.20	23.76	19.96	0.0023
Yahoo movies	Age	2750	4855	11,911	7605	220,595	18.52	29.01	22.61	0.0024
Yahoo movies (above average)	Age	2748	4852	10,544	7600	180,880	17.15	23.80	19.93	0.0023
TaFeng	Age	17,330	14,310	23,719	31,640	723,449	30.5	22.86	26.14	9.6400e–04
TaFeng (above avg)	Age	5051	11,299	18,126	16,350	234,355	12.93	14.33	13.59	7.9078e–04
Norwegian companies	Gender	513	908	355	1421	1746	4.92	1.23	1.97	0.0035
Reality mining	Affiliation	–	–	12,043	95	76,674	6.37	807.09	12.63	0.067
Book-crossing	Age	38,168	23,662	284,175	61,830	835,495	2.94	13.51	4.83	4.7551e–05
Book-crossing (above average)	Age	25,729	16,421	127,709	42,150	259,333	2.03	6.15	3.05	4.8177e–05
LibimSeTi	Gender	43,510	57,606	135,359	101,116	13,594,717	100.43	134.45	114.98	9.932e–004
LibimSeTi (above average)	Gender	40,878	54,459	135,346	95,337	8,169,662	60.36	85.69	70.83	6.3314e–04
Flickr	Comments	8,177,007	3,030,449	497,472	11,195,144	34,645,469	69.64	3.09	5.926	6.2208e–06
KDDa	Task performance	1,235,867	7,171,885	19,306,083	8,407,752	305,613,510	15.82	36.34	22.05	1.8828e–06
KDDb	Task performance	2,684,437	16,579,660	29,890,095	19,264,097	566,345,888	18.94	29.39	23.04	9.8357e–07

Table 5 Kemeny–Young ranking per method on all datasets

Kemeny Ranking	Top node func.	Aggregation func.	Relational classifier
1	**tanh**	**Cosine function**	**wvRN**
2	*Inverse degree*	**Sum of shared nodes**	**nlb**
3	*Inverse frequency*	Jaccard	cdRN
4	*Beta distribution*	Max	nlb 100
5	w = 1	Zero–one	
6	Adamic and Adar		
7	Delta		
8	Likelihood ratio		

We emphasize the combinations that are not significantly worse than the best method (underlined) at a 5% significance level in bold and the combinations that are significantly worse at 5% but not at 1% significance level in italics. The other methods that are significantly worse at 1% significance level are shown in regular font

5 Results and discussion

In this section, we present the results of the benchmark study examining predictive performance and run-time performance. We report results for each combination of top node weighting scheme, aggregation function to define the weight in the projected graph, and relational classifier. This leads to a total of $8 \times 5 \times 4$ combinations, that are assessed on the 58 datasets (including the casted multiclass datasets). As a benchmark technique, we employ an SVM with a linear kernel and L2 regularization on the bigraph adjacency matrices using the libLINEAR toolbox (Fan et al. 2008). As mentioned in Sect. 2, linear SVM is a commonly-used and well-performing method for sparse classification problems and was therefore chosen to benchmark our proposed technique against. Based on the results of our benchmark we formulate some general recommendations for node prediction in bipartite graphs.

5.1 Predictive performance

Table 7 in the "Appendix" presents the predictive performance results for every combination of techniques, based on the area under the ROC curve (AUC) (Fawcett 2006). To report AUC values, we run a 10-fold cross-validation procedure. We divide the dataset into 10 subsamples of equal size. In each run we use 8 subsamples for training data, 1 for a validation set (for hyperparameter tuning) and 1 for a test set. After 10 runs, each of the 10 data subsamples is used exactly once for testing and once for validation in the process. The reported AUCs represent an average over the 10 folds. For every dataset, we rank the performance of the techniques into a partial ranking and then combine them together into a final ranking using Kemeny–Young optimisation (Conitzer et al. 2006; Young and Levenglick 1978). The goal of the Kemeny–Young method is to find an ordering of the techniques that minimizes the number of pairwise disagreements between the final ranking of the techniques and the partial rankings calculated on the individual datasets. Note that when calculating the final ordering, we also consider the datasets where not all combination schemes were able to scale. More specifically, the aggregation functions Jaccard, Cosine similarity and Maximum do not scale well to

datasets with high dimensions such as the Flickr dataset or larger. Also, a combination of these aggregation functions and the beta function, which employs multiple iterations to tune the parameters, takes very long time to fit for datasets over 100,000 nodes. In such cases, when a dataset does not provide a ranking for one or for both methods that are being compared, then that dataset does not contribute to the total disparity regarding these two techniques. For statistical comparison of the methods, we use a Wilcoxon signed rank test (Demšar 2006) to assess the significant differences between the best performing method and the other classifiers.

From Table 5, we can observe that the highest ranked combination that performs very well over all the datasets is the tangens hyperbolicum function, combined with the cosine aggregation function and the wvRN. Furthermore, there are also a few alternatives that provide comparable results to this top ranked combination. If we take a closer look at the results per dataset (Table 9 in "Appendix"), we can see that generally combinations that include the cosine or sum of shared nodes aggregation functions together with the tangens hyperbolicum, inverse degree and occasionally the beta function provide very good results when combined with any of the relational classifiers. The SVM benchmark against which we compare the network projection methods, has only average performance. It is ranked on the 98th place out of 161 possible techniques and it is significantly worse than the best method at 1% significance level. Additionally the SVM would not run on the big KDDa and KDDb datasets. Although faster implementations of SVM exist, e.g. using stochastic gradient descent, these would come at the expense of predictive performance (de Cnudde et al. 2017).

In the following sections we discuss the predictive performance for our three stages separately.

5.1.1 Predictive performance of the top node weight functions

The rankings of the top-node functions are summarized in Table 5, with the tangens hyperbolicum and the inverse degree (both similar in shape, see Fig. 2) providing the best performance across all domains. We should, however, be careful when interpreting these results and not simply discard top node methods that provide poorer results over all domains. Although specific combinations that include the top-node function can have very strong performances (see Table 7), the overall rankings still get diluted by the weaker combinations (for instance, the ones containing the zero-one aggregation function). If we take a closer look at Table 9 in "Appendix", where we list the best combinations of techniques per dataset, one can easily notice that in most cases the best performing combination contains the beta distribution as an appropriate choice. By analysing the optimal α and β coefficients for the different datasets (listed in Table 8), we conclude that the typical shapes of the function correspond strongly to the intuition that top nodes with smaller degree are more discriminative and therefore should have higher weights. The only exception is the Flickr dataset, where the parameters define the opposite curve of the beta function. Having in mind the prediction task for this dataset, it makes sense that a popular picture that has more users who marked it as favorite,

would receive more comments than a picture with fewer markings. Based on the results from Table 9, we also conclude that the supervised weighting function, likelihood ratio, exhibits very good performance for skewed datasets with a small number of positive labels.[13] What is specific about this function is that it weights only the top nodes that are connected to at least one bottom node with a positive label. This results in projections where the links to some of the neighboring nodes with negative labels are down-weighted, since the top nodes connecting only negative bottom nodes do not contribute to the projection weights.

5.1.2 Predictive performance of the aggregation functions

Table 5 also presents an overview of the results per aggregation function, with the cosine function and the sum of shared nodes as the most suitable methods. Although both functions provide very good performance, the latter is favorable since it can be combined with the wvRN to scale easily to very large datasets (SW-transformation). All the functions perform much better than the zero-one function, which corresponds to an unweighted version of the projection. This indeed supports strongly the idea that adding weights to the projection reflects better the structure of the underlying bigraph and therefore results in better predictions. The Jaccard aggregation function does not perform well, as it penalizes the score if one of the nodes has many links. As an example, let us consider again the bigraph of people visiting locations, with person A visiting 5 different locations; person B visiting these 5 locations and 10 more; and person C visiting the same 5 locations and 100 more. In this case, the Jaccard would penalize the AC link with a much lower score than the AB link, because of the metric's denominator which takes into account all the locations visited by at least one of the persons. This does not make sense for this setting: if we have a total of (for example) a million locations, the odds for visiting the same 5 locations by chance are very small. The max function also shows poor performance, which supports the idea that it is valuable to retain information for more than just one top node.

5.1.3 Predictive performance of the relational classifiers

In Table 5, we can also see the aggregated results over the relational classifiers, where the best classifier wvRN slightly outperforms the nLB. These two classifiers provide similar results in cases with relational autocorrelation (Jensen and Neville 2002) over the target values in the projection. An example of positive relational autocorrelation would be: if I like the same movies as you, we likely are of the same gender. Yet, the opposite can be true as well. For example, in the case of the Norwegian companies dataset, a man is more likely to be in a board with a female and vice versa. Because of the negative relational autocorrelation, the wvRN here yields an average AUC (over all combination schemes) of only 0.2728. This substantially hurts the wvRN average scores. However, as AUCs systematically below 0.5 can be "flipped" to sometimes strong AUCs, this result requires some extra explanation.

Norway is one of the leading countries that enforces equal gender representation in companies' boards (Seierstad and Opsahl 2011), which results in the companies (top

[13] All the datasets for which this function performed best have only between 3.19 and 7.25% positive labels.

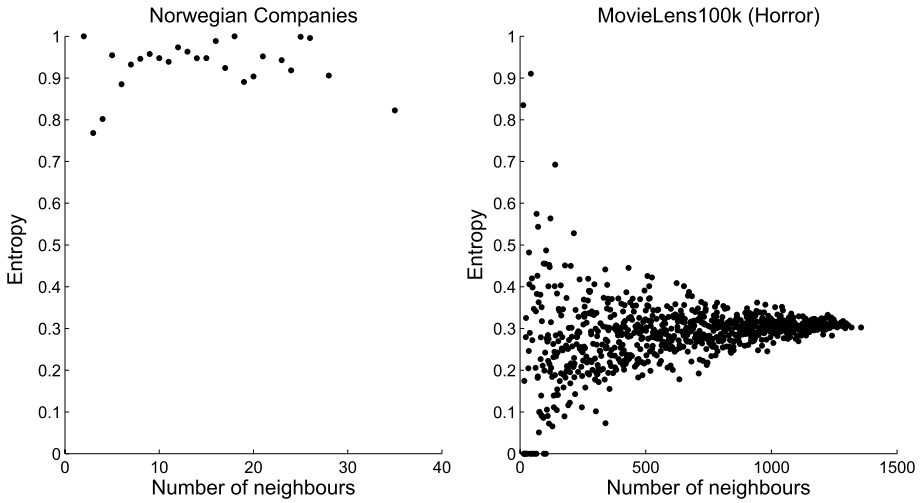


Fig. 6 Average entropy per number of neighbors in the projection. The Norwegian Companies dataset (left) has high average entropy, whereas the MovieLens dataset with target variable Horror genre (right) has lower entropy. The wvRN classifier performs better on datasets with lower entropy

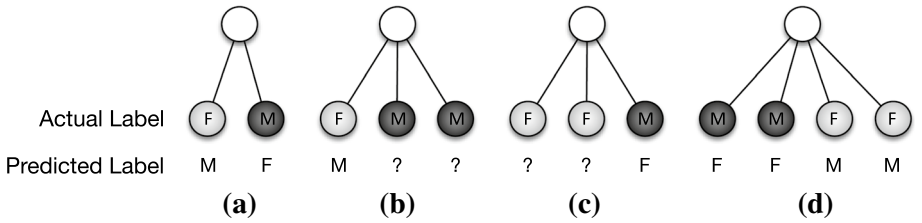


Fig. 7 Bigraph structures of companies (top nodes) and board members (bottom nodes). The node letter presents the actual gender of the board member and below is the predicted gender by the wvRN

nodes) being connected to almost the same number of male and female directors (bottom nodes). In Fig. 6, we use entropy as a measure for class imbalance (Rrnyi 1961), to calculate the heterogeneity of the target variable among the nodes' neighbors in the projection. Very high values of entropy signify that there is nearly an equal number of neighbors from each class, whereas low values suggest that almost all the adjacent nodes have the same class. The results are averaged over the nodes that have the same number of neighbors. As expected, the dataset of Norwegian companies exhibits high entropy values for all the board members (see Fig. 6, left). In comparison, a typical dataset where wvRN performs better has much lower entropy (see Fig. 6, right).[14] In such cases where the class distribution of a node's neighbors is approximately 0.5, cross-validation can cause pathologies in machine-learning evaluations (Perlich and Świrszcz 2011). Consider the following.

Most of the directors (83.6%) are members of the board of only one company and most companies (71%) have only up to 5 board members. This creates many small disconnected

[14] Note that in addition to the entropy, the weights of the links also have impact on the prediction performance of wvRN.

components in the bigraph, like the ones depicted in Fig. 7. When the wvRN relational classifier is applied with cross-validation, it is likely that the focal node's target class will be underrepresented in the remaining neighbor nodes. For example, consider a leave-one-out evaluation. In case (a), a member will be connected to only one member of opposite sex in the projection, hence wvRN will predict the wrong class. In the other cases depicted in Fig. 7, wvRN will predict the majority opposite class or give a score of 0.5 when the remaining classes are balanced (denoted with a question mark). Since it is difficult to know exactly how justifiably to tinker with these results, we will simply leave them as they are. This may possibly penalize wvRN in these cases and artificially bolster the performance of the learning-based methods, or it may be exactly what we would like to happen in these cases. The learning-based classifiers are able to pick up on this: the nLB classifier provides a negative coefficient to the female class distribution for males (and again vice versa), which leads to an AUC of 0.7029 and the cdRN creates reference vectors that take into account how the training nodes are connected to the opposite class, yielding an average AUC of 0.6997. Based on the analysis, we see that wvRN is an appropriate choice for problems that exhibit network assortativity; however, the nLB and cdRN are more powerful and can capture more complex patterns [as discussed in the original paper of Macskassy and Provost (2007)]. The nLB classifier trained with only 100 instances, nLB100, is a much faster variant of the nLB classifier (see the run-time analysis of the methods in Fig. 18 from "Appendix"), but with weaker predictive performance.

5.2 Run-time performance

In this section we examine the run-time performance of the different techniques from the three stages.[15] We start by comparing the average durations of each of the techniques over the datasets. For this, we only consider the datasets with fewer than 100,000 nodes since not all the methods are able to run on the larger datasets. For each of the relational classifiers the maximum and the Jaccard (except for cdRN) aggregation function have the longest durations (See Figs. 14, 15, 16 and 17 in the "Appendix"). The largest impact on run-time performance, however, is the use of the beta function. This function takes so long to run due to the hyperparameter tuning. This process can be done faster by reducing the grid search to only one or two levels or tuning the α and β parameters on a smaller sample of the training data, with limited performance decrease (see Table 8 and Fig. 13 in the "Appendix"). Also, only the parameters that give the required shape (such that the nodes with a smaller degree receive a higher weight as discussed in Sect. 5.1.1) could be taken into account to further speed up the grid search.

The learning of the weights for the nLB classifier can also be done on a smaller sample. Therefore, we also examine the time advantage of using this approach (see Fig. 18 in "Appendix"). In our experiments, even a sample of fewer than 100 instances was enough to tune the parameters of the nLB logistic regression. We consider this nLB classifier trained with only 100 instances as a third relational classifier, named nLB100 in the results. Although it performs slightly worse than the regular nLB classifier in terms of AUC, it can be trained much more quickly. However, when the class-label autocorrelation is uncertain

[15] All experiments are conducted on a 3.40 GHz Intel i7 CPU, with 8 GB RAM and a 64-bit operating system.

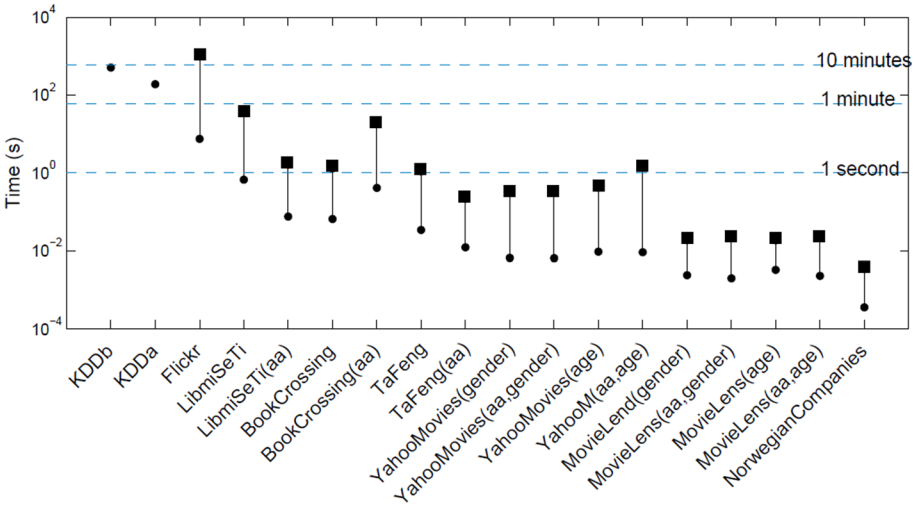


Fig. 8 Time improvement of the SW-transformation over wvRN and sum of shared nodes for different datasets. The square at the top of each bar represents the time needed for the wvRN classifier and circle at the bottom of each bar the time required for SW-transformation for the specific dataset. For the largest datasets, KDDa and KDDb, the wvRN classifier was not able to calculate a solution within the time allowed

Table 6 Top nodes with highest coefficient in the linear model of the SW-transformation in combination with the beta function

Rank	Yahoo movies (gender)	Yahoo movies (age)
1.	The Matrix Reloaded (2003)	Ocean's Eleven (2001)
2.	Terminator 3: Rise of the Machines (2003)	The Ring (2002)
3.	The Hulk (2003)	Scary Movie 3 (2003)
4.	X2: X-Men United (2003)	American Pie 2 (2001)
5.	Bad Boys II (2003)	American Pie (1999)
6.	The Lord of the Rings: The Two Towers (2002)	Pulp Fiction (1994)
7.	The Italian Job (2003)	The Texas Chainsaw Massacre (2003)
8.	The Matrix Revolutions (2003)	Austin Powers in Goldmember (2002)
9.	Bruce Almighty (2003)	Terminator 2—Judgment Day (1991)
10.	28 Days Later (2003)	Gladiator (2000)
11.	Kill Bill Vol. 1 (2003)	The Lizzie McGuire Movie (2003)
12.	American Wedding (2003)	Phone Booth (2003)
13.	Freddy vs. Jason (2003)	Uptown Girls (2003)
14.	S.W.A.T. (2003)	How to Deal (2003)
15.	The Matrix (1999)	Signs (2002)
16.	The League of Extraordinary Gentlemen (2003)	Daredevil (2003)
17.	The Lord of the Rings: The Fellowship of the Ring(2001)	X-Men (2000)
18.	Terminator 2—Judgment Day (1991)	The Matrix (1999)
19.	Seabiscuit (2003)	A Walk to Remember (2002)
20.	Star Wars (1977)	Anger Management (2003)

The higher scores indicate higher probability of being (a) male when predicting *gender* and (b) young when predicting *age* for the Yahoo movies bigraph

and the training time is an issue, it may be better to use the cdRN classifier, which is fast and whose performance is quite robust to different sorts of relational autocorrelation.

In terms of run-time performance, the SW-transformation outperforms all the other aggregation functions in combination with any non-tuning top-node function. It is able to scale to big datasets as it runs very fast. The average time needed for the regular sum of shared nodes and wvRN over the datasets is 65.4 s and the SW-transformation needs only 0.5478 s on average. This technique will be discussed in more detail in the next section.

5.3 SW-transformation

The SW-transformation combines the best relational classifier wvRN and one of the best performing aggregation functions, sum of shared nodes into a fast linear model that scales easily to big datasets (Fig. 8). It is the only technique in the study that scales well (or at all) to the biggest datasets KDDa with 8 million × 20 million nodes and KDDb with 19 million × 30 million nodes. An additional important aspect of the SW-transformation is the comprehensibility of the linear models it provides. A manual check of the top node coefficients (the impact they have to the target variable) can help to verify if the model makes sense. Comprehensibility is highly desirable, and even mandatory, in many domains where the decisions of the classifier must be clearly explained and validated before the classifier can be used (Gregor and Benbasat 1999; Martens et al. 2007; Martens and Provost 2014). In Table 6 we conduct an additional verification of the results, by examining the top nodes' coefficients using the combination of the beta and SW-transformation. We list the top 20 ranked instances with the highest coefficients when predicting gender and age for the Yahoo Movies bigraph. The rankings indeed appear intuitive and include (1) movies that are generally targeted to a male audience (*Terminator, X-man, Kill Bill*, etc.) and (2) movies usually preferred by younger people, such as *American Pie, Scary Movie, The Texas Chainsaw Massacre*, etc.

5.4 General recommendations

We have provided an extensive empirical study of the predictive and run-time performance for a number of choices in the framework over a large collection of bipartite datasets. The results indicate that it is difficult to simply claim that a certain combination of methods performs best across all domains. Instead, based on the empirical study, we would recommend experimenting with several choices as components for the three stages, those that generally provide good results: tangens hyperbolicum, cosine similarity, sum of shared nodes, wvRN and nLB100. For most datasets, they are among the fastest and most accurate combinations in our benchmark (Figs. 19, 20, 21, 22, 23, 24, 25 and 26 in "Appendix"), with less than 5% AUC difference from the combination that performs best. This is not the case for some very skewed MovieLens datasets (with only 1.25–6.5% positive labels), where we predict genres like Fantasy, Film-Noir, War or Mystery. In such cases, as discussed above, the supervised weighting function likelihood ratio or the tunable beta function might be more appropriate choices. Furthermore, our recommendations have weak results for the Reality Mining dataset, where most of the people have visited the same places (a person on average shares all the locations he/she visited with 50% of the other people). For such datasets, where projections are fully (or almost fully) connected, traditional classifiers (such as SVM) might be better alternatives.

6 Limitations and future research

This study presents a framework for node classification within bigraphs, aimed at utilizing the predictive power that comes from the relational structure of the bigraph. As such, it largely simplifies the problems under study in many domains, where additional information about the nodes and the edges might be available. The amount of information available from both sources, i.e. the network structure and the local information, varies greatly for each dataset, and so does the predictive power that comes from the two of them. The major advantage of the projection approach is to simplify the classification problem, which can render the solution much more efficient and easy to implement—and as the related work discussion revealed, has been a natural approach for practitioners. Casting the bipartite problem as a unipartite projection may be sufficient for certain settings; alternatively it may allow practitioners to get a first solution in place fast, and provide researchers with a solid baseline against which to compare new methods for classification on bigraphs (as wvRN did for within-network classification).

On the other hand, the simplification of the problem also presents the main limitation: as discussed at the outset, projecting the bigraph to a unimodal graph discards information. For example, the identities of particular top nodes—which are lost in the projection—can be useful for classification (Perlich and Provost 2006). However, particularly useful top nodes can be added back in as features of bottom nodes to create an attributed graph. Whether this approach would be successful in domains with millions (or more) of top nodes depends on whether there is a moderately small number of predictive top nodes. Perlich and Provost (2006) show that when learning from domains with a high-dimensional categorical attribute (its values essentially being top nodes), that in some domains creating features based on just the ten most discriminative values performs as well as more sophisticated methods, and in other domains it doesn't perform well at all.

More generally, in attributed graphs, any features of the bottom nodes could be included in the analysis in several ways. One could apply a traditional model on the structured information and use the scores as priors in the relational methods (Macskassy and Provost 2007). Alternatively, the scores from the relational classifiers could be used to complement the structured data in a traditional propositional model. The scores can be considered as additional features that capture information on the relations between the nodes. A third approach would be to include the attributes and the network links together in a full-blown relational model on the projected network. For example, in this paper we experimented with nLB (Macskassy and Provost 2007; Lu and Getoor 2003) on the (univariate) projection; instead, we could apply the full-blown Link-based classifier (Lu and Getoor 2003) on the projection with bottom-node attributes. Features of the top nodes could also be passed through via aggregation (Perlich and Provost 2003, 2006), as with top-node identities. However, we may want to consider bringing to bear more sophisticated statistical relational learning methods. A well-chosen projection-based approach still would be an important baseline against which to compare the more sophisticated approaches.

Another interesting extension to the framework might be to consider k-partite graphs, with many-to-many relations between several sets of nodes (e.g., persons, books, authors, genres). The framework could be generalized by adding an additional stage that aggregates the information from several node sets into the projection. The function could be static and consider an equal value for every set (e.g., an additive function that sums the similarity from every set) or parametrized. For the latter, the Dirichlet distribution could be used—the multivariate version of the beta distribution we use for the bipartite case.

In this study, we consider classification of unweighted bipartite graphs with binary labels. A direct extension of the framework would be generalizing to weighted bigraphs. In a weighted bigraph, the node links have a value associated with them, representing the strength or the capacity of the link. The weight of the link may or may not be relevant for the classification problem. Examples of weights that we can imagine to be relevant to associated tasks include the rating scores that a person gave to the movie or the book, the frequency of visits to a specific location, the amount purchased at the particular merchant, and so on. Weights can be included in the projection framework by taking them into account when calculating the similarity between pairs of bottom nodes, and then combining them with the top node scores for every distinct pair of bottom nodes in the bigraph. In the study we also assume a binary target variable (label). We do also consider multiclass datasets, where the node labels can belong to one of K classes—in our experimental setup, we cast these datasets to multiple bigraphs with binary labels. Alternatively, the classifiers can be used to calculate the probability score for each class and then determine the class with the maximal score. For the wvRN, the class scores automatically sum to one because of the normalizer Z. For the nLB classifier, the class vector contains the scores for each label. Similarly, cdRN can build several reference vectors for all classes with multiple entries; the probability scores can be calculated as the vector similarity between the class vector of a node and the reference vector for each class.

Finally, the three-step framework opens up the opportunity for adding new techniques and experimentation with mixing-and-matching component techniques to create new approaches to node classification in bigraph data. As mentioned in Sect. 2, papers on graph embedding techniques have reported good node classification performance as well as scalability and interpretability, including recent work applying embedding techniques directly to a bipartite graph. However, to the best of our knowledge, none of the existing research on heterogeneous graph embeddings has compared to a simpler (non-embedding) approach for node classification. We suggest that the very straightforward SW-transformation be employed as a benchmark for such studies.

An interesting avenue for future research might be to include graph embeddings as an additional or alternative step in the framework. Since homogeneous graph embeddings can be applied to unigraph networks, the relational classifier in our setting could be substituted by a combination of a graph embedding technique and a classifier built on the embedding. More recent work has also focused on bipartite network embeddings for the task of node classiffication. When implemented efficiently this might have an advantage over calculating the bigraph projection.

7 Conclusion

Bigraph datasets are an intuitive way to represent relational, behavioral and transactional data. Although prior studies individually have applied projection to bigraph data, this is the first systematic study of predictive modeling on bigraph data using projection. The modular three-stage projection framework we propose has the flexibility to compose many different classification methods—some previously used and most novel. Techniques composed using the framework were empirically evaluated in terms of predictive and run-time performance. The comparison with a traditional classifier shows encouraging results: the linear SVM has only average performance when compared to

the collection of projection-based methods (even linear ones), and the popular implementation does not scale to the largest datasets.

When composing techniques, in our experiments, among the top-node functions, the tangens hyperbolicum performs best. For the latter stages, the cosine aggregation function, followed by the sum of the shared nodes in combination with the wvRN relational classifier, give the best results. We combine the latter two as the basis of a new technique (the SW-transformation). It is a very fast, linear method that is able to scale to datasets of millions of nodes easily, while providing a comprehensible model.

The purpose of this paper is to study classification on bigraph data using projection systematically. We do not claim to have found the best combination of elements; the framework opens the design space. Follow-up work could suggest an even better alternative for one of the stages. Nonetheless, based on the results, we would recommend that researchers and practitioners faced with classification on bigraph data seriously consider the SW-transformation, due to its speed, solid predictive performance, and comprehensibility. At the very least, the SW-transform provides a very solid baseline method for future studies of methods for predictive modeling with (sparse) bigraph data.

Appendix

A Data distributions

See Figs. 9, 10, 11 and 12.

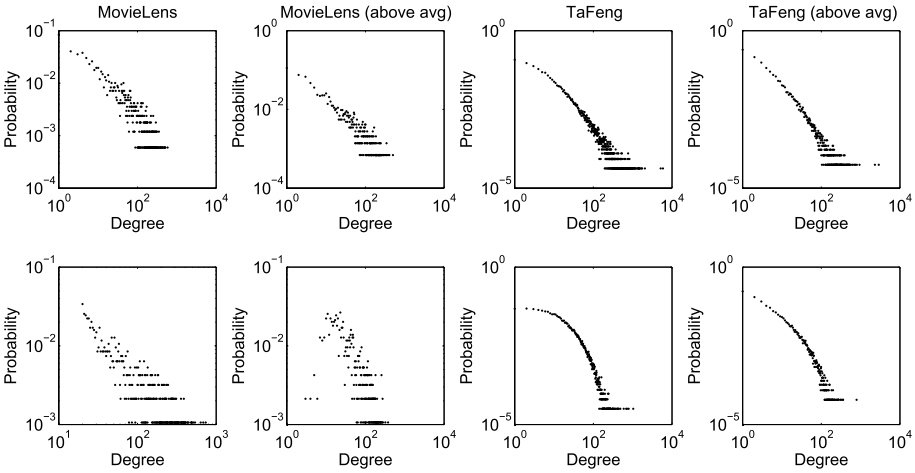


Fig. 9 Degree distributions of the top nodes (upper row) and bottom nodes (bottom row) for different datasets

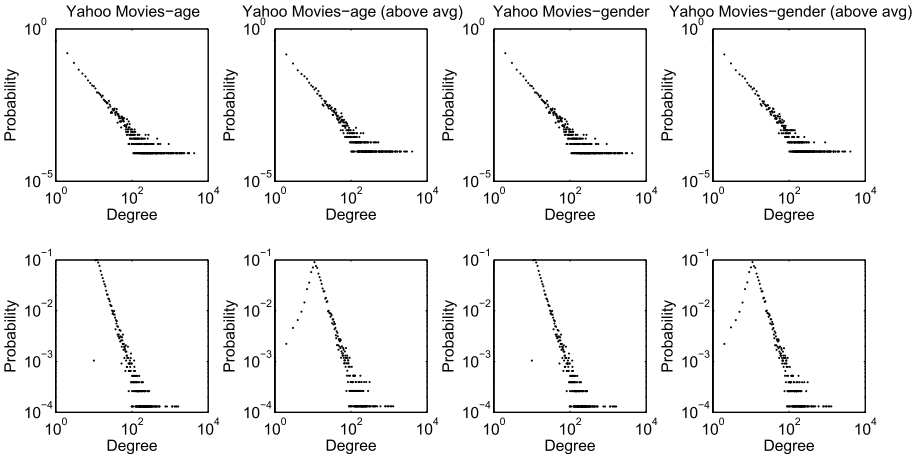


Fig. 10 Degree distributions of the top nodes (upper row) and bottom nodes (bottom row) for different datasets

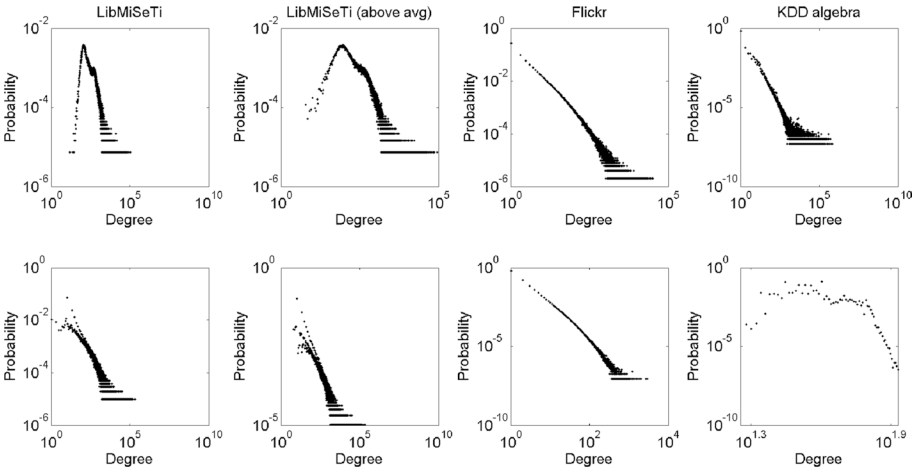


Fig. 11 Degree distributions of the top nodes (upper row) and bottom nodes (bottom row) for different datasets

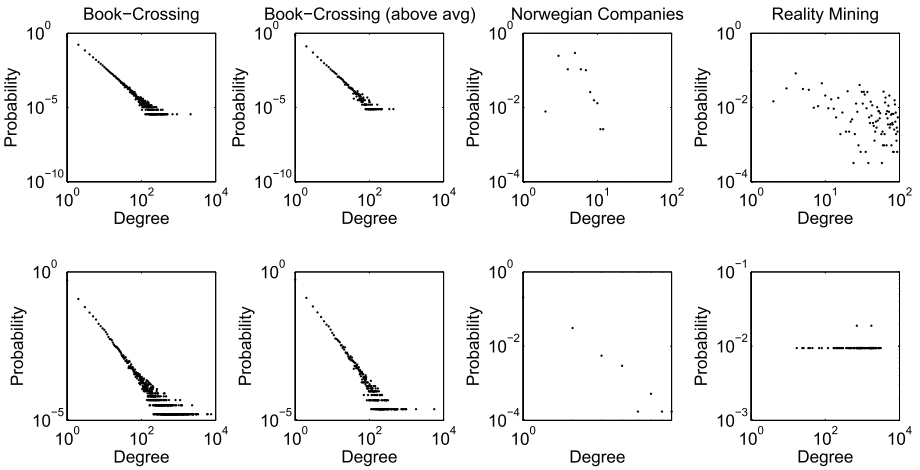


Fig. 12 Degree distributions of the top nodes (upper row) and bottom nodes (bottom row) for different datasets

B Results tables

See Figs. 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and Tables 7, 8, 9.

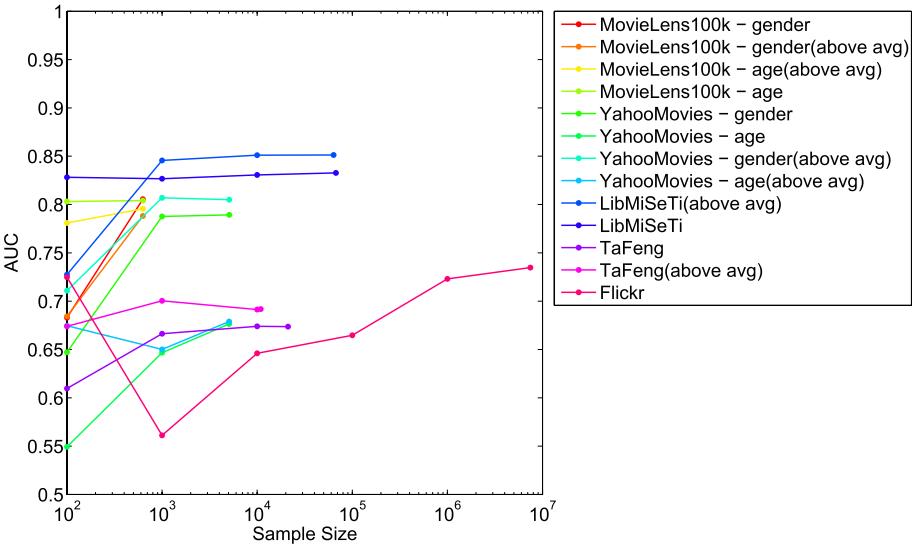


Fig. 13 Predictive performance of the beta function in combination with SW-transformation when the parameters are tuned on a sample of the training data and trained on the full training data. The difference in predictive performance is limited

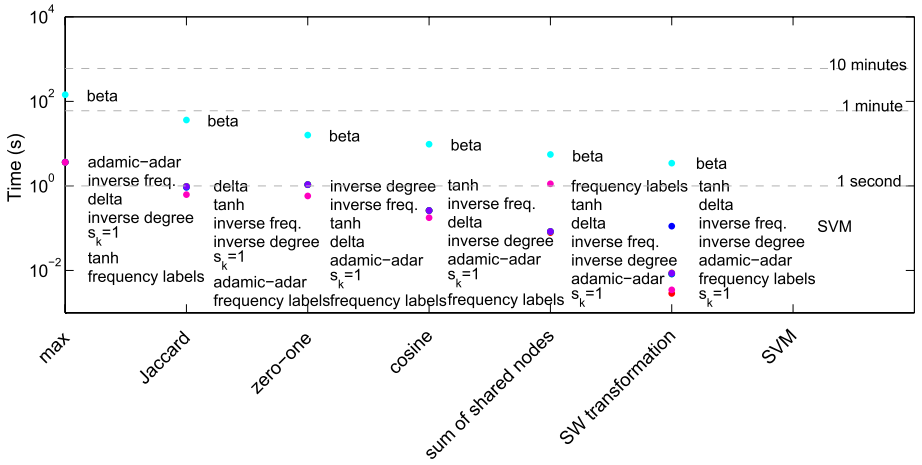


Fig. 14 Aggregated run-time results for each of the top node and aggregation functions with wvRN (including the SW-transformation). Since most of the top-node functions (except for the beta) have similar durations, the markers on the plots are very close to each other (and given in descending order). The SW-transformation outperforms all the other aggregation functions in combination with any non-tuning top-node function

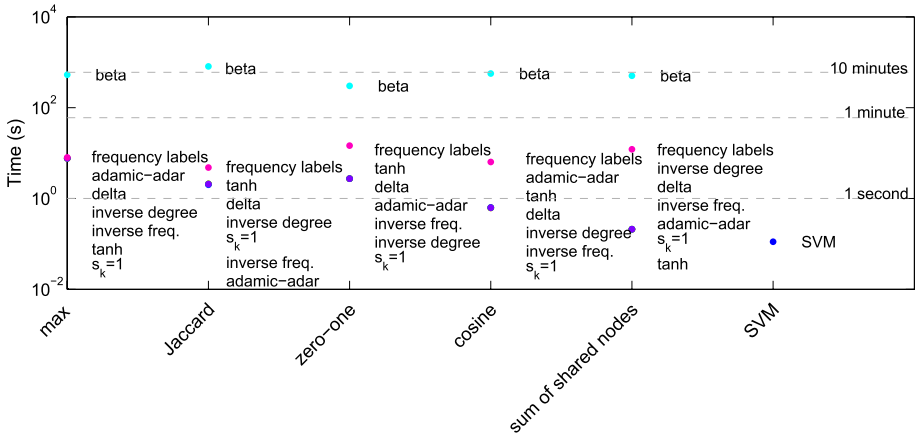


Fig. 15 Aggregated run-time results for each of the top node and aggregation functions with the nLB classifier

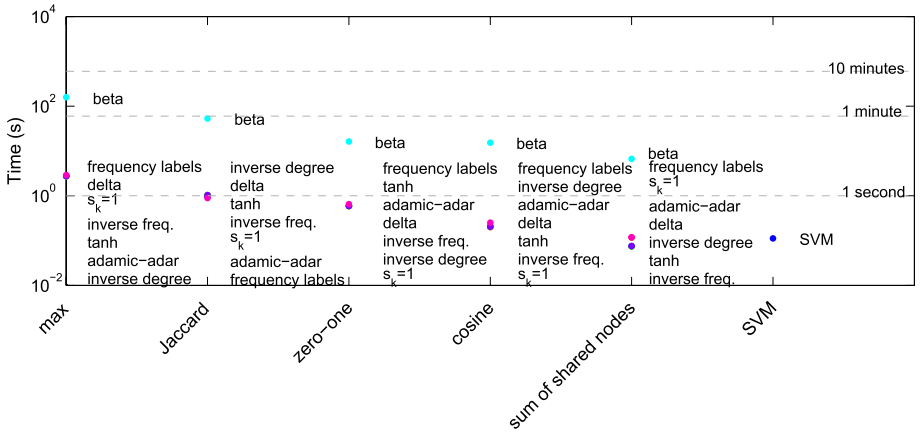


Fig. 16 Aggregated run-time results for each of the top node and aggregation functions with the nLB100 classifier (nLB with 100 training instances)

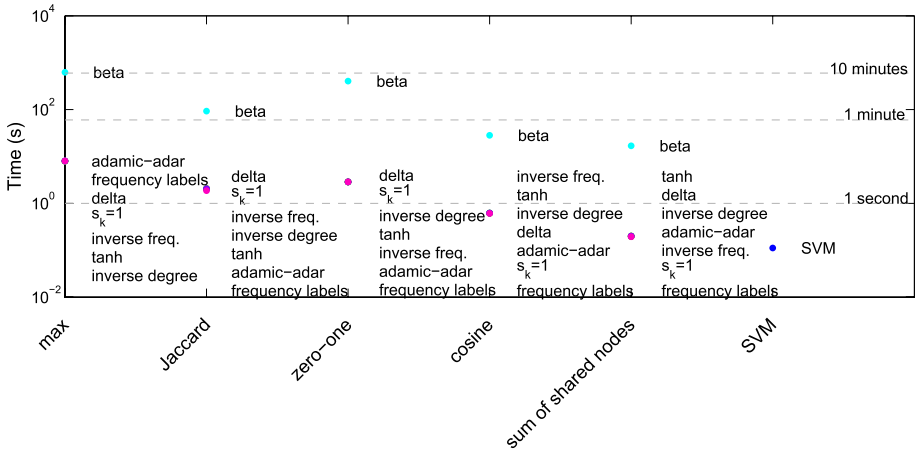


Fig. 17 Aggregated run-time results for each of the top node and aggregation functions with the cdRN classifier

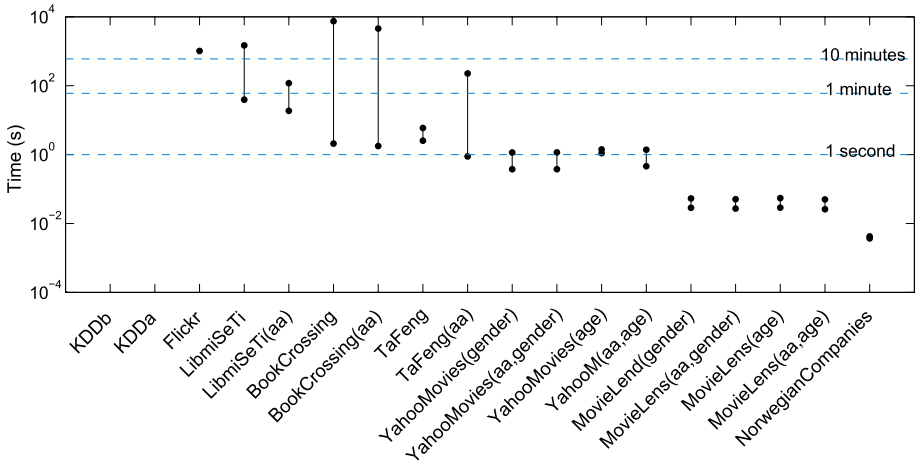


Fig. 18 Time improvement of nLB with sampling over 100 instances as compared to no sampling for different datasets. The top of each bar represents the time needed for the nLB classifier and the bottom of each bar the time required to train the nLB with 100 instances for the specific dataset

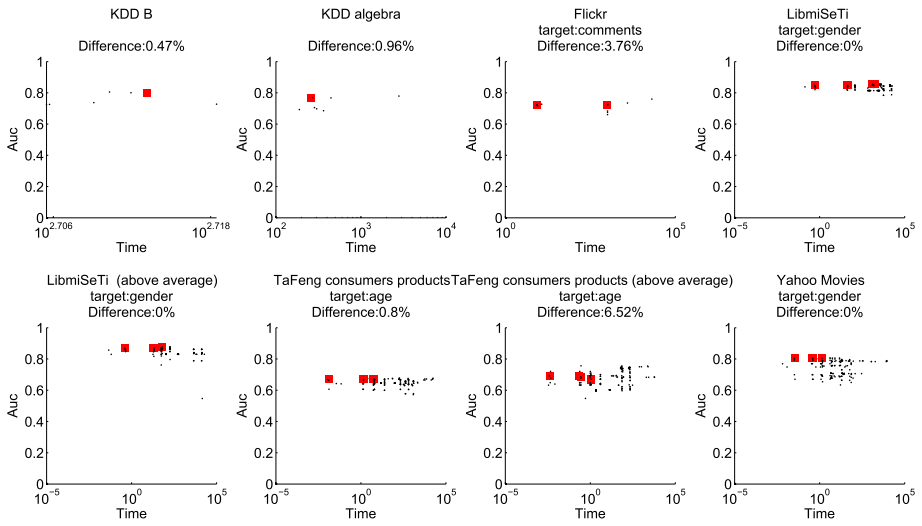


Fig. 19 Ranking of all combinations of methods, with the proposed combinations highlighted in red (Color figure online)

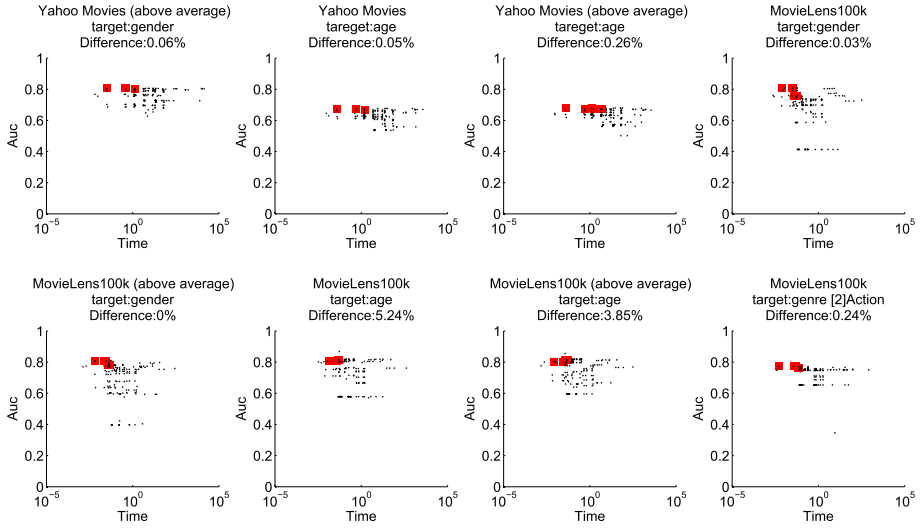


Fig. 20 Ranking of all combinations of methods, with the proposed combinations highlighted in red (Color figure online)

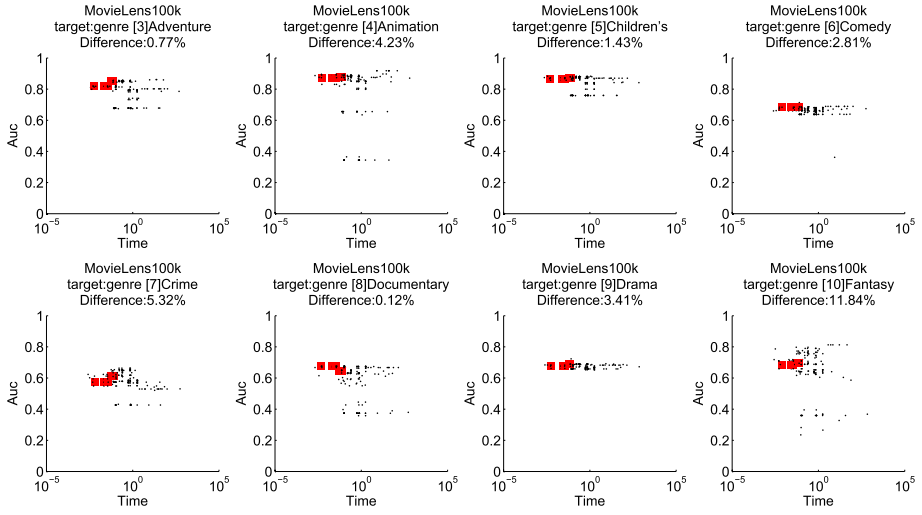


Fig. 21 Ranking of all combinations of methods, with the proposed combinations highlighted in red (Color figure online)

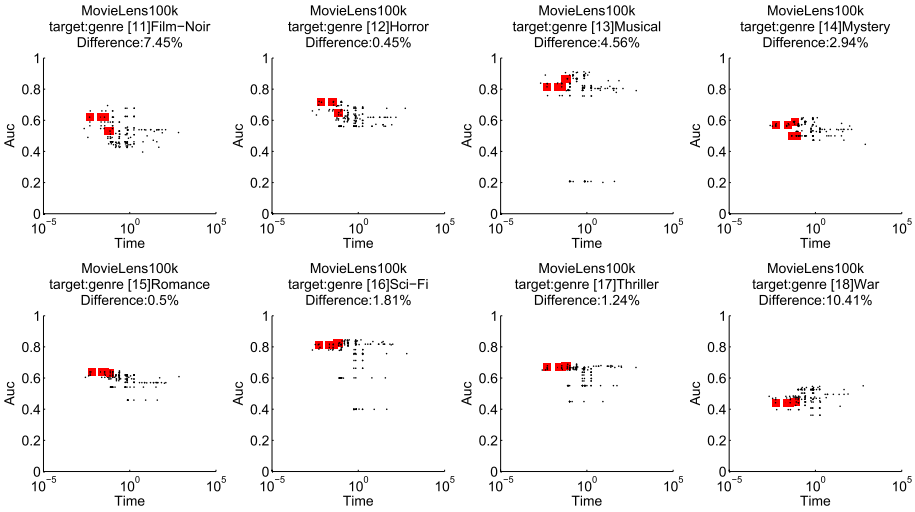


Fig. 22 Ranking of all combinations of methods, with the proposed combinations highlighted in red (Color figure online)

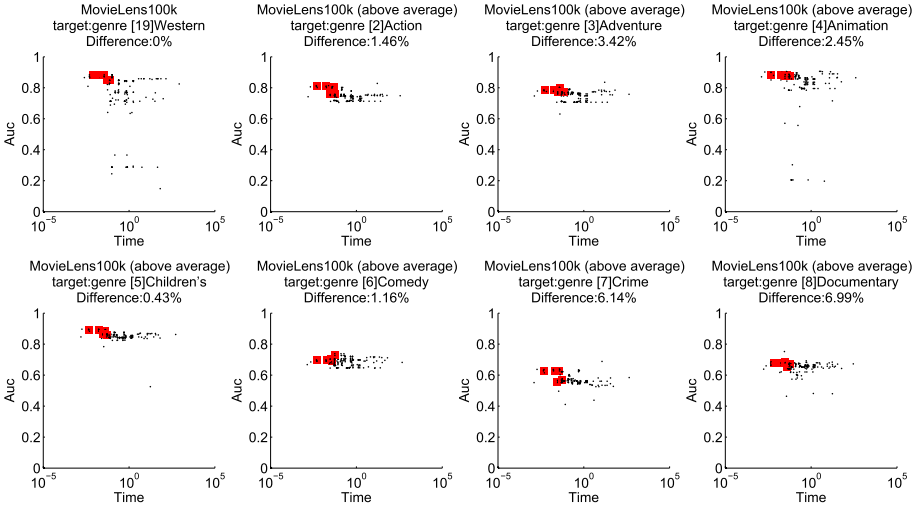


Fig. 23 Ranking of all combinations of methods, with the proposed combinations highlighted in red (Color figure online)

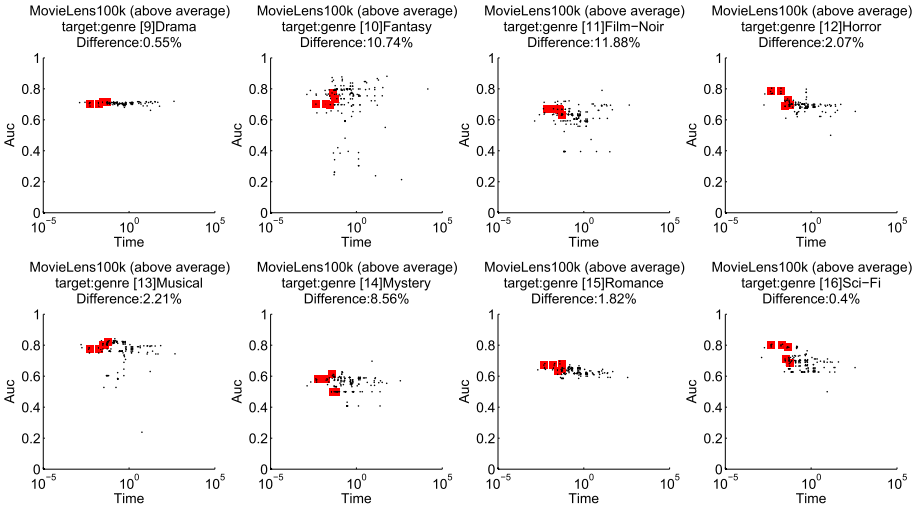


Fig. 24 Ranking of all combinations of methods, with the proposed combinations highlighted in red (Color figure online)

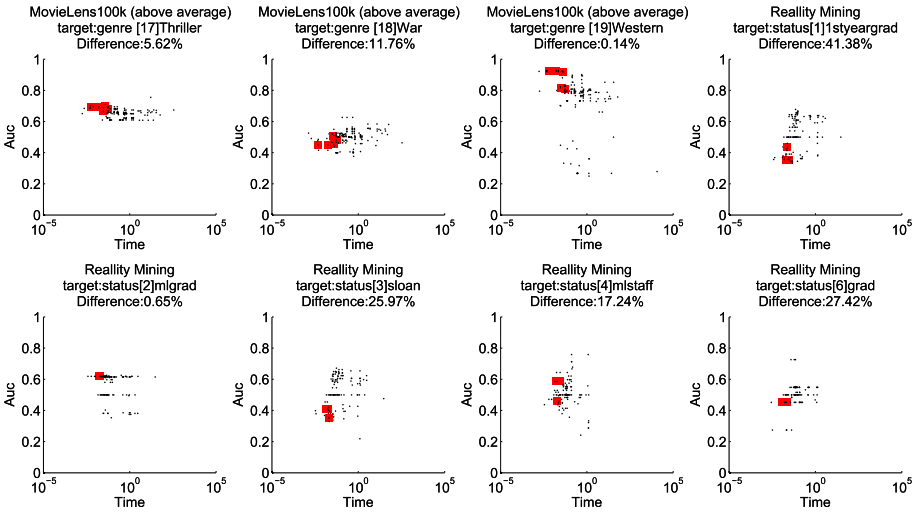


Fig. 25 Ranking of all combinations of methods, with the proposed combinations highlighted in red (Color figure online)

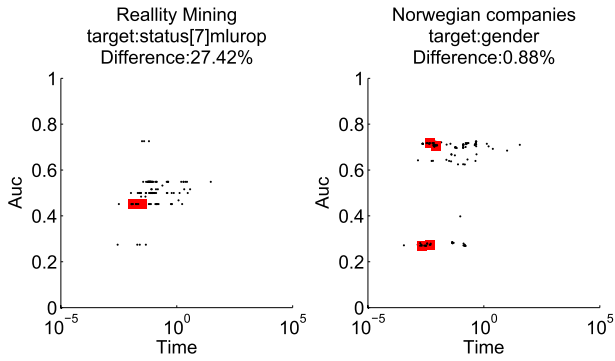


Fig. 26 Ranking of all combinations of methods, with the proposed combinations highlighted in red (Color figure online)

Table 7 Kemeny–Young ranking for all the combinations of techniques

Top node weight	Aggregation function	Classifier	Best rank	Worst rank	Avgerage rank
tanh	Cosine function	wvRN	**3.5**	**142.0**	**40.9**
Inverse degree	Cosine function	wvRN	**3.5**	**140.5**	**41.0**
tanh	Cosine function	cdRN	**3.0**	**120.0**	**47.3**
tanh	Sum of shared nodes	wvRN	**2.0**	**148.0**	**46.1**
Beta distribution	Sum of shared nodes	wvRN	**1.0**	**158.0**	**56.4**
tanh	Cosine function	nlb	**8.5**	**121.5**	**45.9**
Inverse degree	Cosine function	nlb	*4.5*	*124.0*	*46.0*
Inverse degree	Sum of shared nodes	wvRN	**2.0**	**143.0**	**46.4**
Inverse frequency	Cosine function	wvRN	*2.0*	*140.0*	*39.3*
tanh	Jaccard	wvRN	5.5	155.5	50.6
tanh	Cosine function	nlb 100	*8.5*	*134.0*	*52.3*
Inverse degree	Cosine function	nlb 100	4.5	134.0	52.2
Inverse degree	Cosine function	cdRN	**6.0**	**123.0**	**48.2**
tanh	Sum of shared nodes	nlb	**2.0**	**134.5**	**49.0**
Inverse degree	Sum of shared nodes	nlb	**2.0**	**134.5**	**49.3**
tanh	Sum of shared nodes	nlb 100	**2.0**	**134.5**	**54.7**
tanh	Sum of shared nodes	cdRN	**1.5**	**130.5**	**50.8**
Inverse degree	Sum of shared nodes	nlb 100	*2.0*	*134.5*	*55.2*
Inverse degree	Sum of shared nodes	cdRN	**1.5**	**130.5**	**51.4**
Inverse frequency	Sum of shared nodes	wvRN	**2.0**	**149.0**	**43.7**
Beta distribution	Cosine function	cdRN	**1.0**	**158.0**	**57.1**
Inverse frequency	Sum of shared nodes	nlb	*8.0*	*111.0*	*47.7*
Inverse frequency	Cosine function	cdRN	**10.0**	**102.0**	**43.7**
Inverse frequency	Cosine function	nlb	*7.5*	*104.0*	*43.2*
Inverse frequency	Jaccard	wvRN	*2.0*	*154.0*	*47.5*
Inverse degree	Jaccard	wvRN	5.5	155.5	52.1
Inverse frequency	Sum of shared nodes	nlb 100	*8.0*	*134.0*	*51.3*
Inverse frequency	Sum of shared nodes	cdRN	*9.0*	*159.0*	*51.9*
Inverse frequency	Cosine function	nlb 100	7.5	134.0	49.1
Beta distribution	Sum of shared nodes	cdRN	*3.0*	*161.0*	*65.0*
Inverse frequency	Jaccard	cdRN	**2.0**	**144.0**	**57.4**
tanh	Jaccard	nlb	11.0	139.5	62.5
tanh	Jaccard	cdRN	*2.0*	*142.5*	*61.9*
Inverse degree	Jaccard	cdRN	*12.0*	*142.5*	*64.0*
w = 1	Sum of shared nodes	wvRN	*1.0*	*144.0*	*51.8*
Beta distribution	Jaccard	wvRN	4.0	161.5	71.4
Beta distribution	Cosine function	wvRN	**4.0**	**156.0**	**58.2**
Beta distribution	Max	wvRN	*3.5*	*151.5*	*65.7*
Beta distribution	Sum of shared nodes	nlb	**4.0**	**160.5**	**72.7**
Beta distribution	Jaccard	nlb	**4.0**	**156.0**	**55.8**
Beta distribution	Cosine function	nlb	**4.0**	**156.0**	**58.1**
Beta distribution	Cosine function	nlb 100	2.0	161.5	68.2
Inverse frequency	Jaccard	nlb	4.0	145.5	58.2
Beta distribution	Max	nlb	*3.5*	*151.5*	*65.6*
w = 1	Cosine function	wvRN	2.0	141.0	52.6
Adamic and Adar	Cosine function	wvRN	3.0	135.0	51.6

Table 7 (continued)

Top node weight	Aggregation function	Classifier	Best rank	Worst rank	Avgerage rank
w = 1	Cosine function	cdRN	*7.0*	*105.0*	*56.5*
Adamic and Adar	Sum of shared nodes	wvRN	*1.0*	*138.0*	*53.0*
w = 1	Jaccard	wvRN	2.0	158.0	54.9
Adamic and Adar	Cosine function	cdRN	*10.0*	*107.0*	*58.2*
Inverse degree	Jaccard	nlb	10.5	142.5	64.2
Delta	Cosine function	wvRN	1.0	145.0	57.8
tanh	Jaccard	nlb 100	5.5	157.5	76.7
Inverse degree	Jaccard	nlb 100	5.5	157.5	77.6
Beta distribution	Sum of shared nodes	nlb 100	4.0	161.0	76.0
tanh	Max	wvRN	5.5	159.0	73.3
Inverse degree	Max	wvRN	5.5	150.5	73.3
Inverse frequency	Jaccard	nlb 100	4.0	158.0	72.8
w = 1	Sum of shared nodes	nlb	14.5	103.5	56.8
Adamic and Adar	Sum of shared nodes	nlb	15.0	100.5	57.8
w = 1	Cosine function	nlb	9.0	103.0	56.1
w = 1	Sum of shared nodes	nlb 100	14.5	134.0	59.8
w = 1	Sum of shared nodes	cdRN	13.0	160.0	62.4
Adamic and Adar	Cosine function	nlb	8.5	97.0	56.1
w = 1	Cosine function	nlb 100	9.0	157.5	64.8
Adamic and Adar	Cosine function	nlb 100	8.5	157.5	64.6
Beta distribution	Jaccard	nlb 100	1.0	162.0	87.3
w = 1	Jaccard	cdRN	*4.0*	*156.0*	*65.2*
Adamic and Adar	Jaccard	wvRN	3.0	153.0	58.1
Delta	Cosine function	nlb	1.5	139.0	64.6
w = 1	Jaccard	nlb	2.0	158.0	65.4
Adamic and Adar	Sum of shared nodes	cdRN	13.0	158.0	61.5
Adamic and Adar	Sum of shared nodes	nlb 100	15.0	134.0	61.8
Delta	Cosine function	nlb 100	1.5	157.5	73.6
Delta	Cosine function	cdRN	1.0	141.0	65.8
Beta distribution	Jaccard	cdRN	1.0	159.0	75.0
Beta distribution	Max	cdRN	1.0	162.0	84.9
Likelihood ratio	Cosine function	wvRN	3.5	146.5	68.1
w = 1	Jaccard	nlb 100	2.0	159.0	80.7
Likelihood ratio	Cosine function	nlb	3.5	146.5	69.8
Likelihood ratio	Cosine function	nlb 100	3.5	157.5	77.1
Likelihood ratio	Cosine function	cdRN	3.5	152.0	71.3
Adamic and Adar	Jaccard	nlb	6.0	150.0	68.8
Adamic and Adar	Jaccard	cdRN	*8.0*	*152.0*	*69.6*
Delta	Sum of shared nodes	wvRN	1.0	158.0	62.3
tanh	Max	cdRN	13.5	156.0	84.3
Likelihood ratio	Sum of shared nodes	wvRN	3.5	139.0	75.8
Likelihood ratio	Sum of shared nodes	nlb	21.5	139.0	77.5
Likelihood ratio	Sum of shared nodes	nlb 100	21.5	139.0	78.9
Likelihood ratio	Jaccard	wvRN	1.5	158.0	78.6
Likelihood ratio	Jaccard	nlb	2.0	158.0	79.6
Likelihood ratio	Jaccard	cdRN	*1.5*	*154.0*	*78.0*

Table 7 (continued)

Top node weight	Aggregation function	Classifier	Best rank	Worst rank	Avgerage rank
Delta	Sum of shared nodes	nlb	1.5	153.0	72.1
tanh	Max	nlb	2.5	145.5	83.8
Inverse degree	Max	nlb	4.0	145.5	83.9
Delta	Sum of shared nodes	cdRN	1.0	154.0	72.8
tanh	Max	nlb 100	15.5	145.5	88.9
SVM			1.0	162.0	91.3
Inverse degree	Max	nlb 100	11.0	145.5	88.6
Inverse degree	Max	cdRN	13.5	157.0	84.2
Inverse frequency	Max	wvRN	7.0	160.0	81.9
Delta	Sum of shared nodes	nlb 100	1.5	161.5	83.2
Delta	Max	wvRN	2.0	161.0	80.0
Likelihood ratio	Sum of shared nodes	cdRN	3.5	162.0	85.5
Inverse frequency	Max	nlb	2.5	135.5	87.3
Inverse frequency	Max	nlb 100	3.0	135.5	88.0
Inverse frequency	Max	cdRN	2.0	137.0	86.1
Adamic and Adar	Jaccard	nlb 100	6.0	157.5	83.6
Delta	Jaccard	wvRN	1.0	159.0	77.8
Likelihood ratio	Jaccard	nlb 100	2.5	162.0	92.0
Delta	Max	cdRN	2.0	158.0	92.9
Delta	Max	nlb	4.0	160.0	92.3
Delta	Jaccard	nlb	16.5	155.5	88.9
Delta	Max	nlb 100	4.0	162.0	97.1
Delta	Jaccard	cdRN	18.0	154.0	89.3
Beta distribution	Max	nlb 100	24.0	162.0	103.1
Adamic and Adar	Max	wvRN	11.0	148.0	100.7
Delta	Jaccard	nlb 100	16.5	162.0	106.8
Adamic and Adar	Max	cdRN	13.0	162.0	103.7
Adamic and Adar	Max	nlb	16.0	131.5	100.8
Adamic and Adar	Max	nlb 100	12.0	161.0	105.6
Likelihood ratio	Max	wvRN	32.5	148.5	109.6
Likelihood ratio	Max	nlb	32.5	148.5	111.7
Likelihood ratio	Max	nlb 100	42.0	161.0	113.1
Likelihood ratio	Max	cdRN	66.5	160.0	117.3
Any	Zero–one	wvRN	21.0	157.0	119.7
w = 1	Max	wvRN	21.0	157.0	119.7
Any	Zero–one	nlb	9.0	157.0	118.2
w = 1	Max	nlb	9.0	157.0	118.2
] Any	Zero–one	cdRN	22.0	158.0	124.9
w = 1	Max	cdRN	22.0	158.0	124.9
Any	Zero–one	nlb 100	28.0	158.0	127.7
w = 1	Max	nlb 100	28.0	158.0	127.7

We emphasize the combinations that are not significantly worse than the best method (underlined) at a 5% significance level in bold and the combinations that are significantly worse at 5% but not at 1% significance level in italics. The other methods that are significantly worse at 1% significance level are shown in regular font

Table 8 Beta grid search on three levels with the optimal α and β parameters, as well as the corresponding AUC per level

Dataset	Level 1			Level 2			Level 3		
	Alpha	Beta	AUC	Alpha	Beta	AUC	Alpha	Beta	AUC
MovieLens gender	3.1000	12.1000	0.7019	3.1000	15.1000	0.7099	2.7667	16.1000	0.7110
MovieLens gender (above average)	0.1000	6.1000	0.7593	1.1000	9.1000	0.7622	0.4333	8.7667	0.7690
MovieLens age	0.1000	3.1000	0.8087	0.1000	1.1000	0.8106	0.1000	1.4333	0.8110
MovieLens age (above average)	0.1000	3.1000	0.8193	1.1000	6.1000	0.8231	1.1000	7.1000	0.8242
Yahoo Movies (gender)	0.1000	3.1000	0.7985	0.1000	1.1000	0.8046	0.4333	1.4333	0.8060
Yahoo Movies above average (gender)	0.1000	3.1000	0.7955	0.1000	1.1000	0.8026	0.1000	1.1000	0.8026
Yahoo Movies (age)	0.1000	3.1000	0.6637	0.1000	3.1000	0.6637	0.4333	3.7667	0.6698
Yahoo Movies above average (age)	0.1000	3.1000	0.6577	0.1000	1.1000	0.6594	0.1000	1.1000	0.6594
TaFeng	0.1000	3.1000	0.6861	0.1000	1.1000	0.6894	0.4333	2.1000	0.6969
TaFeng (above average)	0.1000	3.1000	0.7198	0.1000	2.1000	0.7199	0.1000	2.4333	0.7199
BookCrossing	0.1000	0.1000	0.5892	0.1000	0.1000	0.5892	0.4333	0.4333	0.5913
BookCrossing (above average)	0.1000	0.1000	0.5716	1.1000	3.1000	0.5732	0.7667	3.4333	0.5738
LibimSeTi	0.1000	3.1000	0.8461	0.1000	1.1000	0.8483	0.4333	1.7667	0.8487
LibimSeTi (above average)	0.1000	3.1000	0.8669	0.1000	1.1000	0.8676	0.1000	1.4333	0.8676
Flickr	6.1000	0.1000	0.7337	6.1000	0.1000	0.7337	5.7667	0.1000	0.7341
KDDa	0.1000	12.1000	0.7888	0.1000	15.1000	0.7892	0.1000	16.1000	0.7791

The aggregation function used is the sum of shared nodes in combination with the wvRN relational classifier. The typical shapes of the function correspond strongly to the intuition that top nodes with smaller degree are more discriminative and therefore should have higher weights, except for the Flickr dataset

Table 9 Best combinations of methods per dataset

Dataset	Target	Top nodes function	Aggregation function	Relational classifier	AUC
KDD B		Delta	Sum of shared nodes	wvRN	0.8054
KDD algebra		Beta distribution	Sum of shared nodes	wvRN	0.7791
Flickr	Target:comments	SVM			0.7602
LibmiSeTi	Target:gender	tanh	Cosine function	wvRN	0.8562
LibmiSeTi (above average)	Target:gender	tanh	Cosine function	wvRN	0.8762
TaFeng consumers products	Target:age	Beta distribution	Sum of shared nodes	nlb	0.6785
TaFeng consumers products (above average)	Target:age	Delta	Sum of shared nodes	nlb 100	0.7564
Yahoo movies	Target:gender	tanh	Sum of shared nodes	wvRN	0.8071
Yahoo movies (above average)	Target:gender	tanh	Sum of shared nodes	nlb	0.8070
Yahoo movies	Tareget:age	Beta distribution	Sum of shared nodes	cdRN	0.6763
Yahoo movies (above average)	Tareget:age	Beta distribution	Cosine function	cdRN	0.6795
MovieLens100k	Target:gender	Inverse degree	Sum of shared nodes	wvRN	0.8071
MovieLens100k (above average)	Target:gender	tanh	Sum of shared nodes	wvRN	0.8104
MovieLens100k	Target:age	SVM			0.8685
MovieLens100k (above average)	Target:age	SVM			0.8543
MovieLens100k	Target:genre [2] Action	Delta	Sum of shared nodes	cdRN	0.7743
MovieLens100k	Target:genre [3] Adventure	Beta distribution	Cosine function	cdRN	0.8615
MovieLens100k	Target:genre [4] Animation	Beta distribution	Jaccard	wvRN	0.9180
MovieLens100k	Target:genre [5] Children's	Likelihood ratio	Jaccard	wvRN	0.8835
MovieLens100k	Target:genre [6] Comedy	SVM			0.7135
MovieLens100k	Target:genre [7] Crime	w = 1	Jaccard	wvRN	0.6632
MovieLens100k	Target:genre [8] Documentary	Delta	Sum of shared nodes	nlb	0.6775
MovieLens100k	Target:genre [9] Drama	SVM			0.7232
MovieLens100k	Target:genre [10] Fantasy	Beta distribution	Sum of shared nodes	wvRN	0.8131
MovieLens100k	Target:genre [11] Film-Noir	SVM			0.6948
MovieLens100k	Target:genre [12] Horror	Delta	Sum of shared nodes	cdRN	0.7207
MovieLens100k	Target:genre [13] Musical	Likelihood ratio	Jaccard	wvRN	0.9118

Table 9 (continued)

Dataset	Target	Top nodes function	Aggregation function	Relational classifier	AUC
MovieLens100k	Target:genre [14] Mystery	Likelihood ratio	Jaccard	wvRN	0.6166
MovieLens100k	Target:genre [15] Romance	Delta	Cosine function	cdRN	0.6443
MovieLens100k	Target:genre [16] Sci-Fi	Likelihood ratio	Jaccard	wvRN	0.8451
MovieLens100k	Target:genre [17] Thriller	Delta	Cosine function	cdRN	0.6883
MovieLens100k	Target:genre [18] War	Beta distribution	Max	cdRN	0.5502
MovieLens100k	Target:genre [19] Western	tanh	Sum of shared nodes	cdRN	0.8836
MovieLens100k (above average)	Target:genre [2] Action	Beta distribution	Jaccard	nlb 100	0.8283
MovieLens100k (above average)	Target:genre [3] Adventure	Beta distribution	Jaccard	nlb 100	0.8358
MovieLens100k (above average)	Target:genre [4] Animation	Beta distribution	Sum of shared nodes	wvRN	0.9061
MovieLens100k (above average)	Target:genre [5] Children's	w = 1	Sum of shared nodes	wvRN	0.8965
MovieLens100k (above average)	Target:genre [6] Comedy	Delta	Cosine function	nlb	0.7386
MovieLens100k (above average)	Target:genre [7] Crime	Beta distribution	Jaccard	nlb 100	0.6885
MovieLens100k (above average)	Target:genre [8] Documentary	SVM			0.7523
MovieLens100k (above average)	Target:genre [9] Drama	Beta distribution	Max	cdRN	0.7194
MovieLens100k (above average)	Target:genre [10] Fantasy	Beta distribution	Jaccard	cdRN	0.8810
MovieLens100k (above average)	Target:genre [11] Film-Noir	Beta distribution	Jaccard	nlb 100	0.7901
MovieLens100k (above average)	Target:genre [12] Horror	Delta	Sum of shared nodes	wvRN	0.8038
MovieLens100k (above average)	Target:genre [13] Musical	Delta	Jaccard	wvRN	0.8415
MovieLens100k (above average)	Target:genre [14] Mystery	Beta distribution	Jaccard	nlb 100	0.6971
MovieLens100k (above average)	Target:genre [15] Romance	Delta	Cosine function	wvRN	0.6972
MovieLens100k (above average)	Target:genre [16] Sci-Fi	Delta	Sum of shared nodes	wvRN	0.8063
MovieLens100k (above average)	Target:genre [17] Thriller	Beta distribution	Jaccard	nlb 100	0.7558
MovieLens100k (above average)	Target:genre [18] War	Likelihood ratio	Jaccard	wvRN	0.6477
MovieLens100k (above average)	Target:genre [19] Western	Adamic and Adar	Sum of shared nodes	wvRN	0.9275
Reallity mining	Target:status [1] 1styeargrad	Beta distribution	Jaccard	nlb	0.8505

Table 9 (continued)

Dataset	Target	Top nodes function	Aggregation function	Relational classifier	AUC
Reallity mining	Target:status [2] mlgrad	Likelihood ratio	Max	wvRN	0.6255
Reallity mining	Target:status [3] Sloan	Delta	Cosine function	cdRN	0.6710
Reallity mining	Target:status [4] mlstaff	Delta	Max	wvRN	0.7586
Reallity mining	Target:status [6] Grad	Likelihood ratio	Sum of shared nodes	cdRN	0.7258
Reallity mining	Target:status [7] mlurop	Likelihood ratio	Sum of shared nodes	cdRN	0.7258
Norwegian companies	Target:gender	tanh	Max	nlb	0.7745

References

Adamic, L. A., & Adar, E. (2003). Friends and neighbors on the web. *Social Networks*, *25*(3), 211–230.

Allali, O., Magnien, C., & Latapy, M. (2011). Link prediction in bipartite graphs using internal links and weighted projection. In *Conference on computer communications workshops (INFOCOM WKSHPS)* (pp. 936–941). IEEE.

Barber, M. J. (2007). Modularity and community detection in bipartite networks. *Physical Review E*, *76*(6), 066102.

Benchettara, N., Kanawati, R., & Rouveirol, C. (2010). Supervised machine learning applied to link prediction in bipartite social networks. In *Advances in social networks analysis and mining (ASONAM)* (pp. 326–330). IEEE.

Borgatti, S. P., & Everett, M. G. (1997). Network analysis of 2-mode data. *Social Networks*, *19*(3), 243–269.

Borgatti, S. P., & Halgin, D. S. (2011). Analyzing affiliation networks. In J. Scott & P. J. Carrington (Eds.), *The Sage handbook of social network analysis* (pp. 417–433). Thousand Oaks: SAGE Publications.

Brozovsky, L. & Petricek, V. (2007). Recommender system for online dating service. In *Proceedings of conference znalosti 2007*. Ostrava: VSB.

Cancho, R. F. I., & Solé, R. V. (2001). The small world of human language. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, *268*(1482), 2261–2265.

Chen, X., Yu, G., Wang, J., Domeniconi, C., Li, Z., & Zhang, X. (2019). Activehne: Active heterogeneous network embedding. arXiv preprint arXiv:1905.05659.

Cho, E., Myers, S. A., & Leskovec, J. (2011). Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1082–1090). ACM.

Conitzer, V., Davenport, A., & Kalagnanam, J. (2006). Improved bounds for computing Kemeny rankings. In *AAAI* (Vol. 6, pp. 620–626).

Cui, P., Wang, X., Pei, J., & Zhu, W. (2018). A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, *31*(5), 833–852.

de Cnudde, S., Martens, D., Evgeniou, T., & Provost, F. (2017). A benchmarking study of classification techniques for behavioral data. Working papers, University of Antwerp, Faculty of Applied Economics.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30.

Dong, Y., Chawla, N. V., & Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 135–144). ACM.

Doreian, P., Batagelj, V., & Ferligoj, A. (2004). Generalized blockmodeling of two-mode network data. *Social Networks*, *26*(1), 29–53.

Du, W., Yu, S., Yang, M., Qu, Q., & Zhu, J. (2018). GPSP: Graph partition and space projection based approach for heterogeneous network embedding. In *Companion proceedings of the the web conference* (Vol. 2018, pp. 59–60).

Eagle, N., & Pentland, A. (2006). Reality mining: Sensing complex social systems. *Personal and Ubiquitous Computing*, *10*(4), 255–268.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, *9*, 1871–1874.

Faust, K. (1997). Centrality in affiliation networks. *Social Networks*, *19*(2), 157–191.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874.

Forbes, C., Evans, M., Hastings, N., & Peacock, B. (2011). *Statistical distributions*. New York: Wiley.

Gallagher, B., Tong, H., Eliassi-Rad, T., & Faloutsos, C. (2008). Using ghost edges for classification in sparsely labeled networks. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 256–264). ACM.

Gao, M., Chen, L., He, X., & Zhou, A. (2018). Bine: Bipartite network embedding. In *The 41st international ACM SIGIR conference on research and development in information retrieval* (pp. 715–724). ACM.

Getoor, L., & Taskar, B. (2007). *Introduction to statistical relational learning* (Vol. 1). Cambridge: MIT Press.

Goel, S., Hofman, J. M., & Sirer, M. I. (2012). Who does what on the web: A large-scale study of browsing behavior. In *ICWSM*.

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, *151*, 78–94.

Gregor, S., & Benbasat, I. (1999). Explanations from intelligent systems: Theoretical foundations and implications for practice. *MIS Quarterly*, *23*(4), 497–530.
Grover, A. & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864). ACM.
Guillaume, J.-L., & Latapy, M. (2006). Bipartite graphs as models of complex networks. *Physica A: Statistical Mechanics and its Applications*, *371*(2), 795–813.
Gupte, M. & Eliassi-Rad, T. (2012). Measuring tie strength in implicit social networks. In *Proceedings of the 3rd annual ACM web science conference* (pp. 109–118). ACM.
Hu, J., Zeng, H.-J., Li, H., Niu, C., & Chen, Z. (2007). Demographic prediction based on user's browsing behavior. In *Proceedings of the 16th international conference on world wide web* (pp. 151–160). ACM.
Huang, H. S., Lin, K. L., & Hsu C.-N., & Hsu, J. Y. J. (2005). Item-triggered recommendation for identifying potential customers of cold sellers in supermarkets. In *Workshop on the next stage of recommender systems research, in conjunction with the 2005 international conference on intelligent user interfaces (IUI 2005)*.
Huang, X., Song, Q., Yang, F., & Hu, X. (2019). Large-scale heterogeneous feature embedding. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, pp. 3878–3885).
Huang, Z., Li, X., & Chen, H. (2005). Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on digital libraries* (pp. 141–142). ACM.
Jensen, D. & Neville, J. (2002). Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the nineteenth international conference on machine learning, ICML '02* (pp. 259–266). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*, 11–21.
Junqué de Fortuny, E., Martens, D., & Provost, F. (2013). Predictive modeling with big data: Is bigger really better? *Big Data*, *1*(4), 215–226.
Khosravi, H. & Bina, B. (2010). A survey on statistical relational learning. In *Canadian conference on artificial intelligence* (pp. 256–268). Springer.
Kim, J. H. (2017). Hypotheses generation using link prediction in a bipartite graph. CoRR arXiv :abs/1708.04725.
Kipf, T. N. & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, *110*(15), 5802–5805.
Lambiotte, R., & Ausloos, M. (2005). Uncovering collective listening habits and music genres in bipartite networks. *Physical Review E*, *72*(6), 066107.
Latapy, M., Magnien, C., & Vecchio, N. D. (2008). Basic notations for the analysis of large two-mode networks. *Social Networks*, *30*, 31–48.
Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, *58*(7), 1019–1031.
Lind, P. G., Gonzalez, M. C., & Herrmann, H. J. (2005). Cycles and clustering in bipartite networks. *Physical Review E*, *72*(5), 056127.
Liu, N., Huang, X., Li, J., & Hu, X. (2018). On interpretation of network embedding via taxonomy induction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '18* (pp. 1812–1820). New York, NY, USA: ACM.
Li, X., Wang, H., Gu, B., & Ling, C. X. (2015). Data sparseness in linear SVM. *IJCAI* (pp. 3628–3634).
Lu, Q., & Getoor, L. (2003). Link-based classification. In *ICML* (Vol. 3, pp. 496–503).
Macskassy, S. A., & Provost, F. (2003). *A simple relational classifier*. New York: NYU Stern School of Business.
Macskassy, S. A., & Provost, F. (2007). Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, *8*, 935–983.
Martens, D., Baesens, B., Van Gestel, T., & Vanthienen, J. (2007). Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, *183*(3), 1466–1476.
Martens, D. & Provost, F. (2011). Pseudo-social network targeting from consumer transaction data. Working paper CeDER-11-05, New York University—Stern School of Business.
Martens, D., & Provost, F. (2014). Explaining data-driven document classifications. *MIS Quarterly*, *38*(1), 73–100.
Martens, D., Provost, F., Clark, J., & de Fortuny, E. J. (2013). Mining fine-grained consumer payment data to improve targeted marketing. Working paper, New York University—Stern School of Business.

Martens, D., Vanthienen, J., Verbeke, W., & Baesens, B. (2011). Performance of classification models from a user perspective. *Decision Support Systems*, *51*(4), 782–793.

Newman, M. E. (2001a). Scientific collaboration networks. I. Network construction and fundamental results. *Physical Review E*, *64*(1), 16–131.

Newman, M. E. (2001b). Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*, *64*(1), 16–132.

Opsahl, T. (2011). Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*, *35*(2), 159–167.

Perlich, C. & Provost, F. (2003). Aggregation-based feature invention and relational concept classes. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 167–176). ACM.

Perlich, C., & Provost, F. (2006). Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning*, *62*(1–2), 65–105.

Perlich, C., & Świrszcz, G. (2011). On cross-validation and stacking: Building seemingly predictive models on random data. *ACM SIGKDD Explorations Newsletter*, *12*(2), 11–15.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710). ACM.

Provost, F., Dalessandro, B., Hook, R., Zhang, X., & Murray, A. (2009). Audience selection for on-line brand advertising: Privacy-friendly social network targeting. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 707–716). ACM.

Provost, F., & Fawcett, T. (2013). *Data science for business: What you need to know about data mining and data-analytic thinking*. Newton: O'Reilly Media Inc.

Provost, F., & Kolluri, V. (1999). A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, *3*(2), 131–169.

Provost, F., Martens, D., & Murray, A. (2012). Geo-social network advertising. In *2012 winter conference on business intelligence*.

Provost, F., Martens, D., & Murray, A. (2015). Finding mobile consumers with a privacy-friendly geosimilarity network. *Information Systems Research*, *26*(2), 243–265.

Raeder, T., Stitelman, O., Dalessandro, B., Perlich, C., & Provost, F. (2012). Design principles of massive, robust prediction systems. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1357–1365). ACM.

Robins, G., & Alexander, M. (2004). Small worlds among interlocking directors: Network structure and distance in bipartite graphs. *Computational & Mathematical Organization Theory*, *10*(1), 69–94.

Rrnyi, A. (1961). On measures of entropy and information. In *Fourth Berkeley symposium on mathematical statistics and probability* (pp. 547–561).

Seierstad, C., & Opsahl, T. (2011). For the few not the many? The effects of affirmative action on presence, prominence, and social capital of women directors in norway. *Scandinavian Journal of Management*, *27*(1), 44–54.

Sun, J., Qu, H., Chakrabarti, D., & Faloutsos, C. (2005). Neighborhood formation and anomaly detection in bipartite graphs. In *Proceedings of the fifth IEEE international conference on data mining, ICDM '05* (pp. 418–425). Washington, DC, USA: IEEE Computer Society.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077). International World Wide Web Conferences Steering Committee.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1225–1234). ACM.

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., & Guo, M. (2018). Graphgan: Graph representation learning with generative adversarial nets. In *Thirty-second AAAI conference on artificial intelligence*.

Weber, I., Garimella, V. R. K., & Borra, E. (2013). Inferring audience partisanship for youtube videos. In *Proceedings of the 22nd international conference on world wide web companion* (pp. 43–44). International World Wide Web Conferences Steering Committee.

Young, H. P., & Levenglick, A. (1978). A consistent extension of Condorcet's election principle. *SIAM Journal on Applied Mathematics*, *35*(2), 285–300.

Yu, H.-F., Lo, H.-Y., Hsieh, H.-P., Lou, J.-K., McKenzie, T. G., Chou, J.-W., Chung, P.-H., Ho, C.-H., Chang, C.-F., Wei, Y.-H., et al. (2010). Feature engineering and classifier ensemble for KDD cup 2010. In *Proceedings of the KDD cup 2010 workshop* (pp. 1–16).

Zha, H., He, X., Ding, C., Simon, H., & Gu, M. (2001). Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on information and knowledge management* (pp. 25–32). ACM.

Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2018). Network representation learning: A survey. *IEEE Transactions on Big Data*, *6*(1), 3–28.

Zhang, Y., Xiong, Y., Kong, X., & Zhu, Y. (2017). Learning node embeddings in interaction graphs. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 397–406).

Zhou, T., Ren, J., Medo, M., & Zhang, Y.-C. (2007). Bipartite network projection and personal recommendation. *Physical Review E*, *76*(4), 046115.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., & Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on world wide web* (pp. 22–32). ACM.

Zweig, K. A., & Kaufmann, M. (2011). A systematic approach to the one-mode projection of bipartite graphs. *Social Network Analysis and Mining*, *1*(3), 187–218.