A Hybrid Approach of Classifier and Clustering for Solving the Missing Node Problem

Sigal Sina
Bar-Ilan University, Israel
sinasi@macs.biu.ac.il

Avi Rosenfeld
Jerusalem College of Technology, Israel
rosenfa@jct.ac.il

Sarit Kraus
Bar-Ilan University, Israel
sarit@cs.biu.ac.il

Navot Akiva
Bar-Ilan University, Israel
navot.akiva@gmail.com

Abstract

An important area of social network research is identifying missing information which is not explicitly represented in the network or is not visible to all. In this paper, we propose a novel **H**ybrid **A**pproach of **C**lassifier and **C**lustering, which we refer to as **HACC**, to solve the *missing node identification* problem in social networks. HACC utilizes a classifier as a preprocessing step in order to integrate all known information into one similarity measure and then uses a clustering algorithm to identify missing nodes. Specifically, we used the information on the network structure, attributes about known users (nodes) and pictorial information to evaluate HACC and found that it performs significantly better than other missing node algorithms. We also argue that HACC is a general approach and domain independent and can be easily applied to other domains. We support this claim by evaluating HACC on a second *authorship identification* domain as well.

Introduction

Data clustering has been used for more than 50 years in order to find a natural grouping of a set of patterns, points, or objects based on similarity or dissimilarity measures (Jain 2010). The main challenge is to find the measures that can improve clustering accuracy. To achieve this goal, in this paper we propose **HACC**, a novel **H**ybrid **A**pproach of **C**lassifier and **C**lustering, for solving the *missing node identification* problem in social networks. Additionally, we argue that our approach is general and domain independent and can be easily applied to other domains. The novelty of HACC is its use of a small collection of labeled objects to tune and learn a classifier as a preprocessing step in order to integrate all known information into one similarity measure. The classifier learning is done only once and in advance, using labeled objects of different clusters than the clusters of the unlabeled data. Later, HACC can use this classifier to facilitate and improve the clustering result of different, large collections of unlabeled objects. To the best of our knowledge, HACC is the first algorithm to combine the use of supervised classifiers as a preprocessing step before activating the unsupervised clustering algorithm of objects of different clusters than the unlabeled ones.

Machine learning methods including clustering and classification are well-established and have been used in a variety of fields (Bishop 2006; Jain 2010). Several researchers have also suggested combining several methods of classification in order to yield better results (Kuncheva 2004). Examples of hybrid methods containing both clustering and classification classification were presented by Nguyen and Armitage, (Nguyen and Armitage 2008) and Shrot et al. (Shrot et al. 2014). However, these works used the clustering algorithm as a first step of unsupervised classification before a second, refinement step of supervised classification. Other work (Basu, Bilenko, and Mooney 2004) suggested a semi-supervised clustering framework, where the performance of unsupervised clustering algorithms can be improved with limited amounts of supervision in the form of labels on the data or constraints. They proposed a principled probabilistic framework based on Hidden Markov Random Fields (HMRFs) for semi-supervised clustering that combined the existing constraint-based and distance-based approaches in a unified model. However, their model assumed that part of objects needs to be clustered is labeled, while in HACC the classifier is learned in advance. Another difference is that the HACC classifier is based on a different set of labeled objects and clusters than the objects currently being processed.

To better understand HACC's novelty, consider the authorship identification domain. Existing works on authorship identification are unsupervised or supervised, where the author's model is learned based on the same author's previous publications (Stamatatos 2009). The generalization of the learned model is for new publications of the same authors, as the tagging of both sets is the same. In contrast, HACC learns models from one set of authors and generalizes its model to identify another set of authors. Here, the generalization is from one set of authors and their publications to identify a new set of authors and their new set of publications. As such, the tagging sets of the learned data and the testing data are different – not only the data points. The advantage of HACC's type of learning is that it facilitates a hybrid model that creates a classifier which is not used for categorization, but instead as input for a second learning model in HACC.

The main contribution of HACC is the way it utilizes a pre-learned classifier as a preprocessing step. HACC creates one combined *affinity* measure between each object pair, which is the classifier's *confidence* score, that measures the probability that two objects needing to be clustered will be in the same cluster. We first demonstrate how HACC can be implemented in order to solve the *missing node identification* problem (Eyal, Rosenfeld, and Kraus 2011), which locates and identifies missing nodes within the network. For this domain, we conducted a thorough study which includes an intensive evaluation and analysis of 5 different types of classifiers and 12 different sets of domain features. We show that HACC performs significantly better than other missing node algorithms. To support the claim that HACC is domain independent and can be easily applied to other web domains, we also implemented HACC for an *authorship identification* task and again found that HACC performs significantly better than the baseline algorithm (Akiva and Koppel 2013).

HACC and Problem Definition

HACC proposes combining supervised classifiers as a preprocessing step before activating a clustering algorithm in order to effectively cluster a group of unlabeled objects. The goal of the clustering task is to discover the natural grouping of these unlabeled objects.

Problem Definition Suppose we have a set of data objects, D, which has an inherent clustering scheme and should be clustered into x clusters such that $C(D)$ represents all x clusters. For example, the data's clusters may represent nodes originating from a person within a social network or clusters of documents that an author wrote. D is divided into a set, O, of unlabeled data and a small additional collection of labeled objects $\hat{O}$ from D which is already labeled with its representative clusters. However, $\hat{O} << O$ and no overlap exists between the two sets (i.e. $\hat{O} \cap O = \emptyset$). Additionally, assuming $|C(O)|=K$ and $|C(\hat{O})|=\hat{K}$, then $x = K+\hat{K}$, meaning that the clusters from $\hat{O}$ do not appear within O such that $C(\hat{O}) \cap C(O)=\emptyset$. Our goal is to utilize $\hat{O}$ to help create the best possible division of O to K clusters.

The HACC Algorithm HACC is a two-staged algorithm, where its novelty is how it uses $\hat{O}$ to predict object pairs' *confidence* score in order to improve the clustering performance. The HACC algorithm is defined as follows:

1. Builds a classifier (Offline):
   (i) Inputs vectors of similarity features for all object pairs in $\hat{O}$ together with a label whether or not they belong to the same cluster or not;
   (ii) Returns a probabilistic *confidence* score whether they are in the same cluster.
2. Clusters objects in O (Online):
   (i) Calculates the similarity feature vectors, this time from all pairs of data objects in O, and inputs them into the classifier created at stage one to obtain the *confidence* score for each object pair;
   (ii) Uses the *confidence* scores as a combined *affinity* measure to create an *affinity* matrix for all pairs in O;
   (iii) Runs a clustering algorithm, such as K-means, that uses the *affinity* matrix as its input.

Note that the feature vectors of the classifier are similarity measures and thus are associated with each pair of objects and not with a single object.

The Evaluation Measure We used *purity* as our primary evaluation measure in order to gauge the effectiveness of clustering algorithms as it is an accepted measure of checking the quality and accuracy of clustering algorithms (Strehl and Ghosh 2003). The purity measure is calculated in the following two steps: Step one – each cluster is classified according to the true classification of the majority of samples in that cluster. Here, we classified each cluster according to the most frequent grouping label of the objects in that cluster; Step two – the number of correctly classified samples in all clusters are counted and divided by the number of samples. In our case, the number of samples that are classified is $|O|$. Formally, in our problem setting, where c_k is defined as the set of objects which were assigned to cluster k, and the objective truth for a sample $o \in O$ is denoted $t(o)$, purity is defined as: $purity(C) = \frac{1}{|O|} \sum_k max_{c \in C(O)} |c_k \cap \{o \in O \mid t(o) = c\}|$.

To demonstrate how HACC can be applied to two web domains, we now present a thorough study of the *missing node identification* problem as well as a brief study of the *authorship identification* problem.

The Missing Node Identification Problem

We begin with a background about the *missing node identification* problem. We then formally define the problem which we address, describe how HACC is implemented in this domain, and our experimental setup.

Overview and Background We focus on a specific variation of the missing nodes problem where the missing nodes requiring identification are "friends" of known nodes, as previously done with the MISC and SAMI based algorithms (Eyal, Rosenfeld, and Kraus 2011; Sina, Rosenfeld, and Kraus 2013). An unidentified friend is associated with a "placeholder" node to indicate the existence of the missing friend. Thus, a given missing node may be associated with several "placeholder" nodes, one for each friend of the missing node. Following this approach, the missing node challenge is to try to determine which of the "placeholder" nodes are associated with the same unidentified friend and thus to find the correct clustering of the "placeholder" nodes. To better understand this problem, consider the following example: A hypothetical company, Social News Inc., is running an online news service within LinkedIn. Many LinkedIn members are subscribers of this company's services, yet it would like to expand its customer base. Social News maintains a network of users, which is a subset of the group of LinkedIn users, and the links between these users. The users of LinkedIn who are not members of the service are not visible to their system. Social News Inc. would like to discover the LinkedIn nodes to lure them into joining their service. The company is thus faced with the missing nodes identification problem. By solving this problem, Social News Inc. could improve its advertising techniques and target users who have not yet subscribed to their service.

The *missing node identification* problem has recently generated research interest. Previously, Eyal et al. (Eyal, Rosenfeld, and Kraus 2011; Eyal et al. 2014) presented the MISC algorithm in order to solve this problem. MISC com-

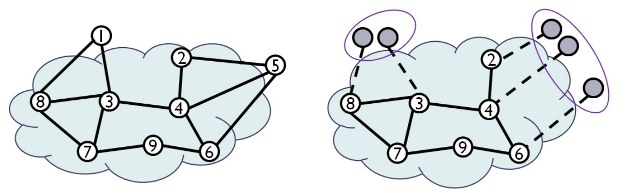


Figure 1: A full network, the known network and the visible network obtained (right) by adding the placeholders for the missing nodes 1 and 5, which need to be united into clusters.

bines spectral clustering algorithms (Ng, Jordan, and Weiss 2001) and metrics built for the missing link problem (Liben-Nowell and Kleinberg 2007). The main idea behind their work was to embed a set of data points, which should be clustered, in a graph structure representing the *affinity* between each pair of points based on the structure of the network. Kim and Leskovec (Kim and Leskovec 2011) tackled the *network completion* problem, which is a similar problem that deals with situations where only a part of the network is observed and the unobserved part must be inferred. They proposed the `KronEM` algorithm, which uses an Expectation Maximization approach and where the observed portion of the network is used to fit a Kronecker graph model of the full network structure. Sina et al. (Sina, Rosenfeld, and Kraus 2013) extended the *missing node identification* problem by considering how to incorporate information about the known nodes in the network in order to improve performance. They presented the `SAMI-A` and `SAMI-N` algorithms, which used different ways to integrate the additional attributes' information about the known nodes with the network structure's *affinity* measure. They showed that these algorithms, which leverage the additional information, achieve predictions with significantly better quality than both the `KronEM` and `MISC` algorithms, which only used information about the network's structure. However, none of these previous works considered how to generally incorporate information, including pictorial information which has become ubiquitous in many social networks, as we do in this work. The challenge in this type of data is that it requires a model that can handle missing features as many of the nodes have no pictures.

The idea of using information from specific nodes was previously considered in a variety of different problems. Several previous works (Gong et al. 2011; Yin et al. 2010) proposed a model to jointly infer missing links and missing node attributes by representing the social network as an augmented graph where the nodes' attributes are represented as special nodes in the network. Other approaches studied different ways of leveraging information about known nodes within the network in order to better solve the missing link or missing attribute problems. For example, Freno et al. (Freno, Garriga, and Keller 2011) proposed a supervised learning method which uses both the graph structure and node attributes to recommend missing links. Pictures in social networks are also considered in a variety of different problems. These problems include cascade prediction in the network (Cheng et al. 2014), placing images on the world map (Hauff and Houben 2012) and learning influence in the social network (Goyal, Bonchi, and Lakshmanan 2010). However, these works use the pictures' descriptions, tags and some properties, such as color histograms, but do not consider the relative similarity of pictures as in the PIC measure we propose.

Problem Definition We assume that there is a social network represented as an undirected graph G=(V, E), in which n=$|V|$ and e=$\langle v, u\rangle$$\in$$E$ represents an interaction between $v$$\in$$V$ and $u$$\in$$V$. In addition to the network structure, each node $v_i$$\in$$V$ is associated with an attribute vector $\vec{AV_i}$ of length l and a set of pictures IM_i of size l_2. We assume that each attribute in the attribute vectors is a binary value, i.e. each node has or does not have the attribute. Formally, we define a binary attribute matrix A of size nxl where $A_{i,j}$ indicates whether or not a node $v_i$$\in$$V$ has an attribute j. Some of the nodes in the network are missing and are not known to the system. We denote the set of missing nodes $V_m \subset V$, and assume that the number of missing nodes is given as K=$|V_m|$.

We denote the rest of the nodes as known, i.e., V_k=$V \setminus V_m$, and the set of known edges is $E_k = \{\langle v, u\rangle \mid v, u \in V_k \wedge \langle v, u\rangle \in E\}$. In order to identify the missing nodes, we focus on the visible part of the network that is available G_a=(V_a, E_a). In this network, each of the missing nodes is replaced by a set of placeholders. Formally, we define a set V_p for placeholders and a set E_p for the associated edges. For each missing node $v$$\in$$V_m$ and for each edge $\langle v, u\rangle$$\in$$E$, $u$$\in$$V_k$, a placeholder is created. That is, for each original edge $\langle v, u\rangle$ we assume that there is a placeholder v' for v in V_p and the placeholder is connected to the node u with a new edge $\langle v', u\rangle$, which belongs to E_p. When these components are considered together, V_a=$V_k$$\cup$$V_p$ and E_a=$E_k$$\cup$$E_p$. As for a given missing node v, there may be many placeholders in V_p. The missing node challenge is to try to determine which of the placeholders should be clustered together and associated with the original v, thus allowing us to reconstruct the original social network G.

To better understand this formalization, consider the following example: Assume we have a gamers' network where users may choose to register or remain anonymous before playing. An edge between two users indicates that these two users played a game together. We might have additional information regarding the registered users, such as pictures, origin, group membership and playing time. We consider the registered users to be the known nodes, while the anonymous users are the placeholders. We would like to identify which of the anonymous users are actually the same person.

Domain Features The success of a clustering algorithm depends on its *affinity* matrix - a calculated measure between each pair of nodes in the network, which is used to determine which of the placeholders are associated with the same source node. In `HACC`, we input vectors of similarity features for all node pairs to create the classifier model. This model returns as output a *confidence* score which is entered into a matrix and then used for the clustering algorithm. We considered as input a variety of known similarity measures as fields within the vectors. These include features based on network structure that were previously studied (Eyal, Rosenfeld, and Kraus 2011). Specifically, we used the two structure similarity measures that previously yielded the best results: the Relative Common Neighbors (RCN) measure (Liben-Nowell and Kleinberg 2007) and the Adamic/Adar (AA) measure (Adamic and Adar 2003), and a third structure similarity measure which is the combination of the two measures (RCN+AA), formally defined as $(RCN{+}AA)_{ij}{=}\frac{1}{2}(RCN_{ij}{+}AA_{ij})$. Additionally, we used two other similarity measures which are not based on the general structure of the network, but similarities between the attributes of a specific pair of nodes, v_i and v_j. The first measure is based on a previously developed measure of the common attributes between nodes (ATT) (Sina, Rosenfeld, and Kraus 2013) and a second measure based on picture similarity (PIC) is new to this work. We defined the picture similarity, PIC, as: $PIC_{ij}{=}MatchConf(i,j)$, where $MatchConf(i,j)$ represents the highest matching confidence between the pictures associated with nodes i and j, and zero if one of the nodes does not have any pictures. To calculate the matching confidence, we used a free web service for face detection and recognition http://betafaceapi.com/.

The MI-HACC Algorithm

Algorithm 1 (MI-Alg) presents the pseudocode for solving the *missing node identification* problem. The strength of the HACC approach is the novel way the *affinity* matrix is calculated in line 1. Our proposed algorithm, which we refer to as **MIC** (**M**issing node **I**dentification using a **C**lassifier) uses the MI-Alg algorithm as the basis of the implementation of the HACC approach. We implemented several variations of the MIC algorithm for different types of classifiers and different subsets of node features as input (defined above). First, and only once offline, we calculated the feature vectors which are composed of the similarity measures of structure, attributes and pictures for the *training* dataset, and used them to learn and store 5 types of classifiers: decision tree (tree), neural network (nnet), k-nearest-neighbors (kNN), AdaBoost and Bayes. During runtime for each one of the learned classifiers, we calculated the selected similarity measures for the *test* dataset, used it as the classifier's input and set the pairwise *affinity* according to the classifier *confidence* score for each pair of nodes. We created 4 data type variations for each one of the classifiers which contained a different subset of the similarity measures: the first was based on structure only (S), the second on structure and attributes (SA), the third on the structure and pictures (SP), and last on the structure, pictures and attributes (SAP). For example, we define the matrix of affinity measures for the decision tree classifier with the SAP variation $MC^{Tree-SAP-RCN}$ using the RCN measure as: $MC^{Tree-SAP-RCN}_{ij} = predict(tree, RCN_{ij}, PIC_{ij}, ATT_{ij})$. Overall there are 60 instances of the MIC algorithm: 5 classifier types (tree, nnet, kNN, AdaBoost and Bayes), 4 data types (S,SA,SP and SAP) and 3 structure similarity types (RCN,AA and RCN+AA).

Note that although MI-Alg is a generalization of the MISC algorithm (Eyal, Rosenfeld, and Kraus 2011), it also introduces a major change in addition to HACC's novel calculation of the *affinity* matrix. In contrast to MISC, MI-Alg no longer processes the affinity measures of missing nodes through spectral clustering but instead exclusively uses the K-means clustering algorithm on the placeholders' affinity matrix. We chose to implement this change for two reasons: First, we wanted to improve the performance of the original MISC algorithm in large-scale networks. MISC applies spectral clustering to the affinity matrix of the entire network (of both known nodes and placeholders), while MI-Alg applies K-means only on the matrix of a small number of placeholder nodes; Second, we had to overcome a memory consumption challenge caused by the ATT measure calculation, which only yielded a partial solution in previous work (Sina, Rosenfeld, and Kraus 2013). Our preliminary tests show that this change improved the runtime and memory consumption without performance degradation.

Algorithm 1 MI-Alg (**M**issing node **I**dentification)

```
Input: $G_k{=}\langle V_k, E_k\rangle$ – the known part of the network
$G_a{=}\langle V_a, E_a\rangle$ – the available visible part of the network
$K$ – the number of missing nodes
$V_p$ – the placeholder nodes
$\alpha : G(V,E) \rightarrow \mathbb{R}^{|V_p|\times|V_p|}$ – a procedure for calculating the affinity matrix of placeholder nodes in a graph
Output: $C \in \mathbb{N}^{|V_a \setminus V_k|}$ - a vector indicating the cluster index of each placeholder node,
$\hat{G}{=}\left(\hat{V}, \hat{E}\right)$ – prediction of the full network graph

1: $\mathbf{A} \leftarrow \alpha(G_a)$ – calculate the affinity matrix of the placeholder nodes in the graph
2: C $\leftarrow$ k_means($\mathbf{A}$,N) – cluster the rows which match the placeholder nodes to K clusters
3: $\hat{V} \leftarrow V_k$, $\hat{E} \leftarrow E_k$ – initialize the output graph to contain the known network
4: For each cluster $c \in C$ create a new node $v_c \in \hat{V}$
5: For each placeholder $v$ in cluster $c$ and edge $(u,v) \in E_a$, create an edge $(u,v_c) \in \hat{E}$
6: Return $C$, $\hat{G}{=}\left(\hat{V}, \hat{E}\right)$
```

The Dataset and The Experimental Setup

We used a previously developed social network dataset, Steam (Becker et al. 2012) (http://steamcommunity.com), to empirically evaluate our work. The Steam community network is a large social network of players on the Steam gaming platform. The data we applied is from 2011 and contains 9 million nodes ("anonymous" users) and 82 million friendship edges. Each user had the following data: country (the origin of the user; 50% voluntarily provided their country), member since (the date when the user opened his Steam account), a list of game playing times and a list of group memberships. We chose three groups of attributes: country, playing time and player group association. These three groups formed a total of 60 attributes – one for the country, 20 attributes of different game playing times and 39 different official associations. We first crawled the social network and only selected nodes that had at least 2 games or groups. This crawling reduced the dataset size to 1.3 million nodes (users). We then extracted different network samples, using a Forest Fire (FF) walk (Leskovec and Faloutsos 2006). We crawled a 16K network from this reduced dataset, marked it as the *training* dataset and removed the nodes from the dataset. We then re-sampled the 16K node *training* dataset to extract several 2K *training* networks, which we used as $\hat{O}$ for learning the MIC classifiers and the parameters of the comparison algorithm MIWS (described below). Lastly we extracted several *test* networks O of different sizes from the remaining dataset. As the Steam dataset did not have pictures, we then added public pictures from Facebook's users who elected to make their Facebook profiles public. We crawled 100 sets of pictures from 100 randomly selected users from this directory and saved them anonymously. Each set contained 3-8 pictures from a specific profile. We used 50 sets for our *training* dataset and the other 50 sets for the test dataset. For each two pictures in each of the picture datasets (training/test), we calculated the matching confidence probability using Betafaceapi, a free web service for face detection and recognition package (http://betafaceapi.com/). Most of the pictures (80%) contained only 1 figure, while the others contained 2 or more figures. In this case, we used the highest calculated matching confidence probability.

Experimental Setup The experimental setup began by sampling a network from the dataset. This sampling was repeated 10 times, each starting from a different random node, in order to avoid randomization errors and biases, creating 10 different networks for each size. In the following step, K nodes were randomly selected as missing nodes. Each one of these nodes was then randomly associated with one of the picture sets. This was repeated 10 times from each one of the networks, resulting in different random instances of the problem setting for each combination of graph sizes and K. The randomly selected missing nodes were removed from each instance. Each missing node was replaced with a placeholder for each of its links to remaining nodes in the network. Each such placeholder was associated with one picture of the removed node. If the node had fewer pictures than numbers of links, some of the placeholders would not have an associated picture (on average only 54% of the placeholders had an attached picture). The resulting graph was inputted into the algorithm, which produced a clustering of the placeholders. By uniting the placeholders in each cluster into a new node and connecting the new node to the neighbors of the placeholders in the corresponding cluster, the algorithms created a predicted network. The goal of these algorithms was to predict a network that resembled the structure of the original network from which the random nodes were removed. The clustering produced by the algorithms was evaluated using the purity measure described above, and the average purity value achieved for all of the instances and/or for the instances of a specific K was reported. This result was also averaged over the different instances and/or for the instances of a specific K and was reported as well.

The Comparison Algorithm We compared our algorithm to a baseline, denoted **MIWS** (**M**issing node **I**dentification using a **W**eighted **S**um), which is a generalization of the previously published SAMI-A algorithm (Sina, Rosenfeld, and Kraus 2013)[1]. MIWS calculates an *affinity* measure based on a weighted sum between the 3 types of similarity measures: structure, attributes and pictures. Formally, we define the matrix of affinity measures MA^{RCN} using the RCN measure as: $MA^{RCN}_{ij} = (1-w_a-w_p)*RCN_{ij}+w_a*ATT_{ij}+w_p*PIC_{ij}$ where w_a is an input parameter that represents the relative weight of the attributes' similarity measure and w_p is an input parameter that represents the relative weight of the pictures' similarity measure. Similarly, we define MA^{AA}_{ij} for the AA measure and MA^{RCN+AA}_{ij} with the average of the RCN and AA measures. Here we also considered different subsets of similarity measures, thus overall there were 12 instances of the MIWS algorithm: 4 data types (S,SA,SP and SAP) and 3 structure similarity types (RCN,AA and RCN+AA).

Training the Algorithms

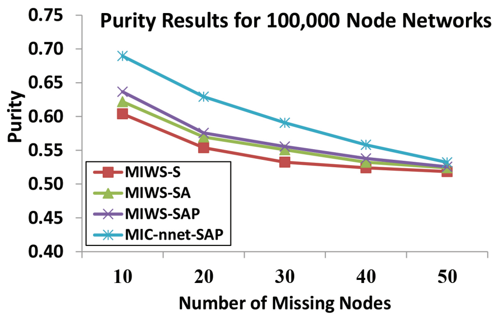
We used the training datasets to empirically learn the domain dependent classifiers for MIC and the domain optimal weights w_a and w_p for the comparison algorithm MIWS.

The Classifiers For the MIC algorithms, we used the *training* datasets to learn the 5 types of classifiers (tree, nnet, kNN, AdaBoost and Bayes) with the 4 data types (S,SA,SP and SAP) and 3 structure affinity types (RCN,AA and RCN+AA). Using the 2K *training* sample networks and the 50 *training* sets of pictures, we calculated the 5 affinity measures (RCN, AA, RCN+AA, ATT and PIC) for the 50 missing nodes configuration and saved the data as a list of rows as follows. For each two placeholders, i, j (i≠j), the row includes the affinity measures (RCN_{ij}, AA_{ij}, $RCN+AA_{ij}$,ATT_{ij}, PIC_{ij}) and an additional I_{ij} value of 1 or 0, which indicates whether or not the two placeholders originated from the same node. We repeated the run 10 times over the six training networks. As this data was unbalanced (only 2% of the data have an indicator I_{ij} of value 1), we undersampled the rows with an indicator I_{ij} of value 0, using a uniform probability of 5%. This undersampling produced data in which 28% of the rows had an indicator I_{ij} of value 1. We used this data to train and save the 60 variations of classifiers (using Matlab 2013). To evaluate these classifiers, we generated test data using an additional 2 runs over the six training networks. This time we did not undersample the data. We found that the average of the accuracy and precision (for class 0) measures were usually (all excluding kNN) high (over 95%), while the recall values (for class 1) were between 48%-65%.

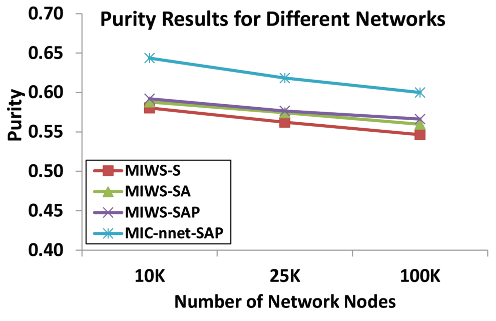
The Weight Parameters In the MIWS algorithms, the weights w_a and w_p are used for the weighted sum between the structure affinity, the attributes affinity and the pictures affinity. We ran the MIWS algorithm with 3 data types (SA,SP and SAP) and 3 structure similarity types (RCN,AA and RCN+AA) using the 2K *training* sample networks, the same 5 missing node values (10, 20, 30,

[1]We chose not to compare MIC to the original MISC algorithm, which uses spectral clustering, nor to KronEM, another recently developed algorithm which only uses the network graph structure, as our preliminary tests showed that MIWS outperformed both of these algorithms.

40 and 50) and a range of weights between 0.2 and 0.8 with steps of 0.1. We repeated the run 10 times over the six training networks we had. Based on the *training* results, the best results for the SA and SP data types were achieved with w_a=0.5 for MIWS-SA-RCN and w_a=0.6 for both MIWS-SA-AA and MIWS-SA-RCN+AA, w_p=0.5 for both MIWS-SP-RCN and MIWS-SP-AA and w_p=0.4 for MIWS-SA-RCN+AA. For the SAP data type, the best results were achieved with w_a=0.3, w_a=0.3 for MIWS-SAP-RCN and w_a=0.5 and w_a=0.2 for both MIWS-SAP-AA and MIWS-SAP-RCN+AA.



(a) Different missing nodes



(b) Different network sizes

Figure 2: The purity results with the RCN+AA measure.

Evaluation and Results

We evaluated the two sets of configurations for the MIC and MIWS algorithms. We proceeded to compare the 10 networks of sizes 10K, 25K and 100K described above where we randomly removed sets of 10, 20, 30, 40, and 50 missing nodes. We repeated each configuration 10 times to obtain 100 results for each one of the missing nodes' values. Table 1 shows the average results for all missing node values for the different algorithm variations for the 10K, 25K and 100K node networks. MIC-nnet-SAP for all 3 variations of structure affinity (with no significant difference found between their results) outperformed MIWS and also all other MIC variations for all network sizes. For example, the result for the 100K node networks of RCN+AA MIC-nnet-SAP was found to be significantly better compared to RCN+AA MIWS-SAP (the ANOVA results at 0.05 is 4.75E-54). The results also confirm our claim that the additional pictorial information can improve the missing node identification compared to the previously known algorithms (MIWS-S and MIWS-SA) as depicted in Figure 2. This improvement is also true for MIC. For example, the results with the AA measure of MIC-nnet-SAP and MIC-nnet-SP were found to be significantly better than MIC-nnet-SA and MIC-nnet-S (the ANOVA results at the 0.05 level are p=4.49E-22, 2.78E-53, 0.16E-2 and 2.48E-22, respectively). Moreover, the results also show that HACC outperformed the previous algorithm even when a single, same feature such as the structure feature is exclusively considered. Table 1 shows that the MIC-nnet-S provided better results than MIWS-S for the 3 variations of structure affinity.

The Authorship Identification Problem

To demonstrate the generality of HACC, we chose to implement it on a second *authorship identification* task. Similar to the *missing node identification* problem, in this task there also were objects that needed to be identified as originating from a person. However, in this task instead of finding missing nodes, we needed to identify which anonymous documents originated from a given person.

Overview and Background Authorship analysis has a long history and the problem of clustering a set of anonymous documents according to authorship has been extensively explored (Stamatatos 2009). While the research originally started by focusing on literary works of disputed or unknown authorship, it also has been extended to web applications for purposes such as verifying the authorship of emails, blogs and web talkbacks (Cheng, Chandramouli, and Subbalakshmi 2011). Recent work (Layton, Watters, and Dazeley 2013) tried to cluster multiple documents, each of which written by a single author, while other work (Akiva and Koppel 2013) focused on a harder problem that aimed to segment a single unsegmented document according to authorship, with no guarantee that any unit longer than a single sentence was written by a single author. Other works also considered the problem of author identification or decomposing a document according to authorship, but their methods were supervised (Stamatatos 2007; Graham, Hirst, and Marthi 2005).

Problem Definition We studied the previously proposed author-clustering problem variation (Layton, Watters, and Dazeley 2013; Akiva and Koppel 2013). Briefly, given a collection of unlabeled texts taken from the works of multiple authors, our objective was to divide the texts of the respective authors into sets, given the number K of authors.

Domain Features We considered the M words that appear in the greatest number of different articles in the combined corpus as previously done (Akiva and Koppel 2013). We used two variations of this vector, one with a binary representation of a text and the other with its frequency. Accordingly, we calculated two similarity features for each pair of texts: one based on the binary vectors (Ratio) and the other based on the vectors with the word frequency (Cosine).

The AI-HACC Algorithm For the HACC implementation, we first found the M words that appeared in the greatest number of different articles and for each pair of texts we calculated the Ratio and the Cosine measures. We tuned the classifier in the same way we did for the *Missing Node Identification*, using a different *training* dataset (described next),

Table 1: The purity results for the 10K, 25K and 100K node networks.

	Data Type	S			SA			SP			SAP		
	Algorithm	RCN	AA	RCN+AA	RCN	AA	RCN+AA	RCN	AA	RCN+AA	RCN	AA	RCN+AA
10K	MIWS	0.5921	0.5666	0.5804	0.5986	0.5767	0.5879	0.5984	0.5729	0.5866	0.6029	0.5810	0.5921
	MIC-tree	0.5933	0.5887	0.5904	0.5922	0.5895	0.5900	0.6286	0.6203	0.6259	0.6279	0.6219	0.6278
	MIC-nnet	0.5956	0.5946	0.5950	0.6012	0.6017	0.5997	0.6397	0.6317	0.6409	**0.6423**	**0.6412**	**0.6437**
	MIC-kNN	0.5687	0.5672	0.5730	0.4880	0.4851	0.4921	0.4485	0.4495	0.4532	0.5139	0.5087	0.5139
	MIC-AdaBoost	0.5927	0.5864	0.5898	0.5933	0.5798	0.5912	0.5123	0.5087	0.5154	0.5186	0.5140	0.5193
	MIC-Bayes	0.5853	0.5412	0.5860	0.5835	0.5297	0.5851	0.5860	0.5448	0.5880	0.5837	0.5360	0.5861
25K	MIWS	0.5675	0.5497	0.5622	0.5762	0.5652	0.5743	0.5809	0.5570	0.5715	0.5812	0.5693	0.5765
	MIC-tree	0.5729	0.5739	0.5705	0.5723	0.5728	0.5702	0.5991	0.5967	0.5993	0.6009	0.6011	0.6008
	MIC-nnet	0.5773	0.5778	0.5774	0.5868	0.5868	0.5864	0.6149	0.6080	0.6139	**0.6185**	**0.6174**	**0.6182**
	MIC-kNN	0.5481	0.5541	0.5557	0.4617	0.4613	0.4628	0.4255	0.4274	0.4266	0.4815	0.4824	0.4817
	MIC-AdaBoost	0.5718	0.5692	0.5693	0.5700	0.5625	0.5692	0.4788	0.4798	0.4793	0.4883	0.4856	0.4878
	MIC-Bayes	0.5613	0.5044	0.5603	0.5588	0.5016	0.5601	0.5600	0.5074	0.5606	0.5600	0.5089	0.5610
100K	MIWS	0.5475	0.5369	0.5465	0.5545	0.5545	0.5597	0.5670	0.5520	0.5625	0.5601	0.5551	0.5662
	MIC-tree	0.5557	0.5581	0.5529	0.5543	0.5577	0.5537	0.5755	0.5760	0.5744	0.5795	0.5842	0.5791
	MIC-nnet	0.5616	0.5633	0.5607	0.5753	0.5794	0.5751	0.5927	0.5863	0.5909	**0.5989**	**0.6010**	**0.5999**
	MIC-kNN	0.5251	0.5359	0.5308	0.4340	0.4406	0.4343	0.3984	0.4040	0.4021	0.4461	0.4494	0.4472
	MIC-AdaBoost	0.5550	0.5543	0.5453	0.5470	0.5439	0.5488	0.4530	0.4563	0.4531	0.4635	0.4653	0.4632
	MIC-Bayes	0.5292	0.4640	0.5293	0.5287	0.4649	0.5290	0.5310	0.4677	0.5291	0.5296	0.4704	0.5299

and the classifier *confidence* score as the affinity measure for the K-means clustering algorithm.

The Dataset and the Experimental Setup For our experiments, we used texts on the same topic from the Reuters Corpus Volume 1 (RCV1). Specifically, we used the Reuter-50-50 dataset[2], which contains an online database with 5000 classified articles, 100 articles of 50 authors, divided into training and test sets for each author. One advantage of this dataset was its generality. It had been previously used in web classification tasks such as gender identification in e-mails, blogs and chatrooms (Cheng, Chandramouli, and Subbalakshmi 2011).

Our algorithm used 10 authors as a *training* dataset, $\hat{O}$, to learn the classifiers, and the other 40 authors as the *test* dataset, O, for comparison. The experimental setup began by choosing K authors randomly. It then chose 2-8 random texts for each of the K chosen authors. These selected texts were then delivered to the different algorithms. Finally we calculated the purity of the clustering results for each one of the algorithms according to their labels in the dataset.

The Comparison Algorithm We compared `HACC` to the algorithm presented in (Akiva and Koppel 2013) for the author-clustering problem. Their algorithm, which we denote `AkivaKoppel`, considers only the M words that appear in the greatest number of different articles in the combined corpus and uses only the binary vector representation of a text. Their algorithm takes the binary vectors as input, calculates the similarity measure between each pair of texts and delivers it to the n-cut clustering algorithm.

Evaluation and Results We repeated the experiment 40 times with K=10,20,30,40, which produced 160 different problem instances. We tested 3 configurations using the most common M words (M=500,1000,1500) that appear across the articles. For the `HACC` algorithm, we used a neural network classifier which previously provided the best results. Note that in the 1500 word configuration, the features calculation in `HACC` was done using all of the words. Table 2 shows the average purity results over all the runs with the different selection of the top words for the `AvikaKoppel` and `HACC-nnet`. Overall, `HACC-nnet` outperformed `AvikaKoppel` and demonstrated that the additional step of tuning a classifier in the *training* set can improve the authorship identification solution.

[2]Dataset creator and donator: Zhi Liu, liuzhi8673@gmail.com, National Engineering Research Center for E-Learning, China.

Table 2: Author identification purity results.

Algorithm	500	1000	1500*
AvikaKoppel	0.5058	0.5340	0.5418
HACC-nnet	0.5392	0.5617	**0.5786**

Conclusions and Future Work

This paper represents, to the best of our knowledge, the first work to combine supervised classifiers on clusters different than the clusters of the unlabeled objects as a preprocessing step before activating the clustering algorithm in order to divide a group of objects into K separate sets. The general `HACC` approach is particularly suitable when additional information is available which can be incorporated in a quick and efficient way. We believe that we also are the first to study the *missing node identification* problem by including information from pictures (which become more ubiquitous) together with the network structure and the attribute information of known nodes. Including pictorial data introduces a new research challenge as it requires a model that can handle missing features as many nodes have no pictures. The main key contributions of this paper are: (i) demonstration of how the novel `HACC` approach first utilizes a classifier as a preprocessing step in order to create one combined *affinity* component measure and (ii) showing that this general algorithm can be used to outperform known algorithms in web learning tasks such as the *missing node identification* problem and the *authorship identification* task. In the future, we would like to further explore the potential of our novel approach with other datasets, different domains and different similarity measures.

Acknowledgment

This research is based on work supported in part by MAFAT and ISRAEL SCIENCE FOUNDATION (grant #1488/14).

References

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

Akiva, N., and Koppel, M. 2013. A generic unsupervised method for decomposing multi-author documents. *Journal of the American Society for Information Science and Technology* 64(11):2256–2264.

Basu, S.; Bilenko, M.; and Mooney, R. J. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 59–68. ACM.

Becker, R.; Chernihov, Y.; Shavitt, Y.; and Zilberman, N. 2012. An analysis of the steam community network evolution. In *Electrical & Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of*, 1–5. IEEE.

Bishop, C. M. 2006. *Pattern recognition and machine learning*, volume 1. Springer New York.

Cheng, J.; Adamic, L.; Dow, P. A.; Kleinberg, J. M.; and Leskovec, J. 2014. Can cascades be predicted? In *Proc of the 23rd international conference on WWW*, 925–936. International World Wide Web Conferences Steering Committee.

Cheng, N.; Chandramouli, R.; and Subbalakshmi, K. 2011. Author gender identification from text. *Digital Investigation* 8(1):78 – 88.

Eyal, R.; Rosenfeld, A.; Sina, S.; and Kraus, S. 2014. Predicting and identifying missing node information in social networks. *To Appear at ACM Transactions on Knowledge Discovery from Data (TKDD).*

Eyal, R.; Rosenfeld, A.; and Kraus, S. 2011. Identifying missing node information in social networks. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Freno, A.; Garriga, C., G.; and Keller, M. 2011. Learning to Recommend Links using Graph Structure and Node Content. In *Neural Information Processing Systems Workshop on Choice Models and Preference Learning*.

Gong, N. Z.; Talwalkar, A.; Mackey, L. W.; Huang, L.; Shin, E. C. R.; Stefanov, E.; Shi, E.; and Song, D. 2011. Predicting links and inferring attributes using a social-attribute network (san). *CoRR*.

Goyal, A.; Bonchi, F.; and Lakshmanan, L. V. 2010. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, 241–250. ACM.

Graham, N.; Hirst, G.; and Marthi, B. 2005. Segmenting documents by stylistic character. *Natural Language Engineering* 11(04):397–415.

Hauff, C., and Houben, G.-J. 2012. Placing images on the world map: a microblog-based enrichment approach. In *Proc of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 691–700. ACM.

Jain, A. K. 2010. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters* 31(8):651–666.

Kim, M., and Leskovec, J. 2011. The network completion problem: Inferring missing nodes and edges in networks. *SIAM International Conference on Data Mining (SDM), 2011.*

Kuncheva, L. I. 2004. *Combining pattern classifiers: methods and algorithms.* John Wiley & Sons.

Layton, R.; Watters, P.; and Dazeley, R. 2013. Automated unsupervised authorship analysis using evidence accumulation clustering. *Natural Language Engineering* 19(01):95–120.

Leskovec, J., and Faloutsos, C. 2006. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 631–636. ACM.

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

Ng, A. Y.; Jordan, M. I.; and Weiss, Y. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 849–856. MIT Press.

Nguyen, T. T., and Armitage, G. 2008. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys & Tutorials, IEEE* 10(4):56–76.

Shrot, T.; Rosenfeld, A.; Golbeck, J.; and Kraus, S. 2014. Crisp: an interruption management algorithm based on collaborative filtering. In *CHI*, 3035–3044.

Sina, S.; Rosenfeld, A.; and Kraus, S. 2013. Solving the missing node problem using structure and attribute information. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 744–751. ACM.

Stamatatos, E. 2007. Author identification using imbalanced and limited training texts. In *Database and Expert Systems Applications, 2007. DEXA'07. 18th International Workshop on*, 237–241. IEEE.

Stamatatos, E. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology* 60(3):538–556.

Strehl, A., and Ghosh, J. 2003. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* 3:583–617.

Yin, Z.; Gupta, M.; Weninger, T.; and Han, J. 2010. Linkrec: a unified framework for link recommendation with user attributes and graph structure. In *Proceedings of the 19th international conference on World wide web*, 1211–1212. ACM.