DEPARTMENT OF ENGINEERING MANAGEMENT

Classification over bipartite graphs through projection

Marija Stankova, David Martens & Foster Provost

UNIVERSITY OF ANTWERP
Faculty of Applied Economics

City Campus
Prinsstraat 13, B.226
B-2000 Antwerp
Tel. +32 (0)3 265 40 32
Fax +32 (0)3 265 47 99
www.uantwerpen.be

FACULTY OF APPLIED ECONOMICS

DEPARTMENT OF ENGINEERING MANAGEMENT

Classification over bipartite graphs through projection

Marija Stankova, David Martens & Foster Provost

RESEARCH PAPER 2015-001
JANUARY 2015

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
fax: (32) 3 265 47 99
e-mail: joeri.nys@uantwerpen.be

The research papers from the Faculty of Applied Economics are also available at www.repec.org (Research Papers in Economics - RePEc)

D/2015/1169/001

manuscript No.
(will be inserted by the editor)

Classification over bipartite graphs through projection

Marija Stankova · David Martens · Foster Provost

Received: date / Accepted: date

Abstract Many real-world large datasets correspond to bipartite graph data settings; think for example of users rating movies or people visiting locations. Although some work exists over 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. First, a weighting of the top nodes is defined. Secondly, the bigraph is projected into a unipartite (homogenous) graph among the bottom nodes, where the weights of the edges are a function of the weights of the top nodes in the bigraph. Finally, relational learners/classifiers are applied to the resulting weighted unigraph. This general framework allows us to explore the design space, by applying different choices for 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. There are clear differences in predictive performance with different design choices. Based on these results, we propose several specific combinations that show good accuracy and also allow for easy and fast scaling to big datasets. A comparison with a linear SVM method on the adjacency matrix of the bigraph shows the superiority of the network-oriented approach.

Keywords Bipartite graphs · Two-mode networks · Affiliation networks · Node classification · Big Data

M. Stankova, D.Martens
University of Antwerp, Belgium
Prinsstraat 13, Antwerp, Belgium
E-mail: {marija.stankova, david.martens}@uantwerpen.be

F.Provost
New York University, New York, USA
44 West Fourth Street, 8-86, New York, USA
E-mail: fprovost@stern.nyu.edu

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 (i) having two types of nodes and (ii) edges exist only between nodes of different type. Think for example of relationships based on companies' board members [58], users meeting at locations or events [12], users rating different products [65], consumers making payments to merchants [43], mobile devices visiting locations [54], authors collaborating on scientific papers [48], people communicating on online forums [49], actors playing in the same movies [23], words occurring in the same sentence/search query [23,28], proteins involved in the same metabolic processes [23], etc. The analysis of bigraph data has so far been mainly limited to measuring descriptive statistics, link prediction for recommender systems, and clustering (see Section 5). The task we consider is node classification [41] within bigraphs, where nodes for which the class is known are related to nodes for which the class must be estimated. For this task, no general network-based methodology has been 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 with massive, sparse feature data (for example predicting personality traits from datasets of Facebook users liking pages [33] or predicting demographic attributes [20,25] and brand interest [55] from people's browsing history, predicting political views from history of videos watched on YouTube [60] and etc.). In this paper, we examine an alternative, network-based formulation.

As an example of node classification we can consider a bigraph of users and locations, where users are connected to the locations they visited (e.g. logged into a WiFi IP address [54] or checked in using a social network app [9]). Our goal would be to predict brand interest of the users based on these location data, to target mobile ads. Based on the brand interest of other users visiting the same locations we can infer the (likelihood of the) class of the unknown user [54]. This task differs from the task of collaborative filtering for recommender systems [32]. The collaborative filtering task here would be to predict other locations a user would be interested in visiting next. This presents a link prediction problem, where new associations are looked for between users and locations. Unlike collaborative filtering, our task is to use the links between the nodes (users and locations) to predict a certain *feature* of a node (the user's brand interest) and not the presence of a link between nodes.

The following section reviews the approaches used in prior work to analyze bigraphs. In this paper we consider unweighted bipartite graphs with binary values only, an assumption in line with many datasets. Additionally, our methodology assumes a binary target variable. We do however also consider multiclass datasets in our experimental setup by casting these to multiple binary bigraphs.

1.1 Approaches to analyzing bigraphs and motivation

In previous research there have been two main approaches to analyze bigraphs with the aim of obtaining summary metrics and summary graphs. The first one is using techniques and metrics which are specially designed for the bipartite graphs [36]. This direct approach takes into account the bipartite nature of this particular type of graph. Unfortunately there are few techniques that can be applied directly on the bigraph, so a second, indirect approach is often used—and this is the basis of the methodology that we propose.

Let's separate the two subsets of nodes in the bigraph into the *top* nodes and *bottom* nodes. Choosing the nodes to focus on 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 [36]. This projection approach allows the application of existing network analysis techniques for unigraphs to the bipartite case. This is the basis for our proposal of a general framework for classification in bigraph data. It is very convenient for this problem, as numerous relational classifiers for network data exist for homogeneous graphs. A key to operating on massive bigraph data is that many of the relational classifiers make a first-order Markov assumption on the network, meaning that they will only consider the neighboring nodes.[1] As we will discuss below, the projection should be created in such a way that it can preserve as much information as possible from the original bipartite graph, for example by creating and employing link weights.

If we consider the examples of bigraphs listed before, we can see that many of them involve relationships among people and most of the relationships are based on two types of nodes: persons and so called "focal points" of social interaction or *foci* [13]. These foci can be any type of social (e.g. events, board meetings, online forums, locations people visit etc.) or physical entities (e.g. people interested in the same books or movies, consumers making payments to the same merchants, authors collaborating on scientific papers). By visiting the same locations or being involved in the same social activities, the persons get the opportunity to meet each other and by that create a link in the projected network. In many situations, people tend to become friends with people with whom they share similar interests or characteristics—in our context people with whom they share the same foci. This is one basis for "social selection" [13]. On the other hand, sharing the same foci can also be a consequence of social influence [13], where friends can influence their friends' choices of foci. Selection and social influence are the theoretical principles that explain why ties among similar people are preferentially formed [13]. This results in

[1] Technically, if there are neighboring nodes for whom the value of the random variable being used for prediction is unknown, relational classifiers would have to perform collective inference [41]. Except where explicitly discussed, we will consider the network in question to be the training data, for which all values for the target variable are known.

social networks where people tend to be similar to their friends, also known as network assortativity, which is closely related to homophily [46,13].[2]

We look beyond actual social networks and consider bigraphs of transactional and behavioral data in general. Even when one set of nodes comprises people, links in the projection could be created between persons that do not know each other. Our premise is that people that are similar in one domain (e.g. preferences, behavior) would act similarly in other domains as well [54]. For example, users that have similar preferences for some locations like specific bars or restaurants are likely also similar in age, social status and other features. There has been previous research around this concept of cross-domain similarity. A study by Martens and Provost consider payments from consumers to merchants to define a bigraph, which is used to predict interest in financial products [43]. Provost et al. consider website visits to link browsers (which can be considered as proxies for people) to websites, and aim to predict the likelihood of being interested in a brand [51]. Another study by Provost et al. [54] uses geo-location data to connect people if they visited the same places with the goal of predicting brand interest. These people do not need to know each other, so the resulting projection can be considered as a pseudo-social network. The concept of similarity is not limited to behavioral data that involve people. It can be expanded to bigraph data with any arbitrary nodes, for example bigraphs of proteins connected if they are active in the same metabolic processes [23].

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. Before going into more detail, let us briefly discuss why this projection approach might work better than alternative approaches.

1.2 Alternative classification techniques

Seeing that typical bigraph datasets are very large transactional datasets, our proposed method is designed to scale up easily to millions and even billions of nodes. As an alternative to the graph approach for node prediction, one could also represent the data by the corresponding adjacency matrix, with as many rows as there are bottom nodes and as many columns as there are top nodes. Clearly this will be a very sparse matrix as most elements in this matrix will be zero. Why would our projection approach work better than simply applying classification techniques, such as support vector machines (SVM), to this dataset?[3]

[2] Indeed, although homophily originally corresponded to a principle of social selection, in contemporary usage it often simply refers to assortativity in social networks.

[3] Although it is a natural, and interesting question, whether the projection approach does better than a particular traditional classification technique is not the main point. Rather, we generalize what researchers and practitioners already are doing with this sort of data, and thereby provide a family of methods with many more design options than have been considered previously. Some combinations of options may perform very well for a particular data set. The framework facilitates a systematic exploration.

Let us consider a huge transactional dataset, where for each person on the planet we keep all the locations that he/she visited over the last month with a target variable brand interest (hence using location data to target mobile ads). With a moderately fine-grained specification of location, this would result in a dataset of size 7 billion x 100 billion. Now for each individual person we want to predict the potential brand interest. The network approach would consider only the neighboring nodes in the bigraph, specifically the (for example 10 or 100) locations this person has visited, and consider all other training individuals that also visited those locations (viz., have a one in those columns). This could immediately reduce the problem of considering a few billion to only hundreds or thousands of training points. For those, the strength of the link in the projected unigraph is computed and a relational classifier is applied (details on this follow in the next sections). The kNN approach with typical Euclidean norm would require us to calculate the distance between the location profile of this person and every other person in the world. Even persons that visited none of the locations of the test person can have a different distance to the test person, depending on the other locations visited. Clearly, this does not scale. We will revisit creating specific, sparse distance functions below. An SVM would need to build a predictive model on the huge 7 billion x 100 billion dataset, which will not be feasible, requiring sampling and dimensionality reduction (and dimensionality reduction techniques such as singular value decomposition also scale badly to these settings) [45]. Once more, scalability issues impede the easy application of this alternative. In our empirical work, we further discuss this scalability requirement and include SVMs for the smaller datasets as a benchmark.

Let us discuss in more detail the question of how kNN, and other types of nearest neighbor techniques differ from our proposal. When using a nearest neighbor technique, there is a need for searching the most similar nodes. In other words, a similarity function has to be calculated for each test node over the whole training set, which in general is not scalable to high dimensions. Most common approaches in literature to this problem include nearest neighbor techniques that either reduce the training set [10], look for the approximate neighbors [3] or use indexing structures based on space partitioning to represent the data [10]. For the latter, Weber et al. [61] have shown that for large enough dimensions, the performance of such methods degrades to a basic linear search. On the other hand, this need for a similarity search is eliminated when using our approach. We take advantage of the network structure, which (i) provides a natural "index" to the node's neighbors, and (ii) focuses on similarity functions that consider only shared neighbors, thereby allowing for faster processing especially in a sparsely connected bigraph (as we often encounter, for example from data on human choice behavior [31]). The main reason why nearest neighbor techniques are not suitable for these kind of datasets is the lack of scalability for traditional metrics such as Euclidean distance. One could also choose or create distance metrics that explicitly take into account the sparseness of the data, where the distance between two instances is zero if there are no columns with non-zero elements for both instances; this

essentially would correspond to the bipartite network projection method we propose, where nodes are linked if they share a top node. However, several design questions remain, such as what particular metric to use when the distance is non-zero; we propose a set of possible metrics (Table 2).

When such a sparsity-oriented distance metric is combined with a weighted nearest-neighbor classifier [10], which takes into account the similarity to all nodes and in the combining function weights their contribution per class by their similarities, we derive one instance of the projection method with particular choices for the components: a particular weight assignment ($s_k = 1$), the aggregation function corresponding to the chosen distance metric (with the addition of distance equal to zero if there are no columns with non-zero elements for both instances), and wvRN [41] as the relational classifier. So, the three-stage projection approach we propose can be instantiated to be a specific (non-traditional) instance of a nearest-neighbor classifier. Whether these are the best design choices for a particular problem requires empirical examination, and one of the advantages of a flexible framework is that different design choices can be compared easily and on equal footing.

1.3 Contributions

The work presented in this paper provides four main contributions:
1. It surveys work on analysis in bipartite data via projection. There is a non-trivial amount of work, and it previously has not been collected and analyzed as a specific area of study.
2. It provides a general framework for doing classification in bipartite data via projection. This framework is informed by the survey of prior work. It also generalizes the prior work in an important way: by dividing the process into three stages, we can explore the design space more systematically than prior work has done. In particular, we can look specifically at the different choices for the three stages, introduce new alternatives, and mix-and-match to create new techniques.
3. It presents a large benchmark collection of bipartite classification datasets. To our knowledge, previously no one has assembled a benchmark collection of such datasets. With such a collection we can examine the design choices empirically.
4. It provides an empirical study over the benchmark collection, examining the generalization performance and run-time performance of different methods for bipartite classification via projection. There are clear differences in predictive performance with different design choices. The best performing method is a new combination of existing techniques, and generally new combinations revealed by the 3-stage framework often are among the best performing methods.

The rest of the paper is structured as follows: Section 2 describes a range of functions for weighting top nodes, aggregation functions, and relational classifiers used in this paper. Section 3 gives an overview of the datasets and

the experimental setting used to validate our methods and Section 4 presents our findings. Section 5 gives an overview of the existing literature for bigraph analysis. Section 6 concludes.

2 Methods

A bigraph can formally be defined as the triplet $G = (\top, \bot, E)$, where $\top$ denotes a set of top nodes, $\bot$ is the set of bottom nodes and $E \subseteq \top \times \bot$ is the set of links.

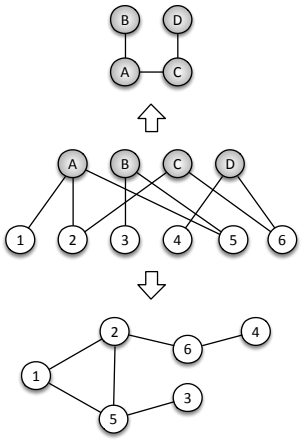
2.1 Bigraph Properties

Latapy et al. [36] extend the basic properties of unipartite graphs to the bipartite case. 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 in the bigraph $n_\bot = |\bot|$ and m the total number of edges. The average degree of the top nodes is defined as $k_\top = \frac{m}{n_\top}$ and analogously the average degree of the bottom nodes as $k_\bot = \frac{m}{n_\bot}$. The average degree k over the whole bipartite graph is $k = \frac{2m}{n_\top + n_\bot}$. The density of the bipartite graph can be obtained by $\delta(G) = \frac{m}{n_\top \cdot n_\bot}$.

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 connecting 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. Therefore, given a bigraph $G = (\top, \bot, E)$, the projection of the bottom nodes ($\bot$ projection), defined as $G' = (\bot, E')$, with the set of edges $E' \subseteq \bot \times \bot$, can be obtained by connecting the nodes in $\bot$, which 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 consider only the bottom node projection. For example, in Figure 1(left) we can see a bigraph, along with its $\top$-projection and its $\bot$-projection. The adjacency matrix of the same bigraph can be seen on Figure 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.

The projection approach 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 example, in the projections of Figure 1 we lose information associated to the opposite node set, like the degree distributions, number of shared nodes and their identity, etc. By intelligently assigning weights to the edges in the projection graph, we can incorporate information about the top nodes and better reflect the underlying structure of the bigraph. Therefore in this paper, 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



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

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

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 weight of the edges in the projection by combining the weights of the shared top nodes between the bottom nodes. This additionally includes information about the number of shared nodes in the projection's weights.
3. Finally, we use relational classifiers on the weighted unigraphs in order to predict the values for the target variables. The relational classifiers use only the graph structure to make predictions, which eliminates the need for local information about the top nodes.

2.2 Determining importance of top nodes

A set of functions for calculating the weights of the top nodes in the bigraph, s_k, are listed in Table 1 and visualised in Figure 2. The simplest weighting scheme for the top nodes is assigning equal importance, $s_k = 1$, to all the nodes. Although it is an easy and basic method to use, it does not make a distinction between the top nodes. Some more complex weighting methods are based on the degree d_k (number of connections) of the top node k. One such a method is the inverse degree, referred to as "linear" by Gupte and Eliassi-Rad [24]. Another one is the inverse degree frequency, which is an analogy to a commonly used measure in information retrieval, called Inverse Document Frequency (IDF) [30], which is closely related to measures of entropy [52]. With IDF, very common terms that occur in many documents are assigned

lower weights since they are less likely to be good discriminators. The inverse 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. This weighting scheme was proposed by Martens and Provost [43]. An alternative weighting method for 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. A different approach is the use of the delta function as defined in Allali et al. [2]. This function takes into account that each top node has influence on the similarity between all pairs of bottom nodes which are connected to it. Therefore, a top node with a degree d, has an impact over $\frac{d(d-1)}{2}$ pairs of bottom nodes of 1/number-of-pairs. The Adamic-Adar measure [1] 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).

Top node weight function	Formula
Simple weight assignment	$s_k = 1$
Inverse degree	$s_k = \frac{1}{d_k}$
Inverse frequency	$s_k = log_{10}(\frac{N}{d_k})$
Hyperbolic tangent	$s_k = tanh(\frac{1}{(d_k)})$
Adamic and adar	$s_k = \frac{1}{log_{10}(d_k)}$
Delta	$s_k = \frac{2}{d_k(d_k-1)}$
Beta distribution	$s_k = B(\alpha, \beta, (\frac{max(d_k)-d_k)}{max(d_k)-min(d_k)}))$
Likelihood ratio	$s_k = \frac{d_k^c}{d_k}$

Table 1: Overview of the functions for determining top nodes weight.

As one can observe from Figure 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 [24]. In other words, (again) top nodes with lower degree receive higher weights.

In contrast, one might argue that 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 collecting 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 tradeoff between specificity and noise tolerance [45]. To this end we employ the beta distribution density function, defined by Equation (1) over the interval $x \in [0, 1]$. Here, α and β are two parameters of the density function, which are positive numbers

and define the shape of the density curve. The beta distribution is commonly used in Bayesian analysis as a prior distribution for binominal proportions [17]. 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.

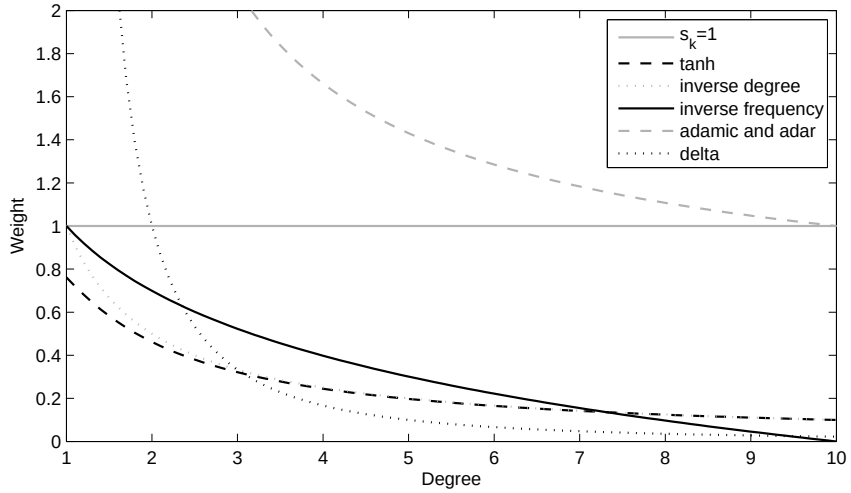


Fig. 2: Functions for determining top nodes weight.

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

$$B(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx \tag{1}$$

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 [43,45]. 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.

2.3 Determining the weights of the edges in the projection

After we have determined the weights s_k of the top nodes we calculate the weights w_{ij} of the edges in the unipartite projection. This weight w_{ij} between

Aggregation function	Formula
Sum of shared nodes	$w_{ij} = \sum_{k \in N(i) \cap N(j)} s_k$
Max of shared nodes	$w_{ij} = \max_{k \in N(i) \cap N(j)} s_k$
Jaccard similarity	$w_{ij} = \frac{\sum_{k \in N(i) \cap N(j)} s_k}{\sum_{k \in N(i) \cup N(j)} s_k}$
Cosine similarity	$w_{ij} = \frac{\sum_{k \in N(i) \cap N(j)} s_k^2}{\sqrt{\sum_{k \in N(i)} s_k^2} \cdot \sqrt{\sum_{k \in N(j)} s_k^2}}$
Zero-one	$w_{ij} = \begin{cases} 1 & \text{if} \sum_{k \in N(i) \cap N(j)} s_k > 0 \\ 0 & \text{if } \sum_{k \in N(i) \cap N(j)} s_k = 0 \end{cases}$

Table 2: Overview of the aggregation functions.

the bottom nodes i and j is an aggregation of the weights s_k of the shared top nodes (see Table 2). The most straightforward method is to simply sum their weights. Another approach is to select the maximum weight of the shared top nodes as a weight of the edge in the projection [24]. 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 [54]. 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. Then, even when all the neighbors of one of the nodes are also neighbors of the other node, the similarity will be low. The cosine similarity function calculates the similarity between pairs of vectors, by calculating the cosine value of the angle between them [54]. Two bottom nodes have a cosine similarity of 1 if they share exactly the same top nodes and 0 if they don't have any neighbors in common. Finally, a very simple weighting 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. 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.

2.4 Relational classifiers

The third step of the framework for node classification within bigraphs is to use a relational or network classifier over the unigraph projection. We consider methods for within-network classification, specifically for univariate networks, as defined by Macskassy and Provost [41]. Specifically, 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, relational classifiers use information about related nodes [41]. Univariate relational (network) classifiers use information about the target variable for the related nodes (their labels or predictions thereof). Macskassy and Provost compared several relational classifiers using

the software package NetKit. 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 [40] 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, calculated as a weighted average (see Equation (2)).

$$P(l_i = c|N(i)) = \frac{1}{Z} \sum_{j \in N(i)} w_{ij} P(l_j = c|N(j)) \tag{2}$$

The second relational classifier used in this study is the class-distribution Relational Neighbor (cdRN) classifier [41]. Unlike the previous classifier it takes into account the class distribution linkages of the whole training set, and not only the immediate neighborhood, 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) (Equation 3). 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 (Equation 4).

$$CV(i)_c = \sum_{j \in N(i), l_j = c} w_{ij} \tag{3}$$

$$RV(c) = \frac{1}{\mid \perp^c \mid} \sum_{i \in \perp^c} CV(i) \tag{4}$$

$\perp^c$ are the bottom nodes in the bigraph known to have label $l_i = c$.

The probability of a node i having class c can than be estimated as the normalized vector similarity between the class vector of node i ($CV(i)$) and the reference vector RV (Equation 5). The vector similarity function we use is cosine similarity, but other functions such as L1, L2 normalized in the range of [0,1] can also be used.

$$P(l_i = c|N(i)) = sim(CV(i), RV(c)) \tag{5}$$

A more complex relational classifier is the network-only Link-Based classifier (nLB) [39]. The nLB 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 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 [39] (Equation (6)). In the next step, the nLB classifier builds a logistic regression model based on these class vectors (Equation (7)).

$$CV(i)_c = \frac{\sum_{j \in N(i)} w_{ij} P(l_j = c|N(j))}{\sum_{j \in N(i)} w_{ij}} \tag{6}$$

$$P(l_i = c|N(i)) = \frac{1}{1 + e^{-\beta_0 - \beta CV(i)}} \tag{7}$$

2.5 Decomposition of metrics

We consider a 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 literature for defining link weights in bigraph projections and 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. As an example, the Adamic-Adar coefficient [1] (see Equation (8)), can be decomposed into the Adamic-Adar top node weighting function (Table 1) and the sum of shared nodes aggregation function (Table 2). This clearly creates opportunities for combining the existing weighting functions in new ways, resulting in completely new measures. Note that some of the combinations do not include a relational classifier, since these studies use the unigraph projection for other tasks rather than classification, like link prediction [2,24], measuring descriptive statistics [47,48,23] and etc. Moreover, some studies consider the unweighed unigraph projections [47,23]. Since this is independent of the top nodes weight, we can apply any top node function in the first step of the framework.

$$w_{ij} = \sum_{k \in N(i) \cap N(j)} \frac{1}{\log_{10}(d_k)} \tag{8}$$

Top node weight	Aggregation function	Rel. classifier	Ref.
any	Zero-one	-	[47,23]
Simple weight assign. ($s_k = 1$)	Sum of shared nodes	-	[2,24]
Inverse degree	Sum of shared nodes	-	[24,48]
Adamic and adar	Sum of shared nodes	-	[1,24]
Delta	Sum of shared nodes	-	[2,24]
Simple weight assign.($s_k = 1$)	Jaccard similarity	-	[2,24]
Inverse degree	Max of shared nodes	-	[24]
Simple weight assign.($s_k = 1$)	Sum of shared nodes	wvRN	[54,41]
Inverse frequency	Sum of shared nodes	wvRN	[54]
Inverse frequency, likelihood ratio	Sum of shared nodes	wvRN	[43,45]
Beta distribution, likelihood ratio	Sum of shared nodes	wvRN	[45]
Simple weight assign. ($s_k = 1$)	Jaccard similarity	wvRN	[54]
Inverse frequency	Jaccard similarity	wvRN	[54]
Simple weight assign. ($s_k = 1$)	Cosine similarity	wvRN	[54]
Inverse frequency	Cosine similarity	wvRN	[54]
Simple weight assign.($s_k = 1$)	Sum of shared nodes	cdRN	[41]
Simple weight assign.($s_k = 1$)	Sum of shared nodes	nLB	[41]

Table 3: Overview of measures for defining links' weight in bigraph projections used in previous literature.

2.6 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; method 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.

2.6.1 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 (cf., [53]). A partial projection can be created over the batch and then used by the relational classifier to calculate partial scores of the labels or for training. For the network-only Link-Based (nLB) classifier (see Equation (6)), the projection of each batch from the training dataset is used to calculate a part of the class vector. When aggregated the whole class vector is obtained and then used by the logistic regression. Analogously, the label scores for the test dataset will be created sequentially. Each projection of a batch will produce a partial score, with all the scores aggregated in the end into a final solution. The sizes of the batches in this study were determined experimentally, by testing different batch sizes for each dataset. If the batch size is too large, it will not be possible to process it 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, if the batches are too small, it also will take too much time to process the dataset. Each batch introduces additional calculation overhead. Therefore the size of the batch should be a balance between the two. Batch processing would also allow for easy scaling up using parallel and distributed computing systems [53].

2.6.2 Sampling

Another technique that enables the network-only Link-Based (nLB) classifier to scale to larger datasets and improve the 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 for training 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.

2.6.3 Grid Search

Fine tuning the parameters of the optimal top nodes function beta (Equation (1)), can require many iterations. In order to reduce the number of iterations a *grid search* with multiple levels, in our case three, can be applied. Here, each level performs a more fine-grained search around the best selected parameter value from the previous level. The grid we use, searches for the optimal α and β on the first level in the range between 0.1 and 12.1 with steps of 3. Each successive level looks for the parameters in the range from the previous level: for example if the best parameter value from the previous level is x, this level will look in the range [x-step, x+step], where step is the size of the step in the previous level. The size of the step is three times smaller at each new level. This yields significant runtime improvements in comparison to extensive search of the space, while giving the same solution granularity. As we will discuss further in Section 4, one can also perform grid search of one or two levels with limited performance degradation.

2.6.4 SW transformation

Finally, we introduce a fast method, called *SW-transformation*, to calculate the label scores for the case where wvRN (Equation (2)) 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. As we describe next (Equations 9-17) this specific combination can be rewritten as a fast linear model over the top nodes. We start by transforming the wvRN formula, where we substitute the weights of the projection w_{ij} with the aggregation function in Equation (9). wvRN takes into account only the labels of the neighboring nodes, therefore in Equation (10) we consider only the neighboring bottom nodes j that have element $a_{ij} = 1$ in the adjacency matrix of the projected unigraph (See Figure 4). The weight w_{ij} of a link in the projection is calculated by summing the weights of the top nodes shared by both nodes i and j. This means that the top nodes that are not neighbors of node i in the bigraph (have elements $x_{ik} = 0$ in the adjacency matrix from Figure 3) can be discarded in Equation (12). The node labels y_j can have values one or zero. When predicting an attribute of a node, the wvRN takes into account only the neighboring nodes that have the label class of interest, $y_{ij} = 1$. This leads to eliminating neighboring nodes with label $y_{ij} = 0$ in Equation (14). The transformation can also be applied to cases when the node labels are score probabilities. Note that the neighboring nodes with label $y_{ij} = 0$ are still counted when calculating the normalization factor $Z = \sum_{j \in N_{(i)}} w_{ij}$ in Equation (9), which is a sum over the link weights to all neighboring nodes.

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

Fig. 3: $m \times n$ matrix representation of a bigraph.

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

Fig. 4: $m \times m$ matrix representation of the projected unigraph.

The result is the SW-transformation in Equation (17), a linear model that calculates the label scores directly on the bigraph. Here the projection (where we calculate the weights of the links between each pair of bottom nodes) is not directly created. Instead, the SW-transformation optimally takes into account only the neighboring bottom nodes that have a label $y_{ij} = 1$. In terms of implementation, this means that the SW-transformation will consider for each test instance in the bigraph adjacency matrix only the columns where the instance has element one (it is connected to that top node). Then it will consider the number of training instances in that column which have label one (the positive neighbors of the node) multiplied by the weight of the top node. This means that we directly calculate the influence of the top node, in the

form of the coefficient of the corresponding linear model.

$$Z \cdot P(l_i = c|N(i)) = \sum_{j \in N(i)} w_{ij} \cdot P(l_j = c|N(j)) \tag{9}$$

$$= \sum_{j|a_{ij} \neq 0} w_{ij} \cdot y_j \tag{10}$$

$$= \sum_{j|a_{ij} \neq 0} \left(\left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot y_j + \left(\sum_{k|x_{ik} = 0} s_k \right) \cdot y_j \right) \tag{11}$$

$$= \sum_{j|a_{ij} \neq 0} \left(\left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot y_j + 0 \right) \tag{12}$$

$$= \sum_{\substack{j|a_{ij} \neq 0 \\ y_j = 0}} \left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot y_j + \sum_{\substack{j|a_{ij} \neq 0 \\ y_j = 1}} \left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot y_j \tag{13}$$

$$= 0 + \sum_{\substack{j|a_{ij} \neq 0 \\ y_j = 1}} \left(\sum_{k|x_{ik} \neq 0} s_k \right) \cdot 1 \tag{14}$$

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

$$= \sum_{k|x_{ik} \neq 0} \left(s_k \sum_{\substack{j|x_{ik} \neq 0, x_{jk} \neq 0 \\ y_j = 1}} 1 \right) \tag{16}$$

$$= \sum_{k|x_{ik} \neq 0} s_k \cdot ns_k \tag{17}$$

where $ns_k = |x_{jk} = 1$ and $y_j = 1|$ and s_k is a weight of a top node in Equation (17).

The SW-transformation yields substantially faster run times (compared to calculating the whole projection) and allows easy scaling of the method to big datasets of up to millions of nodes, as we discuss further in Section 4. If we consider that most of the today's big dataset are very sparse, with nodes being connected to only few other nodes in the projection, we can clearly see the usefulness and the applicability of the SW-transformation.

3 Data and Experimental Setup

For this study, we collected bipartite datasets from various sources: the Koblenz Network Collection (KONECT),[4] the MIT Reality Mining Project,[5] the social networks collection of The Max Plank Institute for Software Systems,[6] 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 example using a bigraph between users and books they rated to predict the number of books they have read is clearly not suitable. The included datasets are summarized in Table 4 and to our knowledge comprise the first large collection of benchmark datasets for node classification over bigraphs. The basic bipartite statistics were defined in Section 2.1. The variables l_0 and l_1 denote the number of bottom nodes in the graph with label 0 and 1 respectively. For multiclass problems these values were omitted.[7]

The *MovieLens* dataset contains information about movie ratings from users of the MovieLens web site, collected from September 1997 through April 1998.[8] The bigraph is defined between users and movies, with links 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 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). The *Yahoo Movies* dataset[9] has a similar setting, where we are predicting the gender and the age of users who rated movies. The dataset collected by Opsahl and Seierstad [58] is used to define a bigraph between *Norwegian companies* and their board members, and we are predicting the gender of the board members. Information about mobile phone usage and location of users in the period between September 2004 and June 2005 was collected by The *Reality Mining* project [12] from the MIT Human Dynamics Lab. From this dataset we 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 stuff, professor, etc.

[4] http://konect.uni-koblenz.de

[5] http://realitycommons.media.mit.edu

[6] http://socialnetworks.mpi-sws.org

[7] 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.

[8] http://www.grouplens.org

[9] http://webscope.sandbox.yahoo.com/

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	2,206	5,436	11,915	7,642	221,330	18.57	28.96	22.63	0.0024
Yahoo Movies (above average)	gender	2,206	5,431	10,547	7,637	181,470	17.20	23.76	19.96	0.0023
Yahoo Movies	age	2,750	4,855	11,911	7,605	220,595	18.52	29.01	22.61	0.0024
Yahoo Movies (above average)	age	2748	4852	10544	7600	180880	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	5,051	11,299	18,126	16,350	234,355	12.93	14.33	13.59	7.9078e-04
Norwegian companies	gender	513	908	355	1,421	1,746	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 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 next dataset is the *Book-Crossing* dataset which includes book ratings from Bookcrossing.com users collected August – September 2004 [65]. The bigraph is defined between users and books. Here, the goal is to predict the age of the users. The *LibimSeTi* dataset [8] contains data about profile ratings from users of the Czech social network LibimSeTi.cz. The variable that we are predicting is the gender of the users. The next dataset is from the social networks collection of The Max Plank Institute for Software Systems and contains data for several million *Flickr* pictures. Here, the bottom nodes of the bigraph represent the pictures, the top nodes are the Flickr users with a link if the user marked a picture as favorite. The class variable to predict is the number of comments of the pictures. The *Ta-Feng* dataset contains transactional data from customers buying products and our prediction goal is the age of the customers, based on the products they bought [26]. 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, National Taiwan University, expanded the original dataset by converting the categorical features into sets of binary features [62], which can be downloaded from the LibSVM website.[10]

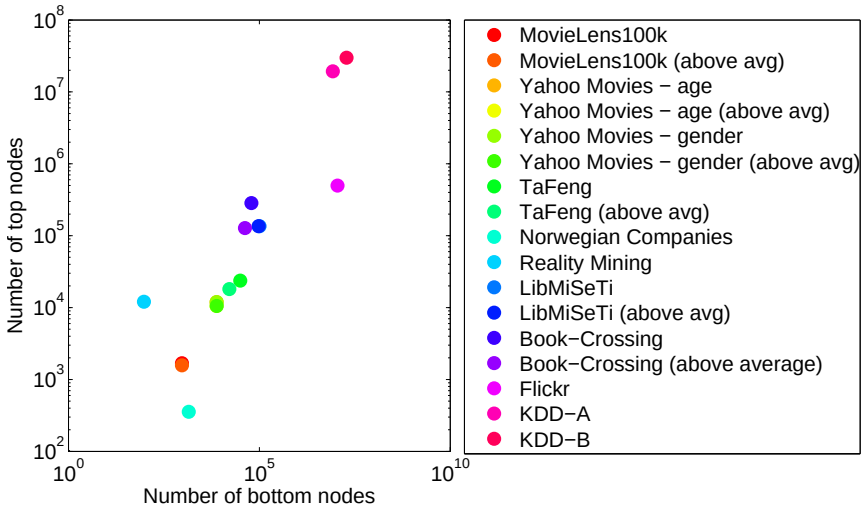


Fig. 5: Size of the datasets.

Additionally, for each dataset where the links represent some kind of ranking, we created new bigraph datasets where bottom nodes are connected to the top nodes only if the rating was positive (defined as higher than the average rating). This created several new datasets, annotated as "above average" in Table 4.

[10] http://www.csie.ntu.edu.tw/c̃jlin/libsvmtools/datasets/

Top node weight	Aggregation function	Relational classifier
Simple weight assignment	Sum of shared nodes	wvRN
Inverse degree	Max of shared nodes	nLB
Inverse frequency	Jaccard similarity	cdRN
Hyperbolic tangent	Cosine similarity	
Adamic and adar	Zero-one	
Beta distribution		
Delta		
Likelihood ratio		

Table 5: Overview of the proposed three step framework.

Figures 6-9 show the degree distributions for the bottom and top nodes of each dataset (the distributions of the probability $P(k)$ that a node has degree k). 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 very many other nodes. The top nodes of the LibMiSeTi dataset do not follow the power law: 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. Figure 5 gives an overview of the sizes of the datasets used in this study. As shown, the datasets differ in size. We have datasets that range from fewer than 100 bottom nodes (Reality Mining, number of people involved) and a few hundred top nodes (Norwegian companies, number of companies involved) up to a few million top and bottom nodes (KDDa and KDDb datasets).

In our empirical assessment, we consider 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 $8 \times 5 \times 3$ combinations (see Table 5), which are assessed on the 58 datasets (including the casted multiclass datasets). However, some of the combinations do not scale well to datasets with high dimensions such as the Flickr dataset or larger. The most scaleable methods are the sum of shared nodes function and the zero-one function, the latter basically converting the non-zero weights of the previous function into one. On the other hand, the beta function, which employs multiple iterations to fine tune the parameters, is very time consuming for bigger datasets.

As a benchmark, we additionally apply an SVM with linear kernel on the adjacency matrix defined by the bigraph using the libLINEAR toolbox [14] with default settings. The evaluation is based on the area under the ROC curve (AUC) [16]. A Wilcoxon signed rank test is used to assess significance differences. All experiments are conducted on a 3.40 GHz Intel i7 CPU with 8 GB RAM and a 64-bit operating system.

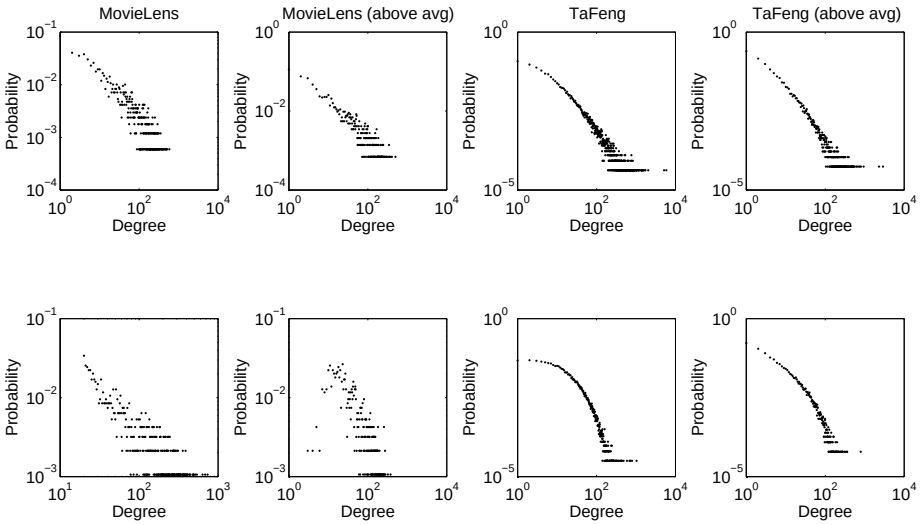


Fig. 6: Degree distributions of the top nodes (upper row) and bottom nodes (bottom row) for different datasets.

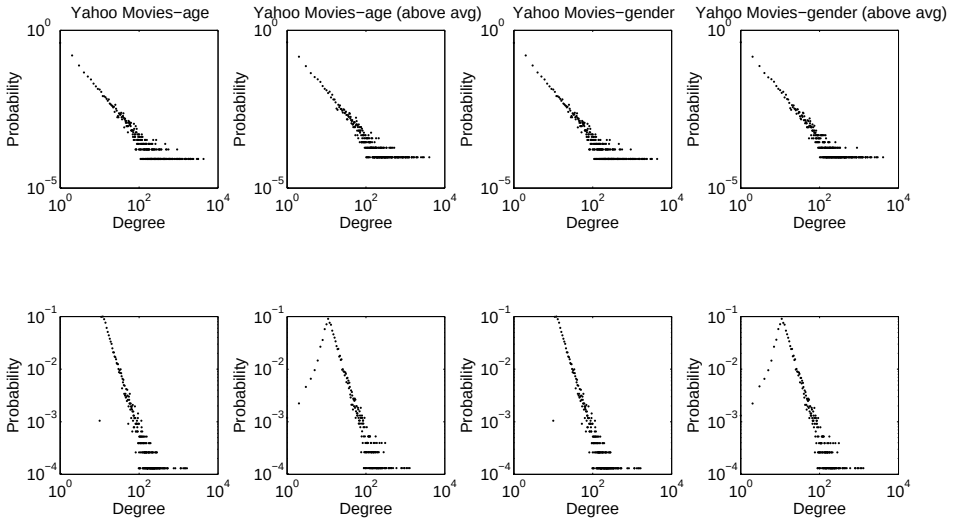


Fig. 7: Degree distributions of the top nodes (upper row) and bottom nodes (bottom row) for different datasets.

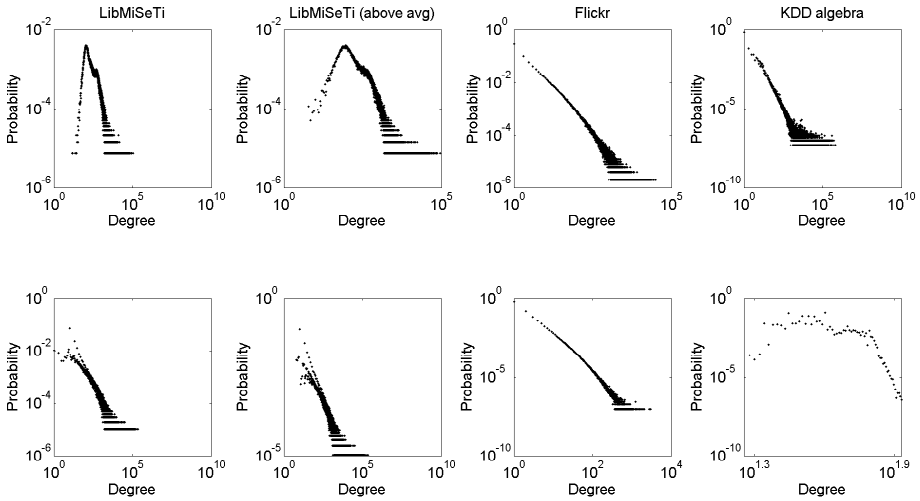


Fig. 8: Degree distributions of the top nodes (upper row) and bottom nodes (bottom row) for different datasets.

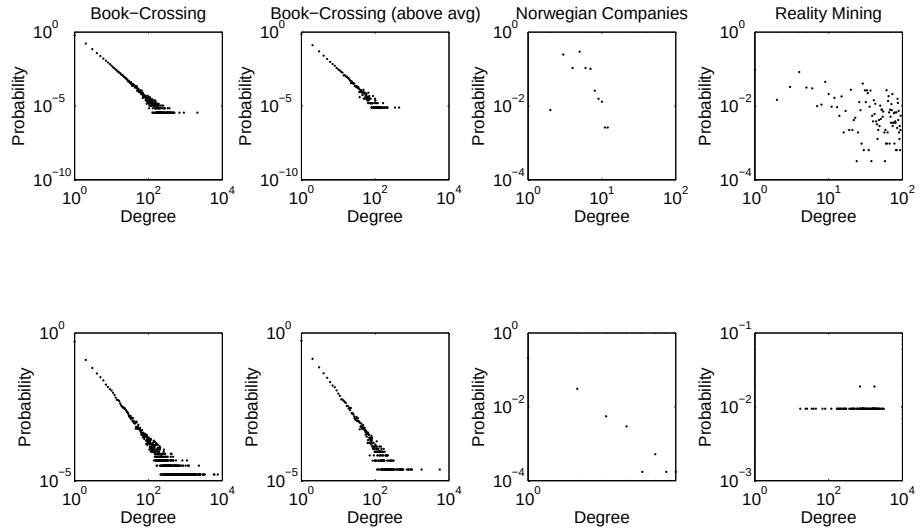


Fig. 9: Degree distributions of the top nodes (upper row) and bottom nodes (bottom row) for different datasets.

4 Results

4.1 Predictive performance

In this section we present the predictive performance results for the techniques used by the three-stage framework (see Tables 6-11 and Tables 14-15 in Appendix A[11]). Based on the AUC, for every dataset we rank the performance of each combination of techniques from the different stages. To create the final rankings, we average the results over all datasets. We also calculate a ranking for each of the techniques individually, by averaging the results over all the combinations where this technique is present. A problem that arises is that some combination schemes are not scalable to large datasets. Several aggregation functions, including the Jaccard, cosine similarity and the maximum function are not scalable to the larger datasets (Flickr and larger). Also a combination of these aggregation functions and the beta function takes very long time to fit for datasets over 100,000 nodes. In cases when a method is not able to provide scores, we both (i) do not consider it when calculating the average ranking (see Tables 7, 9, 11, 15) and (ii) we assign it the lowest ranking (see Tables 6, 8, 10, 14). The combinations that are not significantly worse from the best method (underlined) at a 5% significance level are emphasized in bold. The combinations that are significantly worse at 5% but not at 1% significance level are shown in italic. The combinations that are significantly worse at 1% significance level are shown in regular font.

Tables 14 and 15 present the results for all the combinations of methods. From this, we can observe that the highest ranked combinations that perform very well over all the datasets are the tangens hyperbolicum function, combined with the cosine or sum of shared nodes aggregation functions and the wvRN. Furthermore, there are also a few alternatives that provide comparable results to these top ranked combinations. If we take a closer look at Tables 14 and 15, we can see that generally combinations that include the cosine or sum of shared nodes aggregation functions together with the tangens hypebolicum, inverse degree and occasionally the beta function or the inverse frequency provide very good and close results when combined with any of the relational classifiers. In order to assess the quality of the functions better we continue this section by examining the performance of the functions more carefully, by looking at each of the three stages separately.

4.1.1 Predictive performance of the top node weight functions

The results regarding the top node functions are summarized in Tables 6 and 7, with the tangens hyperbolicum and the inverse degree (both similar in shape, see Figure 2) providing the best average scores across all domains. These results are aggregated over all the combinations of methods that include the specific top node function, including very poor combinations. As can be seen

[11] The complete results table can be downloaded at www.applieddatamining.com.

from the complete rankings in Tables 14-15, specific combinations that include the top node function can have very strong performances. However, the overall rankings still get diluted by the weaker combinations—for example, ones containing the zero-one aggregation function. We should be careful when interpreting these results and do not simply discard top node methods that provide weaker average results over all domains. If we take a closer look at Table 16, which presents the best combinations of techniques per dataset, we can see indeed that combinations including the beta distribution or the likelihood ratio can be the most appropriate choice for some datasets. We will come back to this issue later in the section.

The beta distribution is significantly different from the other functions, as it can adapt its shape to the specific dataset. Depending on the parameters α and β, the shape of the function can change as shown in Figure 10 (a). In Figure 10 (b), we present the shapes of the optimal beta function for the Yahoo Movies dataset with target variable age and gender and for the BookCrossing dataset, where we predict the age of the readers. The shapes correspond strongly to the intuition that top nodes with smaller degree are more discriminative and therefore should have higher weights. This is also valid for the other datasets, where the optimal α and β are listed in Table 12. The only exception is the Flickr dataset, where the performance of the beta distribution is comparable to adding no weight on the top nodes.

Method	Avg Ranking
tanh	**69.7**
inverse degree	**70.6**
inverse frequency	*72.3*
beta distribution	*80.1*
adamic and adar	84.3
w=1	84.5
delta	88.9
likelihood ratio	94.0

Table 6: Ranking per top node function on all datasets. In case where a combination of methods is not able to run on a dataset, it gets the lowest ranking for the specific dataset.

4.1.2 Predictive performance of the aggregation functions

Tables 8 and 9 show the aggregated results per aggregation function, with the cosine function and the sum of shared nodes as the most suitable methods. 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

Method	Avg Ranking
tanh	**66.9**
inverse degree	**67.9**
inverse frequency	*69.7*
beta distribution	*76.4*
adamic and adar	82.5
w=1	82.9
delta	87.7
likelihood ratio	93.0

Table 7: Ranking per top node function on all datasets. In case where a combination of methods is not able to run on a dataset, it gets no ranking for the specific dataset.

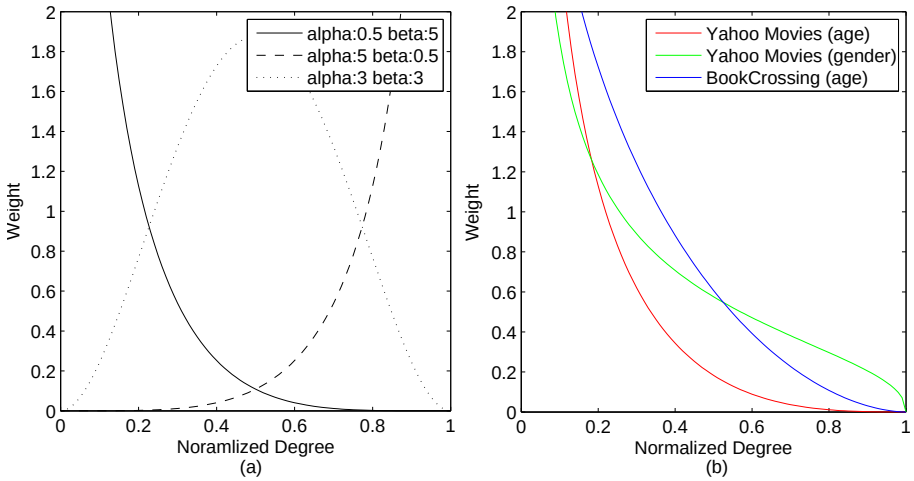


Fig. 10: Shape of the beta function for (a) some specific parameters (b) the Yahoo Movies and BookCrossing datasets obtained on a validation set.

one of the nodes has a lot of links. If we take again the example of people visiting locations, with person A visiting 5 different locations; person B visiting these 5 locations and 10 more; while person C visited the same 5 locations and 100 more. 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, since the odds are very small that A and C visited the same 5 locations by chance. 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. On the other hand, summing the weights of the shared top nodes performs very well as an aggregation function.

Method	Avg Ranking
cosine function	**59.7**
sum of shared nodes	**62.0**
jaccard	74.4
max	90.7
zero - one	115.9

Table 8: Ranking per aggregation function, on datasets where all methods are able to run.

Method	Avg Ranking
cosine function	**55.8**
sum of shared nodes	**58.9**
jaccard	70.4
max	89.5
zero - one	117.2

Table 9: Ranking per aggregation function. In case where a combination of methods is not able to run on a dataset, it gets no ranking for the specific dataset.

4.1.3 Predictive performance of the relational classifiers

Tables 10 and 11 present 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 over the target values in the projection. As an example of postive relational autocorrelation [29], if I like the same movies as someone else, we likely are of the same gender. Yet, the opposite can be true as well. In the Norwegian companies dataset, a man is more likely to be in a board with a female and vice verca. For this reason, the wvRN here yields a pathological average AUC (over all combination schemes) of only 0.2728. This substantially hurts the wvRN average scores, as AUCs systematically below 0.5 can be "flipped" to sometimes strong AUCs. Therefore, this result requires some extra explanation.

Norway is one of the leading countries that enforces equal gender representation in companies' boards [58], which results in the companies (top nodes) being connected to almost the same number of males and females. In Figure 11, we use entropy to show the heterogeneity of the nodes' neighbors in the projection. Entropy represents the class imbalance [57] and it has a value of zero when all the neighbors of a node have the same class and a highest value of one when there is an equal number of nodes from both classes. The results are averaged over nodes with the same number of neighbors. As a comparison, a typical dataset where wvRN performs equal or even better (as is the case with the MovieLens dataset where we predict whether a movie's genre is horror or not) has much lower entropy (see Figure 11). Note that except for the entropy, the weights of the links also have impact on the prediction performance of wvRN. However, for cases where the class distribution of a

node's neighbors is approximately 0.5, cross-validation can cause pathologies in machine-learning evaluations. Consider the following.

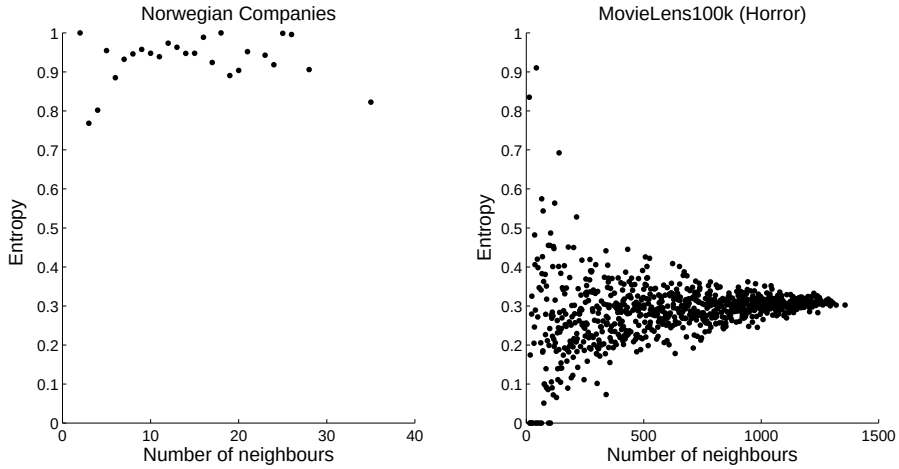


Fig. 11: Average entropy per number of neighbors in the projection, for the Norwegian Companies dataset (left) and the MovieLens dataset with target variable Horror genre (right).

As can be seen from Figure 9, most (83.6%) of the directors (bottom nodes) are members of the board of only one company (top nodes) and most companies (71%) have only up to 5 board members. This creates many small disconnected components in the bigraph, like the ones shown on Figure 12. 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. Therefore, on the projections of these structures it will likely predict the opposite class. For example, consider a leave-one-out-style evaluation. In case (a), a female member will be connected to only one male member in the projection, hence wvRN will predict the wrong class. In the other cases, the links' weights in the projection will be equal, leading to wvRN predicting the majority opposite class or giving score of 0.5 when the remaining classes are balanced. This is denoted with question mark in the predictions under the nodes. 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

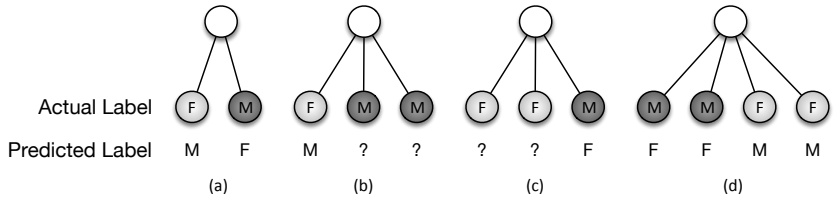


Fig. 12: Bigraph structures of companies and board members. The node letter presents the actual gender of the board member and below is the predicted gender by the wvRN.

reference vectors that take into account how the training nodes are connected to the opposite class, yielding an average AUC of 0.6997.

Method	Avg Ranking
wvRN	**74.1**
nlb	**77.7**
cdRN	78.8
nlb 100	91.5

Table 10: Ranking per relational classifier, on datasets where all methods are able to run.

Method	Avg Ranking
wvRN	**73.2**
nlb	**76.5**
cdRN	77.8
nlb 100	86.0

Table 11: Ranking per relational classifier. In case where a combination of methods is not able to run on a dataset, it gets no ranking for the specific dataset.

As a benchmark against which to compare the network projection methods, we apply a linear SVM. From Tables 14 and 15 can be seen that the performance of the SVM compared to the other combinations, is only average. Additionally the SVM was not able to scale up to the big KDDa and KDDb datasets.

The SW-transformation combines the best relational classifier wvRN and one of the best performing aggregation functions, sum of shared nodes into a linear model that is able to scale well to big datasets. It is the only technique in the study that scales well (or at all) to the biggest datasets KDDa with 8 million $\times$ 20 million nodes and KDDb with 19 million $\times$ 30 million nodes.

4.2 Run-time performance

In this section we examine the run-time performance of the different techniques from the three stages. We start by comparing the average durations of each of the techniques over the datasets. For this, we only consider the datasets with less than 100,000 nodes since not all the methods are able to run on the larger datasets. In Appendix, we give the detailed results of the averaged durations for each of the top node functions and aggregation functions per relational learner (see Figures 14-17). In short, for each of the relational learners the maximum and the Jaccard aggregation function have the longest durations, especially when combined with the beta top node function. The beta top node function takes longest to run due to the tuning procedure. In our setting we tune the beta function with a grid search on three levels. In Table 12 we can see how the AUC improves on different levels of the grid search for several datasets. For each dataset the best chosen parameters of the specific level are shown with the corresponding AUC value. One can observe that usually, the optimal α and β parameters give a shape of the curve such that the nodes with a smaller degree receive a higher weight. For example, in the first level for most of the datasets the optimal α is 0.1 and β is 3.1, which is exactly this shape of the curve. This may be included in the grid search for time improvement, to only take into account the parameters which give this kind of shape. Moreover, the grid search can be reduced to only one or two levels, which results in limited performance decrease (see Table 12). We also examined tuning the α and β parameters on a smaller sample of the training data. In Figure 13 we can see that the predictive performance for most of the datasets are stable even when the beta function is tuned on a much smaller subset of 1000 data points. This speeds up the tuning procedure up to several times, especially for the larger datasets. The rest of the top node functions are faster than the beta function, with similar durations.

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

Table 12: Beta grid search on three levels with the optimal α and β parameters, as well as the coresponding AUC per level. The aggregation function used is the sum of shared nodes in combination with the wvRN relational classifier.

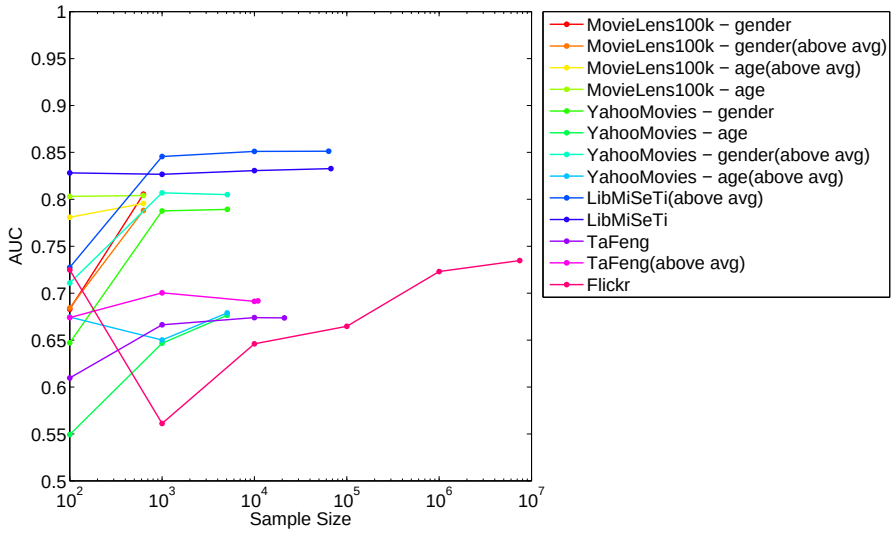


Fig. 13: Predictive performance of the beta function in combination with SW-transformation when the parameters are tuned on a sample of the training data and trained on the full training data

The 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 Figure 18 in Appendix). In our experiments even a sample of less 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 tuned much faster. The time advantage is clearly larger on bigger datasets, like for example the BookCrossing dataset. However, when the class-label autocorrelation is uncertain and the training time is an issue, it may be better simply to use the cdRN classifier, which is fast and whose performance is quite robust to different sorts of relational autocorrelation.

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. For example for the biggest dataset we used, KDDb with dimensions of around 19 million $\times$ 30 million, the SW-transformation with a regular top node function that does not require tuning (i.e., not the beta function) takes around 9 minutes to finish. This dataset did not fit in memory, so we performed batch processing directly from disk. For the KDDa dataset (dimensions of around 8 million $\times$ 20 million) and the Flickr dataset (11 million $\times$ half a million) the SW-transformation takes 5 minutes and less than a minute respectively. This represents a substantial scalability and time improvement over the regular sum of shared nodes and wvRN im-

plementations, even with batch processing. The clear time advantage of the SW-transformation over each dataset can be seen in Figure 19 (in Appendix), where the average time needed for the regular sum of shared nodes and wvRN over the datasets is 65.4 seconds and the SW-transformation needs only 0.5478 seconds on average.

4.3 Linear Models

The SW-transformation is a fast method that scales easily to big datasets. An additional important aspect of the SW-transformation is the comprehensibility of the linear models it provides. For each top node we can get a weight (coefficient) of the impact it has for the target variable. This makes it understandable for humans and with a manual check can be inspected whether the model makes sense. Comprehensibility is needed, and even mandatory, in many domains where the decisions of the classifier need to be clearly explained and validated before the classifier can be used [21, 22, 42, 44]. In Table 13 we make an additional verification of the results, by examining the top nodes' weights using the combination of the beta and SW-transformation. The combination assigns weight to the top nodes, which reflects how discriminative the node is for the target variable. We list the top 20 ranked instances with the highest scores/coefficient in Table 13 for the following cases: (a) gender and (b) age prediction for the Yahoo Movies bigraph of users ranking movies and (c) age prediction for the BookCrossing bigraph of users rating books. If we take a close look at the top ranked nodes in the Table, we can see that the rankings are indeed intuitive. As an example in Table 13(a) where we rank the top nodes (movies) and our goal is to predict whether a user is male, the movies with the highest scores seem to be targeted to a male audience. Movies like *Terminator, X-man, Kill Bill* and etc. can be found on the list. Furthermore, Table 13(b) presents the rankings for the Yahoo Movies targeted to a younger audience (the target variable is age). Here we can see some teenage movies like *American Pie*, *Scary Movie* or horror movies like *The Texas Chainsaw Massacre* which are usually preferred by younger people. The books that discriminate young people best are given in Table 13(c), with titles like *Harry Potter* and *A Walk to remember*. The optimal shapes of the beta function for all the three datasets are shown in Figure 10(b).

Rank	Yahoo Movies (gender)	Yahoo Movies (age)	BookCrossing (age)
1.	The Matrix Reloaded (2003)	Ocean's Eleven (2001)	The Catcher in the Rye J.D. Salinger
2.	Terminator 3: Rise of the Machines (2003)	The Ring (2002)	Harry Potter and the Goblet of Fire by J. K. Rowling
3.	The Hulk (2003)	Scary Movie 3 (2003)	1984 by George Orwell
4.	X2: X-Men United (2003)	American Pie 2 (2001)	The Fellowship of the Ring by J.R.R. Tolkien
5.	Bad Boys II (2003)	American Pie (1999)	Confessions of a Shopaholic by Sophie Kinsella
6.	The Lord of the Rings: The Two Towers (2002)	Pulp Fiction (1994)	The Life and Times of the Wicked Witch of the West by G. Maguire
7.	The Italian Job (2003)	The Texas Chainsaw Massacre (2003)	To Kill a Mockingbird by Harper Lee
8.	The Matrix Revolutions (2003)	Austin Powers in Goldmember (2002)	Interview with the Vampire by Anne Rice
9.	Bruce Almighty (2003)	Terminator 2 - Judgment Day (1991)	Harry Potter and the Sorcerer's Stone by J. K. Rowling
10.	28 Days Later (2003)	Gladiator (2000)	The Vampire Lestat by Anne Rice
11.	Kill Bill Vol. 1 (2003)	The Lizzie McGuire Movie (2003)	A Walk to Remember by Nicholas Sparks
12.	American Wedding (2003)	Phone Booth (2003)	Harry Potter and the Order of the Phoenix by J. K. Rowling
13.	Freddy vs. Jason (2003)	Uptown Girls (2003)	Fast Food Nation: The Dark Side of the American Meal by E. Schlosser
14.	S.W.A.T. (2003)	How to Deal (2003)	Lord of the Flies by William Gerald Golding
15.	The Matrix (1999)	Signs (2002)	She's Come Undone (Oprah's Book Club) by Wally Lamb
16.	The League of Extraordinary Gentlemen (2003)	Daredevil (2003)	The Girls' Guide to Hunting and Fishing by Melissa Bank
17.	The Lord of the Rings: The Fellowship of the Ring(2001)	X-Men (2000)	Harry Potter and the Prisoner of Azkaban by J. K. Rowling
18.	Terminator 2 - Judgment Day (1991)	The Matrix (1999)	Girl, Interrupted by Susanna Kaysen
19.	Seabiscuit (2003)	A Walk to Remember (2002)	Animal Farm by George Orwell
20.	Star Wars (1977)	Anger Management (2003)	Jurassic Park by Michael Crichton

Table 13: Top nodes with highest coefficient in the linear model of the SW-transformation in combination with the beta function. 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 (c) young when predicting *age* for the BookCrossing bigraph.

4.4 Summary of the results

We have provided an extensive empirical study of the predictive and run-time performance of a number of choices for the three stages over a large collection of bipartite datasets. The results indicate that 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 from the three stages that generally provide good results. In the first stage of the framework, two functions, namely the tangens hyperbolicum and the inverse degree dominated in the results. However, since they both provide very similar results one could reduce their choice only to the former one. As for the aggregation functions, the best performing functions are the cosine and the sum of shared nodes. The latter one is favorable since it can be combined with the wvRN (SW-transformation) to scale easily and fast to very large datasets. The wvRN is obviously an appropriate choice for problems that exhibit network assortativity; therefore, it would be well also to consider the learning-based classifiers (especially nLB). Although they provide similar results when we consider datasets that exhibit assortativity, the nLB is more powerful and can capture more complex patterns, as in the case of the Norwegian companies dataset. The nLB100 is a trade-off between speed and accuracy, a much faster variant of the nLB classifier (see Figure 18), but with weaker predictive performance.

In Figures 20 - 27 we plot the predictive and run-time performance of all the combinations of methods on each dataset individually. The combinations of the methods we recommend (tangens hyperbolicum, cosine similarity, sum of shared nodes, wvRN and nLB100) are denoted with red squares. In most cases they are among the fastest and most accurate combinations, with less than 5% AUC difference from the combination that performs best. This is not the case for several MovieLens datasets where we predict a genre like Fantasy, Film-Noir, War or Mystery. What is specific about these datasets is that they are among the most skewed datasets with only 1.25%-6.5% positive labels. Based on the results in Table 16, we can conclude that the beta distribution performs very well for this type of datasets, especially when combined with Jaccard. Unfortunately this combination does not scale well because of the beta tuning procedure and therefore is applicable to only smaller datasets.

Other datasets where our recommendations have weak results are the Reality Mining datasets, where people are connected to the places they visited. The problem here is that most of the people have visited the same places, which makes the projection (almost) fully connected. Moreover, since the places are very popular and visited by almost all the people, on average a person shares all the locations he/she visited with 50% of the other people. This makes the locatioins not very discriminative. For such datasets, where the projections are fully (or almost fully) connected, traditional classification approaches (discussed in Section 1.2), can be better alternatives.

5 Related work on bigraph data analysis

Literature regarding bigraphs has so far been focused on measuring descriptive statistics, link prediction for recommender systems and clustering. There are also many unigraph studies that essentially use unigraph projections of bipartite data. For example the datasets used to create networks based on scientific collaborations [37], co-occurrence of companies in text documents [41], web page co-citation [39], movies linked if they share the same production company or crew [41,40], book co-purchase [19] and so on, in the unigraph literature are in fact bigraph projections. To our knowledge, different projection methods were not compared to maximize performance on the associated task.

There has been some initial research that explores the properties of bigraphs and extending *global network metrics* of unipartite graphs to the bigraph case. Centrality measures, which determine the varying importance of nodes within the graph, like betweenness, degree, closeness and eigenvector centralities [15,7,6], as well as the clustering coefficient [38,49,36,56] have been extended to the bipartite case. Newman [47,48] takes a different approach to measuring bipartite network statistics of authors and papers, by considering the bottom nodes projection of authors which are connected if they have at least one paper in common.

A second research area concerns *link prediction* in bigraphs, which tries to predict the links that will appear at some future time, given the bigraph structure at the present time. For example, Huang et al. [27] use several linkage measures, based on the topology of the bigraph, which predict for each bottom node the top links with the highest measure score. Benchettara et al. [5] describe pairs of unlinked nodes with different topological attributes and consider the pairs as positive instances if a link is created between the nodes at some time in the future, otherwise as negative instances. They apply supervised learning in order to predict future links. A study by Allali et al. [2] considers prediction of only internal links or links which do not change the unipartite projections when added to the original bigraph. Here the links in the original bigraph are predicted if the links of the weighted projection, which are induced by these internal links, have a weight higher than a given threshold. Kunegis et al. [34] propose algebraic methods for link prediction and Zhou et al. [64] consider link prediction on the weighted unipartite projection of the bigraph. Link prediction has been applied in previous literature to recommender systems for online music shops [5], online book stores [27], movies recommendation [64], etc.

Bigraph *clustering* has also been explored in literature. Forunato [18] gives a good overview of clustering techniques for graphs including bipartite graphs. Borgatti and Halgin [7] use standard unipartite techniques for clustering on an extended bigraph matrix, which contains in both dimensions all the nodes from the two sets of the bigraph. Blockmodeling is another popular clustering method for bigraphs [7,11]. Zha et al. [63] propose a method for partitioning the bigraph by minimizing a normalized sum of the edge weights between pairs of nodes which do not belong to the same partition and are of different

node type. Barber [4] extends the measure of modularity to the bipartite case, which calculates the degree to which nodes cluster into communities in respect to the null model, by defining a new null model appropriate for bigraphs. Sun et al. [59] employ a random walk for identifying similar nodes in bigraphs. More specifically, for each bottom node they calculate a relevance score as a measure of similarity to every other bottom node as the number of times a node has been visited during the random walk. They also use these relevance scores to detect anomalous top nodes, by calculating a normality score as a mean over the relevance scores between the neighboring bottom nodes. Top nodes with low normality scores connect nodes that belong to different communities. Clustering has been used for discovering community structures in bigraphs of companies and board directors [4], women attending events [4,11], supreme court voting [11], finding similar users or genres of music from listeners and music groups bipartite graph [35], clustering documents based on the occurring terms [63], looking for similar actors based on the movies they have played in [59], similar authors based on the papers they collaborated [59], conferences based on the authors that published, etc.

Projecting bigraphs into unigraphs results in loss of information [36]. Assigning weights to the projected unigraph preserves some information of the underlying bigraph. In this paper 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 over the unigraph projection. As such, we have an objective function that determines to what extent the predictive information present in the bigraph is also contained in the projected unigraph. There exist some studies in the literature that explore this problem of how to most accurately represent the bigraph with a transformation to unipartite graph. For example, Zweig and Kaufman [66] take the approach of connecting nodes in the projection if they have a much higher number of occurrences of motifs (recurrent and statistically significant sub-graphs or patterns) compared to the random graph model of the given bigraph. For the assessment of the quality of their method they use the Netflix movies-users dataset; more specifically they consider the neighbors of nodes which are part of some television serial. Given a good projection method, the neighbors should be similar nodes, for example, other parts of the serial. Furtehermore, Zou et al. [64] 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 initial weight to the bottom nodes in the bigraph. In the next step, these weights are equally distributed over the neighboring top nodes. Finally, the weights are once more distributed, this time from the top to the bottom nodes. The result is a linear equation for each bottom node, where the coefficients signify the link weight in the projection with direction from the specific bottom node. The authors use the weighted unigraph projection for movie recommendations and compare their link prediction performance to other recommendation algorithms. Recently, Gupte and Eliassi-Rad [24] considered a wide range of

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

The philosophy of this paper is that the best projection is the one that maximizes performance for a target task. Thus, we should have a framework for systematically exploring the design space. We have proposed various options that can be mixed and matched. Presumably there are others as well that would fit within this framework, as well as alternative possible frameworks.

In addition to the unimodal graphs discussed above that are really bigraph projections, node classification where the bigraph is considered explicitly to our knowledge has so far been limited to a few case-specific studies: predicting interest in financial products from a bigraph of consumers and merchants [43], predicting brand interest from a bigraph of browsers visiting websites [51] and a bigraph of people visiting locations [54]. Another exception is the work of Perlich and Provost [50], who consider classification for datasets with high-dimensional categorical attributes. These attributes can be for example locations which a person visited, identifiers of previously bought books (or other products) by customers and etc. If we consider the persons as bottom nodes and the products or locations as top nodes, it is clear that their approach aggregates the bigraph (or more generally the k-partite graph) information, by applying aggregation operators. This creates new features, which combined with the structured data about the persons, are used by a traditional propositional method. An interesting avenue for future work would be to combine this additional information on the bigraph into the projected unipartite network.

6 Conclusion

Bigraph datasets are an intuitive way to represent relational, behavioral and transactional data. The modular three-stage projection framework to leverage such data for node classification has the flexibility to compose a variety of classification methods. The comparison with support-vector machines shows encouraging results: the linear SVM has only average performance when compared to the other combinations of methods (even linear ones) and the popular implementation does not scale to the largest datasets KDDa and KDDb. In our experiments the tangens hyperbolicum top node function performs best. The cosine aggregation function, followed by the sum of the shared nodes in combination with the wvRN relational classifier gives the best results. A combination of the latter two is considered by the SW-transformation, a very fast and scalable linear technique that is able to scale to datasets of up to millions of nodes easily, while providing a comprehensible model.

We do not claim to have found the best combination of elements. Rather we argue that within this framework many possibilities exist. As more and more behavioral datasets become available, the prediction over nodes in the corresponding bigraphs will likely see a similar increase in interest. However, given its speed, solid predictive performance, and comprehensibility, we suggest

that the SW-transformation provides a very solid baseline method for future studies of methods for predictive modeling with (sparse) bigraph data.

Although, as described earlier, prior studies individually have applied projection to bigraph data, to our knowledge this is the first general study of predictive modeling on bigraph data using projection. Presumably future work could provide significant advances. Moreover, in this paper we did not address cases where the original bigraph is weighted, nor the important situation where bottom nodes have local information, nor where other features of the top nodes (aggregated) can provide predictive information or where the specific top nodes are discriminative [50]. Hopefully this paper's framework provides a useful stepping stone to such work.

References

1. Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211 – 230, 2003.
2. Oussama Allali, Clémence Magnien, and Matthieu Latapy. Link prediction in bipartite graphs using internal links and weighted projection. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 936 –941, april 2011.
3. Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.
4. Michael J Barber. Modularity and community detection in bipartite networks. *Physical Review E*, 76(6):066102, 2007.
5. Nesserine Benchettara, Rushed Kanawati, and Celine Rouveirol. Supervised machine learning applied to link prediction in bipartite social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 326–330. IEEE, 2010.
6. Stephen P Borgatti and Martin G Everett. Network analysis of 2-mode data. *Social networks*, 19(3):243–269, 1997.
7. Stephen P Borgatti and Daniel S Halgin. Analyzing affiliation networks. *The Sage handbook of social network analysis*, pages 417–433, 2011.
8. Lukas Brozovsky and Vaclav Petricek. Recommender system for online dating service. In *Proceedings of Conference Znalosti 2007*, Ostrava, 2007. VSB.
9. Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 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*, KDD '11, pages 1082–1090, New York, NY, USA, 2011. ACM.
10. Luc Devroye. *A probabilistic theory of pattern recognition*, volume 31. Springer, 1996.
11. Patrick Doreian, Vladimir Batagelj, and Anuška Ferligoj. Generalized blockmodeling of two-mode network data. *Social Networks*, 26(1):29–53, 2004.
12. Nathan Eagle and Alex Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.
13. David Easley and Jon Kleinberg. *Networks, crowds, and markets*, volume 8. Cambridge Univ Press, 2010.
14. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.
15. Katherine Faust. Centrality in affiliation networks. *Social networks*, 19(2):157–191, 1997.
16. Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

17. Catherine Forbes, Merran Evans, Nicholas Hastings, and Brian Peacock. *Statistical distributions*. John Wiley & Sons, 2011.
18. Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.
19. Brian Gallagher, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 256–264. ACM, 2008.
20. Sharad Goel, Jake M Hofman, and M Irmak Sirer. Who does what on the web: A large-scale study of browsing behavior. In *ICWSM*, 2012.
21. M Sinan Gönül, Dilek Önkal, and Michael Lawrence. The effects of structural characteristics of explanations on use of a dss. *Decision Support Systems*, 42(3):1481–1493, 2006.
22. Shirley Gregor and Izak Benbasat. Explanations from intelligent systems: Theoretical foundations and implications for practice. *MIS quarterly*, pages 497–530, 1999.
23. Jean-Loup Guillaume and Matthieu Latapy. Bipartite graphs as models of complex networks. *Physica A: Statistical Mechanics and its Applications*, 371(2):795–813, 2006.
24. Mangesh Gupte and Tina Eliassi-Rad. Measuring tie strength in implicit social networks. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 109–118. ACM, 2012.
25. Jian Hu, Hua-Jun Zeng, Hua Li, Cheng Niu, and Zheng Chen. Demographic prediction based on user's browsing behavior. In *Proceedings of the 16th international conference on World Wide Web*, pages 151–160. ACM, 2007.
26. Han-Shen Huang, Koung-Lung Lin, Jane Yung-jen Hsu, and Chun-Nan Hsu. Item-triggered recommendation for identifying potential customers of cold sellers in supermarkets. *Proceedings of Beyond Personalization*, pages 37–42, 2005.
27. Zan Huang, Xin Li, and Hsinchun Chen. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 141–142. ACM, 2005.
28. Ramon Ferrer i Cancho and Richard V Solé. The small world of human language. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1482):2261–2265, 2001.
29. David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML*, volume 2, pages 259–266. Citeseer, 2002.
30. Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.
31. Enric Junqué de Fortuny, David Martens, and Foster Provost. Predictive modeling with big data: Is bigger really better? *Big Data*, 1(4):215–226, 2013.
32. Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
33. Michal Kosinski, David Stillwell, and Thore Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805, 2013.
34. Jérôme Kunegis, Ernesto W De Luca, and Sahin Albayrak. The link prediction problem in bipartite networks. In *Computational intelligence for knowledge-based systems design*, pages 380–389. Springer, 2010.
35. Renaud Lambiotte and Marcel Ausloos. Uncovering collective listening habits and music genres in bipartite networks. *Physical Review E*, 72(6):066107, 2005.
36. Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. Basic notations for the analysis of large two - mode networks. *Social Networks*, 30:31–48, 2008.
37. David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.
38. Pedro G Lind, Marta C Gonzalez, and Hans J Herrmann. Cycles and clustering in bipartite networks. *Physical review E*, 72(5):056127, 2005.
39. Qing Lu and Lise Getoor. Link-based classification. In *ICML*, volume 3, pages 496–503, 2003.

40. Sofus A Macskassy and Foster Provost. A simple relational classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003*, 2003.
41. Sofus A. Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *J. Mach. Learn. Res.*, 8:935–983, May 2007.
42. David Martens, Bart Baesens, Tony Van Gestel, and Jan Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European journal of operational research*, 183(3):1466–1476, 2007.
43. David Martens and Foster Provost. Pseudo-social network targeting from consumer transaction data. Working paper CeDER-11-05, New York University - Stern School of Business, 2011.
44. David Martens and Foster Provost. Explaining data-driven document classifications. *MIS Quarterly*, 38(1):73–100, 2014.
45. David Martens, Foster Provost, Jessica Clark, and Enric Junque de Fortuny. Mining fine-grained consumer payment data to improve targeted marketing. Working paper, New York University - Stern School of Business, 2013.
46. Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.
47. Mark EJ Newman. Scientific collaboration networks. i. network construction and fundamental results. *Physical review E*, 64(1):016131, 2001.
48. Mark EJ Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical review E*, 64(1):016132, 2001.
49. Tore Opsahl. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*, 2011.
50. Claudia Perlich and Foster Provost. Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning*, 62(1-2):65–105, 2006.
51. Foster Provost, Brian Dalessandro, Rod Hook, Xiaohan Zhang, and Alan Murray. 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*, pages 707–716. ACM, 2009.
52. Foster Provost and Tom Fawcett. *Data Science for Business: What you need to know about data mining and data-analytic thinking.* " O'Reilly Media, Inc.", 2013.
53. Foster Provost and Venkateswarlu Kolluri. A survey of methods for scaling up inductive algorithms. *Data mining and knowledge discovery*, 3(2):131–169, 1999.
54. Foster Provost, David Martens, and Alan Murray. Geo-social network advertising. In *2012 Winter Conference on Business Intelligence*, 2012.
55. Troy Raeder, Ori Stitelman, Brian Dalessandro, Claudia Perlich, and Foster Provost. Design principles of massive, robust prediction systems. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1357–1365. ACM, 2012.
56. Garry Robins and Malcolm Alexander. Small worlds among interlocking directors: Network structure and distance in bipartite graphs. *Computational & Mathematical Organization Theory*, 10(1):69–94, 2004.
57. Alfred Rrnyi. On measures of entropy and information. In *Fourth Berkeley Symposium on Mathematical Statistics and Probability*, pages 547–561, 1961.
58. Cathrine Seierstad and Tore Opsahl. 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, 2011.
59. Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.
60. Ingmar Weber, Venkata Rama Kiran Garimella, and Erik Borra. Inferring audience partisanship for youtube videos. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 43–44. International World Wide Web Conferences Steering Committee, 2013.
61. Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, pages 194–205, 1998.

62. Hsiang-Fu Yu, Hung-Yi Lo, Hsun-Ping Hsieh, Jing-Kai Lou, Todd G McKenzie, Jung-Wei Chou, Po-Han Chung, Chia-Hua Ho, Chun-Fu Chang, Yin-Hsuan Wei, et al. Feature engineering and classifier ensemble for kdd cup 2010. In *Proceedings of the KDD Cup 2010 Workshop*, pages 1–16, 2010.
63. Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 25–32. ACM, 2001.
64. Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4):046115, 2007.
65. Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.
66. Katharina Anna Zweig and Michael Kaufmann. A systematic approach to the one-mode projection of bipartite graphs. *Social Network Analysis and Mining*, 1(3):187–218, 2011.

A Appendix: Results Tables

Table 14: Average ranking for all the combinations of techniques. In case where a combination of techniques is not able to run on a dataset, it gets lowest ranking for the specific dataset.

Top node weight	Aggregation function	Classifier	Avg Ranking
tanh	sum of shared nodes	wvRN	**41.1**
inverse degree	sum of shared nodes	wvRN	*41.9*
tanh	cosine function	wvRN	**42.0**
inverse degree	cosine function	wvRN	**42.5**
inverse frequency	sum of shared nodes	wvRN	*44.0*
tanh	cosine function	cdRN	**44.2**
tanh	cosine function	nlb	**44.6**
inverse degree	cosine function	cdRN	**45.0**
inverse degree	cosine function	nlb	**45.2**
inverse frequency	cosine function	wvRN	**45.7**
inverse frequency	cosine function	cdRN	**46.0**
tanh	sum of shared nodes	nlb	**46.2**
beta distribution	sum of shared nodes	wvRN	**46.5**
inverse frequency	cosine function	nlb	**47.0**
tanh	sum of shared nodes	cdRN	**47.5**
inverse degree	sum of shared nodes	cdRN	**48.7**
inverse degree	sum of shared nodes	nlb	**48.8**
inverse frequency	sum of shared nodes	nlb	*51.3*
w=1	sum of shared nodes	wvRN	53.2
inverse frequency	sum of shared nodes	cdRN	*53.8*
tanh	jaccard	wvRN	**54.5**
w=1	cosine function	cdRN	**55.9**
inverse frequency	jaccard	wvRN	**55.9**
inverse frequency	jaccard	cdRN	**56.1**
adamic and adar	sum of shared nodes	wvRN	56.3
w=1	cosine function	wvRN	**56.6**
inverse degree	jaccard	wvRN	*56.8*
tanh	sum of shared nodes	nlb 100	*57.8*
tanh	jaccard	cdRN	**57.9**
tanh	cosine function	nlb 100	**58.1**
inverse degree	cosine function	nlb 100	**58.2**
inverse degree	sum of shared nodes	nlb 100	58.8

Continued on next page

Table 14 – *Continued from previous page*

Top node weight	Aggregation function	Classifier	Avg Ranking
beta distribution	cosine function	wvRN	**59.2**
inverse degree	jaccard	cdRN	**59.5**
inverse frequency	sum of shared nodes	nlb 100	59.6
beta distribution	cosine function	nlb	**59.7**
w=1	cosine function	nlb	**59.7**
inverse frequency	cosine function	nlb 100	*59.8*
adamic and adar	cosine function	wvRN	**59.9**
w=1	sum of shared nodes	nlb	60.0
beta distribution	cosine function	cdRN	**60.2**
adamic and adar	cosine function	cdRN	**60.7**
adamic and adar	cosine function	nlb	*62.5*
adamic and adar	sum of shared nodes	nlb	63.0
delta	cosine function	wvRN	63.2
w=1	jaccard	wvRN	**63.3**
delta	sum of shared nodes	wvRN	63.5
w=1	sum of shared nodes	cdRN	63.5
beta distribution	sum of shared nodes	cdRN	*64.0*
beta distribution	jaccard	nlb	*64.2*
beta distribution	max	wvRN	*64.4*
w=1	jaccard	cdRN	**64.9**
tanh	jaccard	nlb	*65.0*
inverse frequency	jaccard	nlb	*65.3*
beta distribution	max	nlb	*65.4*
delta	cosine function	cdRN	*65.6*
adamic and adar	sum of shared nodes	cdRN	66.2
delta	cosine function	nlb	66.6
tanh	max	wvRN	67.2
inverse degree	jaccard	nlb	*67.6*
inverse degree	max	wvRN	67.7
likelihood ratio	cosine function	cdRN	**68.0**
beta distribution	sum of shared nodes	nlb	**68.5**
w=1	sum of shared nodes	nlb 100	68.5
adamic and adar	jaccard	wvRN	*70.3*
beta distribution	jaccard	cdRN	*71.4*
likelihood ratio	cosine function	wvRN	71.5
w=1	jaccard	nlb	*72.1*
likelihood ratio	cosine function	nlb	72.8
adamic and adar	jaccard	cdRN	**72.8**
adamic and adar	sum of shared nodes	nlb 100	72.9
w=1	cosine function	nlb 100	73.2
delta	sum of shared nodes	nlb	73.5
tanh	max	nlb	74.1
beta distribution	cosine function	nlb 100	74.2
tanh	max	cdRN	74.2
inverse degree	max	nlb	74.5
inverse degree	max	cdRN	74.7
delta	sum of shared nodes	cdRN	74.8
likelihood ratio	jaccard	cdRN	**74.8**
likelihood ratio	sum of shared nodes	wvRN	75.4
tanh	jaccard	nlb 100	75.8
adamic and adar	cosine function	nlb 100	76.3
inverse frequency	jaccard	nlb 100	76.4
beta distribution	jaccard	wvRN	76.7
inverse frequency	max	wvRN	76.9
inverse degree	jaccard	nlb 100	77.8

Continued on next page

Table 14 – *Continued from previous page*

Top node weight	Aggregation function	Classifier	Avg Ranking
adamic and adar	jaccard	nlb	77.8
likelihood ratio	sum of shared nodes	nlb	79.1
beta distribution	sum of shared nodes	nlb 100	79.9
inverse frequency	max	nlb	80.1
likelihood ratio	jaccard	nlb	*80.4*
inverse frequency	max	cdRN	80.5
likelihood ratio	sum of shared nodes	cdRN	80.7
delta	max	wvRN	80.8
beta distribution	max	cdRN	80.9
delta	cosine function	nlb 100	81.3
SVM			81.4
likelihood ratio	jaccard	wvRN	81.7
likelihood ratio	cosine function	nlb 100	84.9
w=1	jaccard	nlb 100	85.0
likelihood ratio	sum of shared nodes	nlb 100	85.6
tanh	max	nlb 100	86.3
inverse degree	max	nlb 100	86.5
delta	jaccard	wvRN	86.6
beta distribution	jaccard	nlb 100	87.9
delta	max	nlb	89.3
delta	sum of shared nodes	nlb 100	89.3
inverse frequency	max	nlb 100	89.5
delta	max	cdRN	90.3
delta	jaccard	cdRN	91.5
adamic and adar	jaccard	nlb 100	91.7
delta	jaccard	nlb	93.7
adamic and adar	max	nlb	96.0
adamic and adar	max	wvRN	96.7
likelihood ratio	jaccard	nlb 100	96.9
adamic and adar	max	cdRN	97.6
beta distribution	max	nlb 100	99.7
delta	max	nlb 100	101.0
likelihood ratio	max	wvRN	105.5
w=1	max	nlb	107.1
w=1	zero - one	nlb	107.4
tanh	zero - one	nlb	107.4
inverse degree	zero - one	nlb	107.4
inverse frequency	zero - one	nlb	107.4
adamic and adar	zero - one	nlb	107.4
delta	zero - one	nlb	107.4
adamic and adar	max	nlb 100	107.8
w=1	max	wvRN	108.6
w=1	zero - one	wvRN	108.9
tanh	zero - one	wvRN	108.9
inverse degree	zero - one	wvRN	108.9
inverse frequency	zero - one	wvRN	108.9
adamic and adar	zero - one	wvRN	108.9
delta	zero - one	wvRN	108.9
delta	jaccard	nlb 100	108.9
likelihood ratio	max	cdRN	110.8
beta distribution	zero - one	nlb	111.1
likelihood ratio	max	nlb	112.0
likelihood ratio	max	nlb 100	114.7
w=1	zero - one	cdRN	116.1
tanh	zero - one	cdRN	116.1

Continued on next page

Table 14 – *Continued from previous page*

Top node weight	Aggregation function	Classifier	Avg Ranking
inverse degree	zero - one	cdRN	116.1
inverse frequency	zero - one	cdRN	116.1
adamic and adar	zero - one	cdRN	116.1
likelihood ratio	zero - one	wvRN	116.2
w=1	max	cdRN	116.4
delta	zero - one	cdRN	116.4
beta distribution	zero - one	cdRN	117.6
likelihood ratio	zero - one	cdRN	118.1
beta distribution	zero - one	wvRN	118.5
likelihood ratio	zero - one	nlb	118.6
w=1	max	nlb 100	123.7
adamic and adar	zero - one	nlb 100	124.5
w=1	zero - one	nlb 100	124.9
tanh	zero - one	nlb 100	124.9
inverse degree	zero - one	nlb 100	124.9
inverse frequency	zero - one	nlb 100	124.9
delta	zero - one	nlb 100	124.9
likelihood ratio	zero - one	nlb 100	132.1
beta distribution	zero - one	nlb 100	132.5

Table 15: Average ranking for all the combinations of techniques. In case where a combination of methods is not able to run on a dataset, it gets no ranking for the specific dataset.

Top node weight	Aggregation function	Classifier	Avg Ranking
tanh	cosine function	wvRN	**39.6**
inverse degree	cosine function	wvRN	**40.1**
tanh	sum of shared nodes	wvRN	**41.1**
inverse degree	sum of shared nodes	wvRN	**41.9**
tanh	cosine function	cdRN	**41.9**
tanh	cosine function	nlb	**42.3**
inverse degree	cosine function	cdRN	**42.8**
inverse degree	cosine function	nlb	*42.9*
inverse frequency	cosine function	wvRN	*43.5*
inverse frequency	cosine function	cdRN	**43.8**
tanh	sum of shared nodes	nlb	**44.0**
inverse frequency	sum of shared nodes	wvRN	**44.0**
inverse frequency	cosine function	nlb	*44.9*
tanh	sum of shared nodes	cdRN	**45.4**
beta distribution	sum of shared nodes	wvRN	**46.5**
tanh	cosine function	nlb 100	*46.6*
inverse degree	sum of shared nodes	cdRN	**46.6**
inverse degree	sum of shared nodes	nlb	**46.7**
inverse degree	cosine function	nlb 100	46.8
tanh	sum of shared nodes	nlb 100	**47.2**
inverse degree	sum of shared nodes	nlb 100	*48.3*
inverse frequency	cosine function	nlb 100	48.7
inverse frequency	sum of shared nodes	nlb 100	*49.2*
inverse frequency	sum of shared nodes	nlb	*49.4*
tanh	jaccard	wvRN	51.1
inverse frequency	sum of shared nodes	cdRN	*52.0*
inverse frequency	jaccard	wvRN	*52.7*

Continued on next page

Table 15 – *Continued from previous page*

Top node weight	Aggregation function	Classifier	Avg Ranking
inverse frequency	jaccard	cdRN	**52.9**
w=1	sum of shared nodes	wvRN	*53.2*
inverse degree	jaccard	wvRN	53.6
w=1	cosine function	cdRN	*54.2*
beta distribution	cosine function	wvRN	**54.6**
tanh	jaccard	cdRN	*54.8*
w=1	cosine function	wvRN	55.0
beta distribution	cosine function	nlb	**55.2**
beta distribution	cosine function	cdRN	*55.7*
adamic and adar	sum of shared nodes	wvRN	*56.3*
inverse degree	jaccard	cdRN	*56.5*
w=1	cosine function	nlb	58.3
adamic and adar	cosine function	wvRN	58.4
w=1	sum of shared nodes	nlb	58.6
adamic and adar	cosine function	cdRN	*59.3*
w=1	sum of shared nodes	nlb 100	59.6
beta distribution	sum of shared nodes	cdRN	*59.9*
beta distribution	jaccard	nlb	60.1
beta distribution	max	wvRN	*60.3*
w=1	jaccard	wvRN	60.6
tanh	jaccard	nlb	61.0
adamic and adar	cosine function	nlb	61.2
inverse frequency	jaccard	nlb	61.3
beta distribution	max	nlb	*61.4*
adamic and adar	sum of shared nodes	nlb	61.7
delta	cosine function	wvRN	62.0
w=1	sum of shared nodes	cdRN	62.3
w=1	jaccard	cdRN	**62.3**
beta distribution	cosine function	nlb 100	62.6
delta	sum of shared nodes	wvRN	63.5
inverse degree	jaccard	nlb	63.8
delta	cosine function	cdRN	64.4
w=1	cosine function	nlb 100	64.6
adamic and adar	sum of shared nodes	nlb 100	64.6
beta distribution	sum of shared nodes	nlb	*64.8*
adamic and adar	sum of shared nodes	cdRN	65.1
delta	cosine function	nlb	65.5
tanh	max	wvRN	66.2
inverse degree	max	wvRN	66.7
adamic and adar	jaccard	wvRN	66.8
likelihood ratio	cosine function	cdRN	67.0
tanh	jaccard	nlb 100	67.6
beta distribution	jaccard	cdRN	68.0
adamic and adar	cosine function	nlb 100	68.2
inverse frequency	jaccard	nlb 100	68.3
w=1	jaccard	nlb	68.7
adamic and adar	jaccard	cdRN	*69.5*
beta distribution	sum of shared nodes	nlb 100	69.7
inverse degree	jaccard	nlb 100	69.9
likelihood ratio	cosine function	wvRN	70.6
likelihood ratio	jaccard	cdRN	*71.7*
likelihood ratio	cosine function	nlb	72.0
delta	sum of shared nodes	nlb	72.8
tanh	max	nlb	73.4
tanh	max	cdRN	73.5

Continued on next page

Table 15 – *Continued from previous page*

Top node weight	Aggregation function	Classifier	Avg Ranking
beta distribution	jaccard	wvRN	73.7
inverse degree	max	nlb	73.9
inverse degree	max	cdRN	74.1
delta	sum of shared nodes	cdRN	74.1
delta	cosine function	nlb 100	74.2
adamic and adar	jaccard	nlb	74.9
likelihood ratio	sum of shared nodes	wvRN	75.4
inverse frequency	max	wvRN	76.4
likelihood ratio	jaccard	nlb	77.8
beta distribution	max	cdRN	78.3
likelihood ratio	cosine function	nlb 100	78.4
w=1	jaccard	nlb 100	78.6
likelihood ratio	sum of shared nodes	nlb	78.7
likelihood ratio	jaccard	wvRN	79.3
SVM			79.3
likelihood ratio	sum of shared nodes	nlb 100	79.5
beta distribution	jaccard	nlb 100	79.5
inverse frequency	max	nlb	79.8
tanh	max	nlb 100	80.0
inverse frequency	max	cdRN	80.2
inverse degree	max	nlb 100	80.3
likelihood ratio	sum of shared nodes	cdRN	80.4
delta	max	wvRN	80.5
delta	sum of shared nodes	nlb 100	83.7
inverse frequency	max	nlb 100	83.9
delta	jaccard	wvRN	84.6
adamic and adar	jaccard	nlb 100	85.3
delta	max	nlb	89.4
delta	jaccard	cdRN	90.0
delta	max	cdRN	90.5
likelihood ratio	jaccard	nlb 100	90.6
delta	jaccard	nlb	92.4
beta distribution	max	nlb 100	94.1
adamic and adar	max	nlb	96.5
adamic and adar	max	wvRN	97.2
delta	max	nlb 100	97.4
adamic and adar	max	cdRN	98.3
adamic and adar	max	nlb 100	105.5
delta	jaccard	nlb 100	106.1
likelihood ratio	max	wvRN	106.5
w=1	max	nlb	108.2
w=1	zero - one	nlb	108.6
tanh	zero - one	nlb	108.6
inverse degree	zero - one	nlb	108.6
inverse frequency	zero - one	nlb	108.6
adamic and adar	zero - one	nlb	108.6
delta	zero - one	nlb	108.6
w=1	max	wvRN	109.8
w=1	zero - one	wvRN	110.2
tanh	zero - one	wvRN	110.2
inverse degree	zero - one	wvRN	110.2
inverse frequency	zero - one	wvRN	110.2
adamic and adar	zero - one	wvRN	110.2
delta	zero - one	wvRN	110.2
beta distribution	zero - one	nlb	111.4

Continued on next page

Table 15 – *Continued from previous page*

Top node weight	Aggregation function	Classifier	Avg Ranking
likelihood ratio	max	cdRN	112.1
likelihood ratio	max	nlb	113.4
likelihood ratio	max	nlb 100	113.7
w=1	zero - one	cdRN	117.7
tanh	zero - one	cdRN	117.7
inverse degree	zero - one	cdRN	117.7
inverse frequency	zero - one	cdRN	117.7
adamic and adar	zero - one	cdRN	117.7
likelihood ratio	zero - one	wvRN	117.8
w=1	max	cdRN	118.0
delta	zero - one	cdRN	118.1
beta distribution	zero - one	cdRN	118.5
beta distribution	zero - one	wvRN	119.5
likelihood ratio	zero - one	cdRN	119.9
likelihood ratio	zero - one	nlb	120.3
w=1	max	nlb 100	124.3
adamic and adar	zero - one	nlb 100	125.3
w=1	zero - one	nlb 100	125.7
tanh	zero - one	nlb 100	125.7
inverse degree	zero - one	nlb 100	125.7
inverse frequency	zero - one	nlb 100	125.7
delta	zero - one	nlb 100	125.7
likelihood ratio	zero - one	nlb 100	134.3
beta distribution	zero - one	nlb 100	134.5

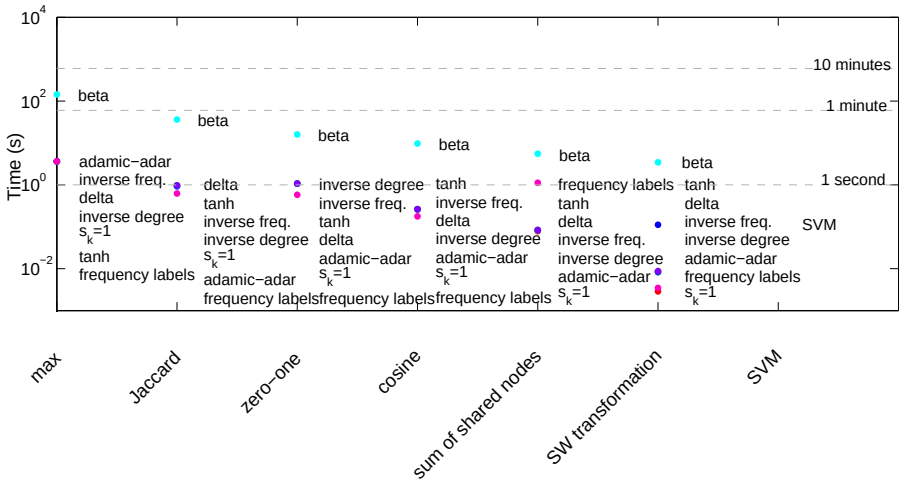


Fig. 14: Aggregated run-time results for each of the top node and aggregation functions with wvRN (including the SW-transformation). Since most of the top node functions (except for the beta) have similar durations, the markers on the plots are very close to each other (and given in descending order).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

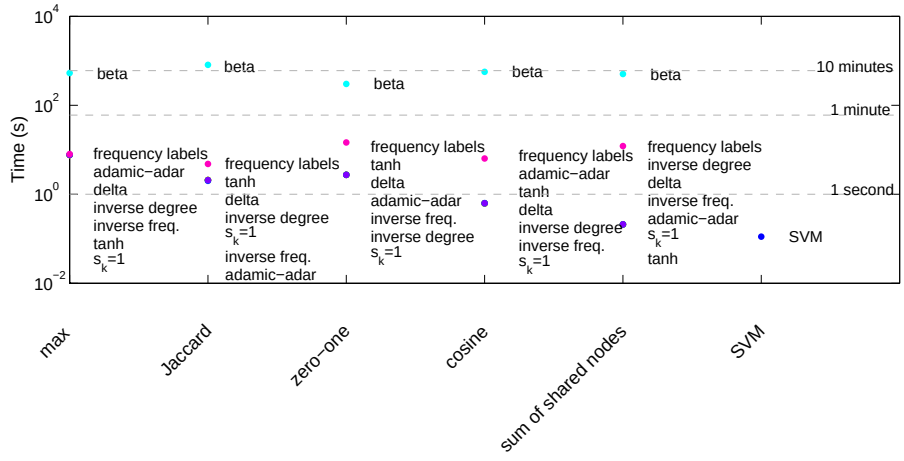


Fig. 15: Aggregated run-time results for each of the top node and aggregation functions with the nLB classifier.

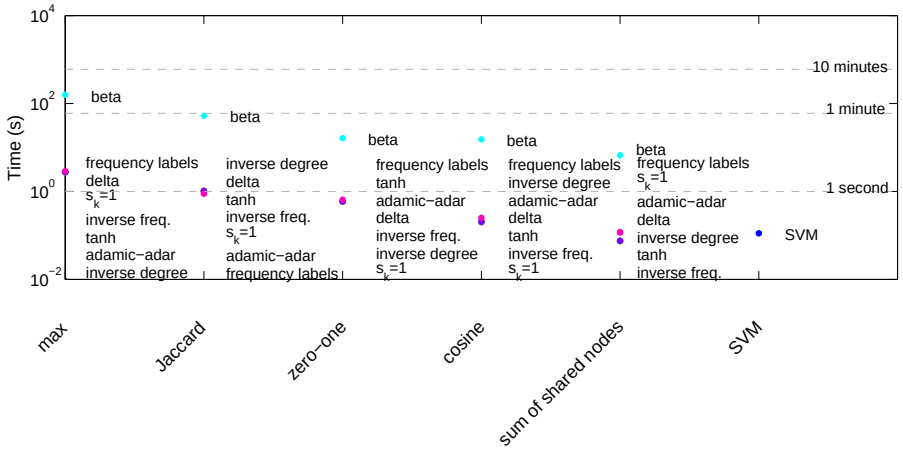


Fig. 16: Aggregated run-time results for each of the top node and aggregation functions with the nLB100 classifier (nLB with 100 training instances).

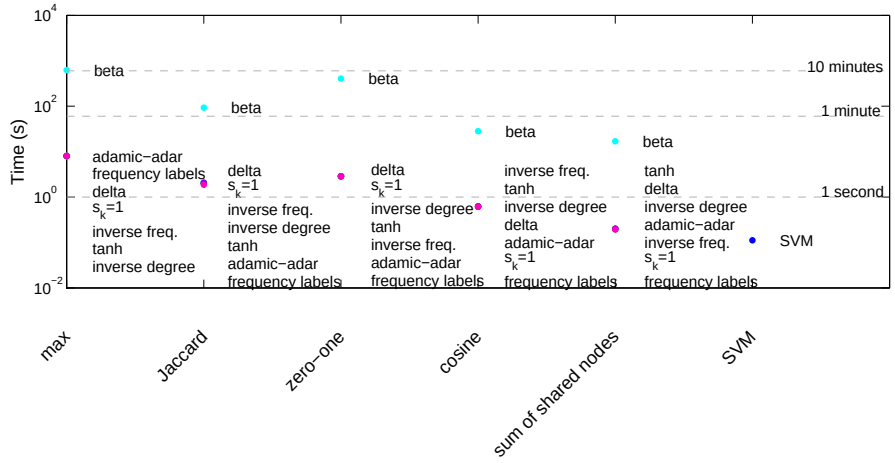


Fig. 17: Aggregated run-time results for each of the top node and aggregation functions with the cdRN classifier.

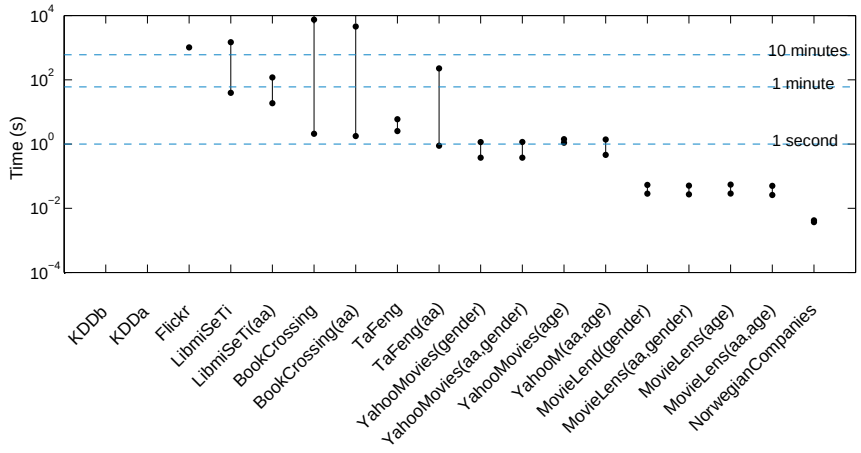


Fig. 18: Time improvement of nLB with sampling over 100 instances as compared to no sampling for different datasets. The top of each bar represents the time needed for the nLB classifier and the bottom of each bar the time required to train the nLB with 100 instances for the specific dataset.

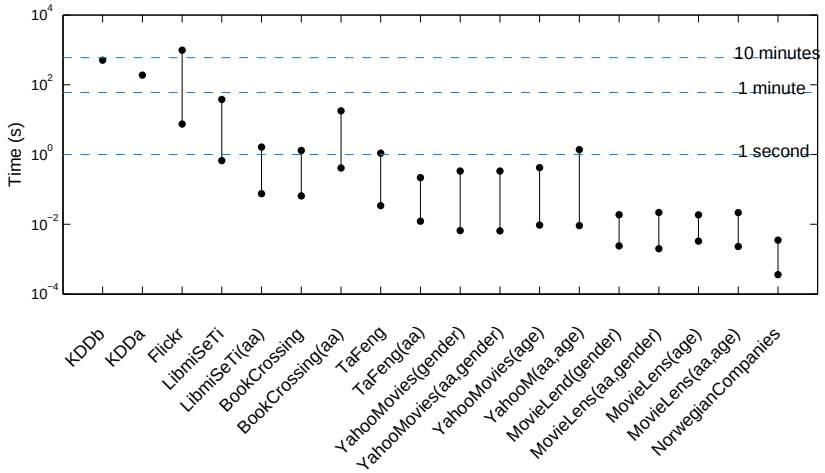


Fig. 19: Time improvement of the SW-transformation over wvRN and sum of shared nodes for different datasets. The top of each bar represents the time needed for the wvRN classifier and the bottom of each bar the time required for SW-transformation for the specific dataset.

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
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	0.6264
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
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	0.7244

Table 16: Best combinations of methods per dataset.

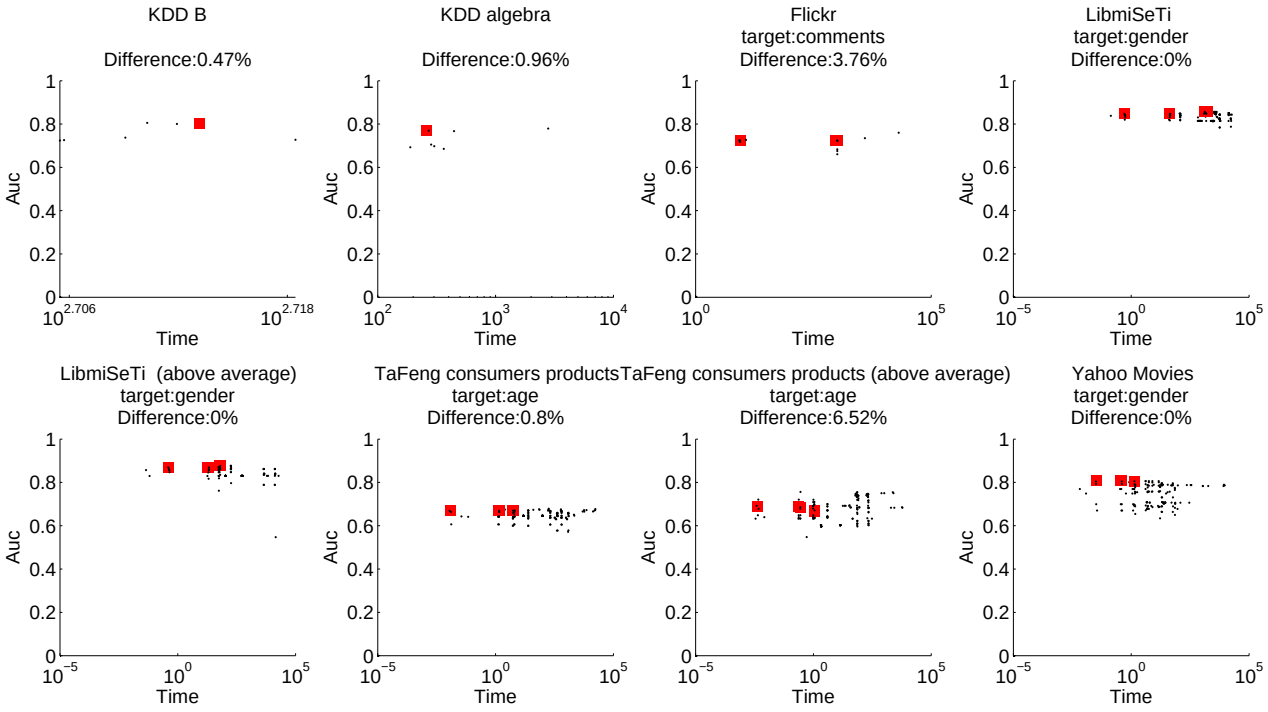


Fig. 20: Ranking of all combinations of methods, with the proposed combinations highlighted in red.

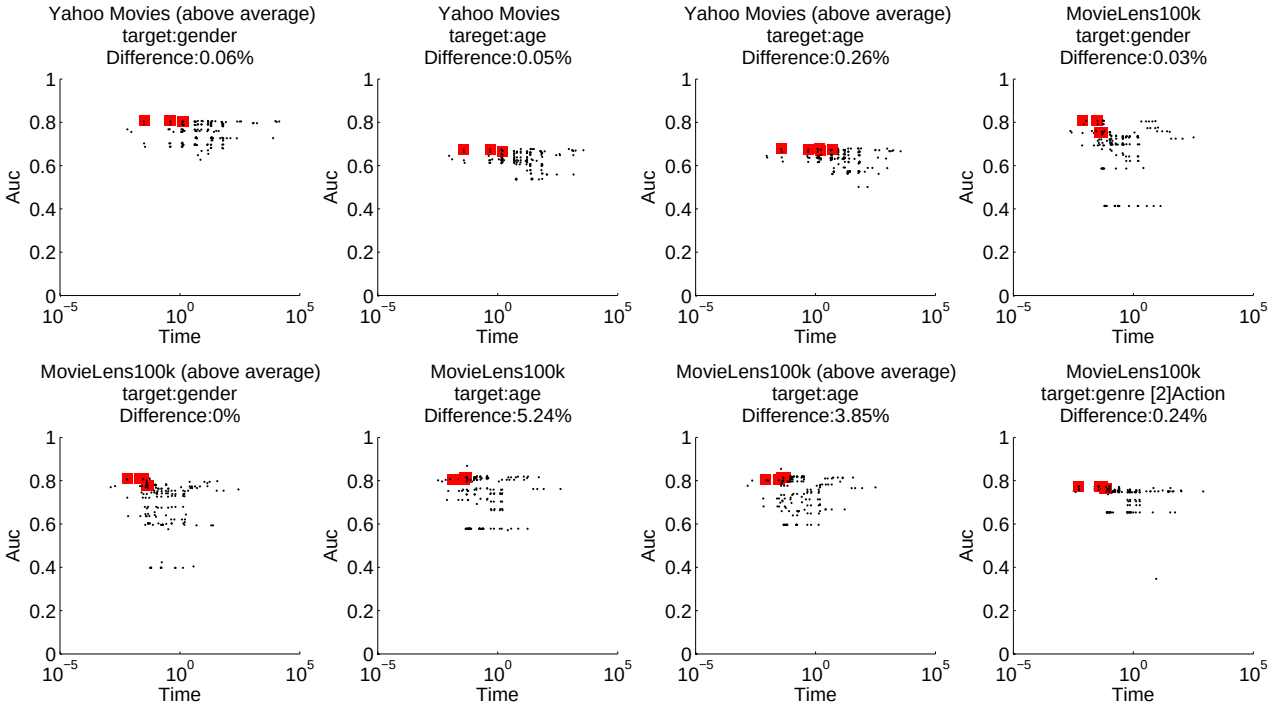


Fig. 21: Ranking of all combinations of methods, with the proposed combinations highlighted in red.

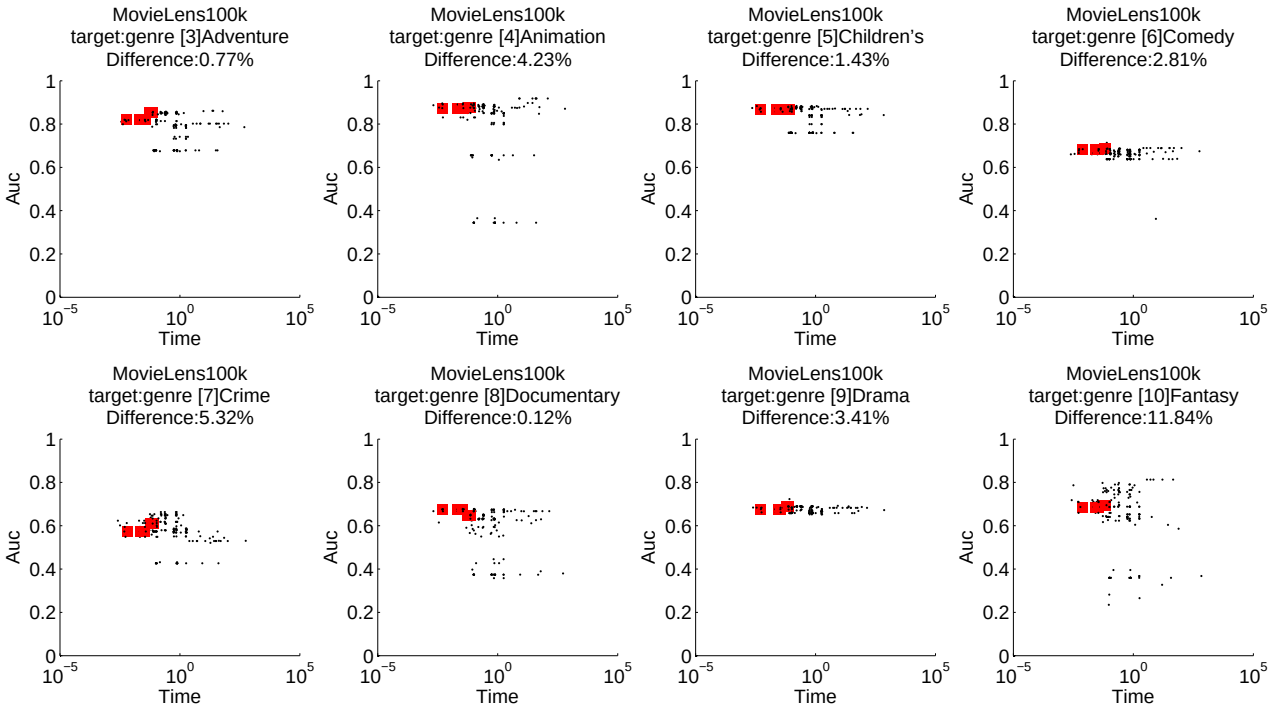


Fig. 22: Ranking of all combinations of methods, with the proposed combinations highlighted in red.

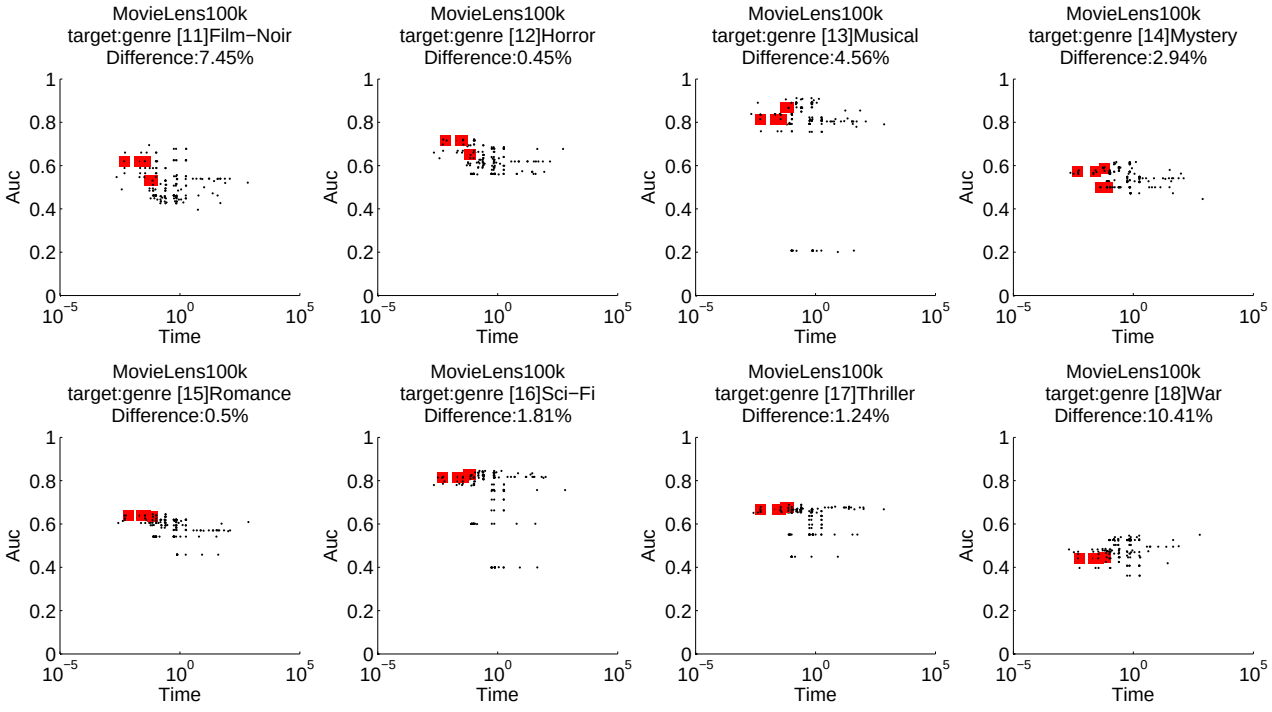


Fig. 23: Ranking of all combinations of methods, with the proposed combinations highlighted in red.

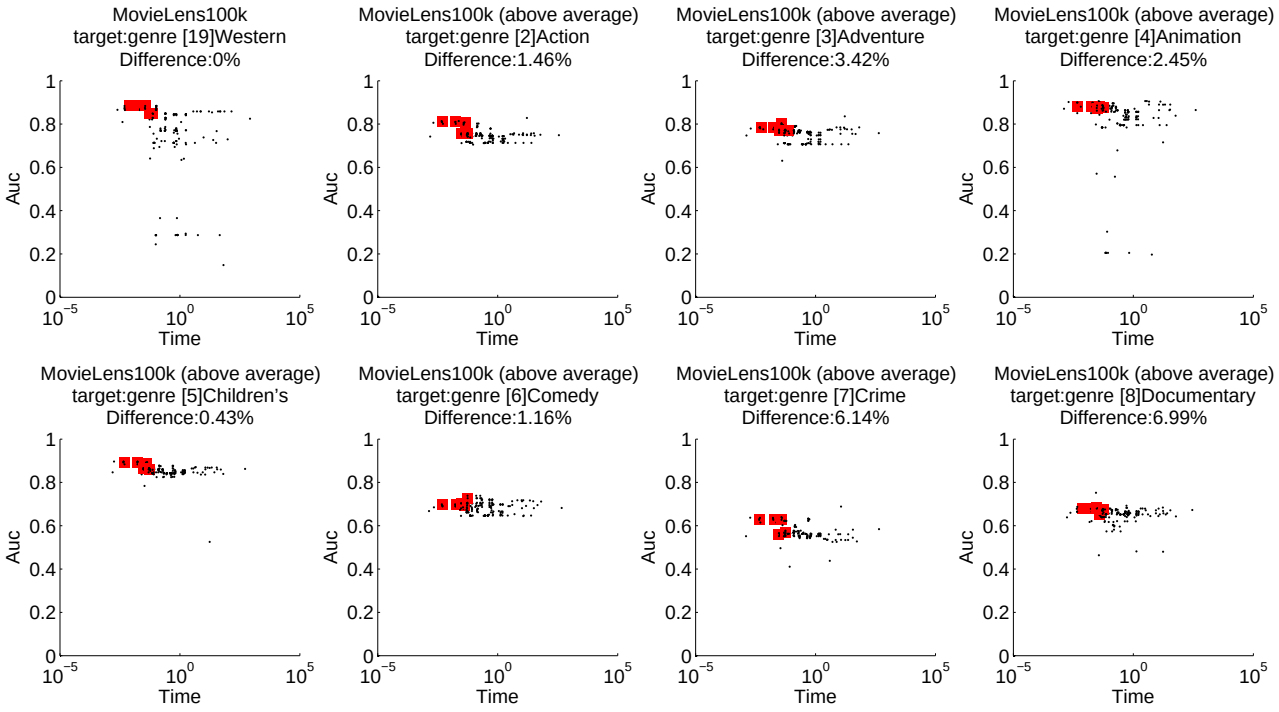


Fig. 24: Ranking of all combinations of methods, with the proposed combinations highlighted in red.

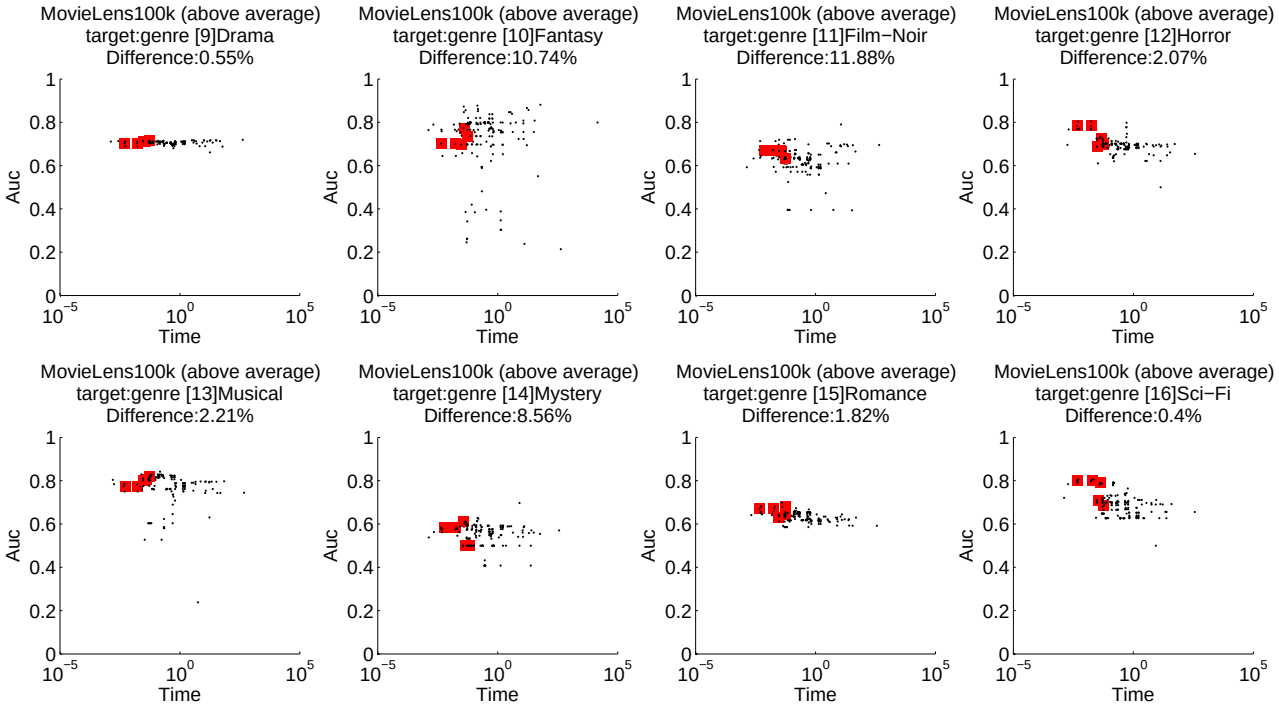


Fig. 25: Ranking of all combinations of methods, with the proposed combinations highlighted in red.

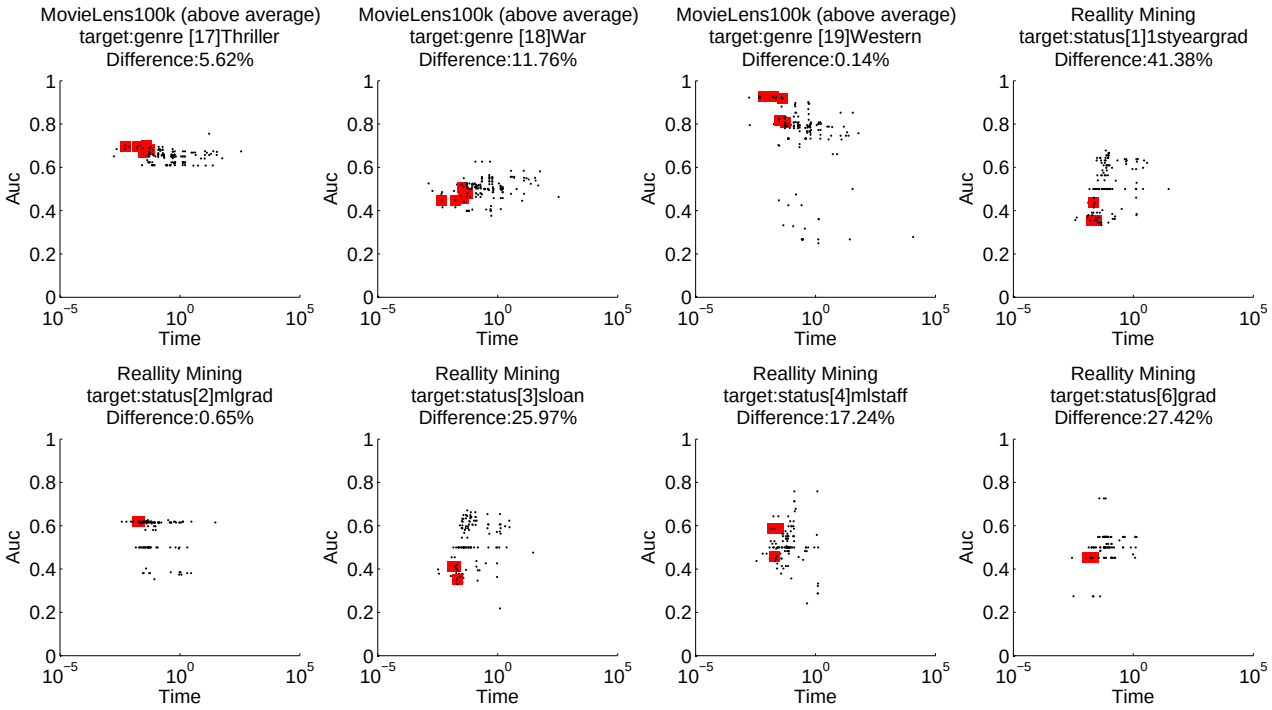


Fig. 26: Ranking of all combinations of methods, with the proposed combinations highlighted in red.

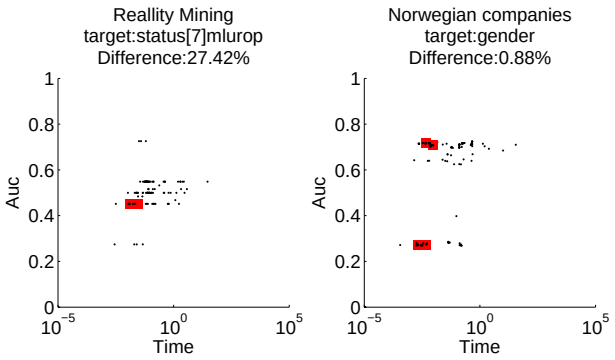


Fig. 27: Ranking of all combinations of methods, with the proposed combinations highlighted in red.