

# Multi-Network Fusion for Collective Inference

Hoda Eldardiry and Jennifer Neville  
Department of Computer Science  
Purdue University  
{hdardiry | neville}@cs.purdue.edu

## ABSTRACT

Although much of the recent work in statistical relational learning has focused on homogeneous networks, many relational domains naturally consist of multiple observed networks, where each network *source* records a different type of relationship between the same set of entities. For example, data about organizations may contain both an email communication network and a network of coworker ties. Since collective classification models rely on propagating information throughout the relational network to improve predictions, multi-network methods will need to consider how to best combine relational information from various link sources. There are two opportunities to combine multi-source link information for relational classification: *data fusion* methods combine the available information during learning, while *classification fusion* methods learn and apply models independently on each network source, then combine model predictions during inference. Past work has focused primarily on data fusion techniques, where features and/or links from various sources are combined. However, as the number of links, sources, and/or features increases, this approach can lead to high variance in the learned model, and can also increase the amount of noise propagated during inference, which will degrade performance. In this work, we focus on classification fusion, which overcomes these limitations by learning independent models to reduce variance. In addition, we develop a novel approach to *collective fusion*, which interleaves the learned models during collective inference. We evaluate our methods on synthetic and real-world social network data, showing that collective fusion significantly outperforms other methods over a wide range of conditions.

## 1. INTRODUCTION

The majority of statistical relational learning techniques have been developed under the assumption of a single-source network environment where the relationships among objects are completely observed in the training set. In particular many collective inference methods assume a homogenous network

with a single type of entity (e.g., people) and a single type of relationship (e.g., friendships) (see [9]). Although a number of relational techniques are applicable to heterogeneous networks with multiple types of entities and relationships (e.g., [5, 10, 11]), the different link types often represent relationships between different entities, rather than multiple relationships between the same set of entities. For example, in citation data there can be nodes corresponding to authors, papers, and venues, with links corresponding to citation, coauthor, and publication relationships. Therefore, past work has focused on developing accurate learning and inference algorithms that are suitable for analyzing single source networks.

However, in many real-world domains, particularly social network domains, relationship data about the *same* set of nodes may be collected from and/or stored in separate *sources*, where each source may record observations for only a subset of the relationships. For example, consider online social network domains where email communications, friendship nominations, group memberships, photo tagging, recommendations, etc. represent different types of observed relationships among people. These *transactional* networks (e.g., email, photo tagging) can be viewed as different sources of social network information, which offer alternative evidence of the “true” underlying social network.

In this work, we focus on learning and inference methods that can exploit the full range of link information available in these types of *multi-source* network domain. Due to the wide range of link strengths in any one observed source (e.g., email), it is likely that noise in the observed links will reduce the accuracy of relational models learned from any single source. We conjecture that models that jointly consider network information gathered from multiple sources, will be able to offset the noise observed in any one source and thus result in more accurate predictions. Moreover, since collective classification methods use inferences about one object to improve inferences about other related objects [15], combining relational information from the various link sources during inference, will enhance the propagation of information throughout the relational graph structure which should improve predictions even further.

In particular, we consider the problem of relational learning and collective classification for a single prediction task over a network dataset that consists of multiple link sources. In this scenario, there are two opportunities to combine vari-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MLG '10, July 24-25, 2010 Washington, DC, USA

Copyright © 2010 ACM 978-1-4503-0214-2/10/07... \$10.00

ous link information. The first approach, which we will call *data fusion* combines the available information during learning (i.e., a model is learned from the joint data set). The second approach, which we will call *classification fusion* considers each link source independently for learning, and then combines the model predictions during inference.

Past work on combining multi-source graphs has primarily focused on data fusion (e.g., [1, 7, 8, 17]). Because multi-source data can naturally be represented as a heterogeneous network (i.e., with multiple link types), this makes it relatively easy to apply many relational modeling techniques in a data fusion approach. Data fusion methods combine multiple link sources into one graph—either by merging the links into a single source, or explicitly modeling the different link types—and learn a single model from the resulting graph. While this approach is simple and efficient, it suffers from a number of limitations. First, aggregating features and links from multiple sources can result in a graph with too many features and/or too many links. This can increase the variance of the learned model, which will increase error. Second, merging link sources can increase the number of edges used during collective inference. This can increase the propagation of noise during the inference process and degrade the accuracy of the model.

Classification fusion is an alternative approach that can overcome the limitations of data fusion and improve prediction accuracy in multi-network domains. Instead of directly combining the sources before learning, classification fusion methods learn an ensemble of models, one for each link source. Then the ensemble of models are combined during inference to reduce the variance of the predictions and improve prediction accuracy.

For independent inference (i.e., non-relational) settings, classification fusion methods resemble conventional ensemble approaches. Models are learned from each source independently, then applied independently to each test set instance, and finally the predictions are aggregated to make classification decisions. We refer to this approach as *decision fusion*. In contrast to independent inference methods, *collective inference* methods (see e.g., [12]) jointly infer labels of related instances in network settings, using inferences about one object to improve inferences about other related objects. Collective inference offers a unique opportunity to explore a novel type of classification fusion. In addition to propagating inferences across nodes in the network, we can also propagate inferences across *models*.

In this work, we compare two data fusion methods (*Feature Fusion* and *Link Fusion*) and two classification fusion methods (*Decision Fusion* and *Collective Fusion*). We propose a novel collective fusion algorithm that learns an ensemble of models, one for each link source, and applies the models simultaneously for collective inference. During the inference process, to predict the label of a particular object, all current model predictions for that object are aggregated. In other words, the model predictions are propagated during the collective inference process to improve *intermediate* predictions, rather than waiting and aggregating only after inference is complete for each model.

We evaluate the four approaches on synthetic and real world datasets using a relational dependency network (RDN) collective inference model [12]. We include a *Single Source* model as a baseline for comparison, which corresponds to “no fusion”. The results indicate that *Collective Fusion* leads to significant improvement in model performance over all other approaches and that the improvement increases as autocorrelation, linkage or number of sources increases. In addition, *Collective Fusion* is the method that is most robust to missing labels.

## 2. FUSION FRAMEWORK

Past work uses the term “data fusion” to refer to the process of combining data from multiple sources such that the resulting information is “better” than using the sources individually. We categorize a machine learning approach that combines information from multiple sources to learn a single model, and then applies the learned model on the combined information, as a *data fusion* method. We categorize a machine learning approach that learns an ensemble of models, one from each source, and then combines the predictions from the ensemble during inference, as a *classification fusion* method.

In this section we outline a framework where we compare and contrast the relational fusion methods we propose. Given multiple link sources that represent different types of relationships between the same set of objects, the goal is to combine link information provided by the various sources to improve the quality of inferences for a collective classification task.

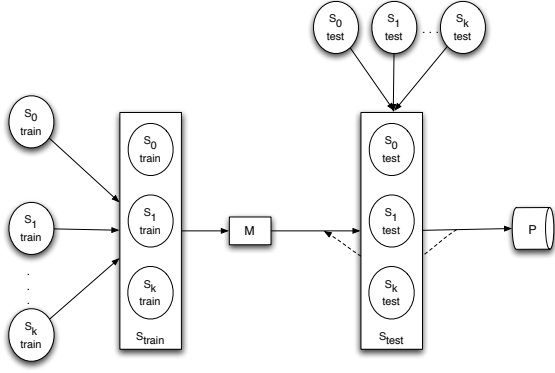
### 2.1 Problem description

From a network of objects described by  $k$  link sources  $\{S_1, S_2, \dots, S_k\}$  where the graph  $S_i = \{V_i, E_i\}$  has  $V_i$  nodes and  $E_i$  edges, we are given a set of  $k$  training graphs  $\{S_{1train}, S_{2train}, \dots, S_{ktrain}\}$  and  $k$  testing graphs  $\{S_{1test}, S_{2test}, \dots, S_{ktest}\}$ . In this initial work, we assume that  $\forall_{i \neq j} V_{itrain/test} = V_{jtrain/test}$ . Nodes correspond to entities, and edges correspond to relations among those entities. Our assumption is that each of the  $k$  sources of information represents a different type of relationship between the same set of entities.

Within this framework, we consider an *across-network* collective classification task. Each node is associated with a binary class label  $Y$ . In addition there may be a set of node features  $\mathbf{X}_V$  and a set of source-specific edge features  $\mathbf{X}_E^{S_i}$ . We train the model on fully labeled training networks and apply the learned models on partially-labeled (disjoint) testing networks in a collective classification manner. The output is a set of marginal probability distributions (i.e., predictions) over the nodes of the testing graphs. The problem can be approached as data fusion or classification fusion. We describe both below.

### 2.2 Data Fusion

In data fusion approaches, the goal is to use the available training graphs to learn a single, joint model, then apply the learned model on the testing graphs to produce a single set of predictions over the nodes. We present two data fusion methods below: *Feature Fusion* and *Link Fusion*.



**Figure 1: Graphic illustration of feature fusion process; solid lines indicate the flow of data, dashed lines indicate the collective inference process.**

### 2.2.1 Feature Fusion

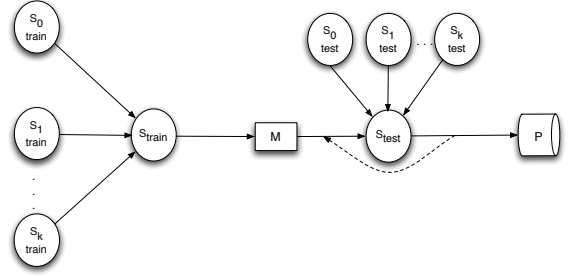
*Feature Fusion* is a feature aggregation approach that is applied during the learning process, using a data representation that maintains the link type information. A single model  $M$  is learned from the set of  $k$  training graphs  $\{S_{1_{train}}, S_{2_{train}}, \dots, S_{k_{train}}\}$ . Link *type* features distinguish which link source they belong to, and these serve to differentiate relationships among the entities in the model. For example, object  $A$  may be linked to node  $B$  through a link of type  $S_1$ , but  $A$  may link to  $C$  and  $D$  through links of type  $S_2$ . In this approach, if link features exist, they are associated with the appropriate source in the learning process.  $M$  is applied to  $\{S_{1_{test}}, S_{2_{test}}, \dots, S_{k_{test}}\}$ . Figure 1 illustrates how *Feature Fusion* works. Including the link type information allows the model to distinguish between links based on their sources. This provides more information to the learning process. However, the model may suffer from high variance due to incorporation of a large amount of features during learning (i.e., overfitting). As a possible solution to overcome overfitting, we propose *Link Fusion*, which we describe next.

### 2.2.2 Link Fusion

*Link Fusion* summarizes the data in a single, simple-graph representation by aggregating the links from all the sources to form one graph. This method simplifies the resulting graph by ignoring the link type information that distinguishes the source of the observed link.

The input is a set of  $k$  training graphs  $\{S_{1_{train}}, S_{2_{train}}, \dots, S_{k_{train}}\}$  and the method forms one *merged* training graph  $S_{train} = \{V, E\}$ . Here  $E = \{E_1 \cup E_2 \cup \dots \cup E_k\}$  is the set of unweighted, undirected edges representing an untyped relationship between objects in  $V$ . The union operation is defined such that  $E$  has an untyped edge  $e = [u, v]$  if there exists an edge  $e = [u, v]$  in at least one of  $\{E_1, E_2, \dots, E_k\}$ .  $S_{test}$  is formed using the same process as  $S_{train}$ . One model  $M$  is learned on  $S_{train}$  and applied on  $S_{test}$  to output  $P$ , a set of probability estimates for the predictions on nodes in  $S_{test}$ . Figure 2 illustrates the *Link Fusion* method.

In *Link Fusion*, links are fused using a simple union operation and the resulting links are untyped. The output graph



**Figure 2: Graphic illustration of link fusion process; solid lines indicate the flow of data, dashed lines indicate the collective inference process.**

will have a single link between objects  $A$  and  $B$ , if  $A$  and  $B$  are linked in at least one data source. Therefore, the strength of the relationship between objects connected in one graph and those connected in more than one graph are assumed equal. This assumption will not hold in most real-world scenarios. And consequently, ignoring the link types will result in some loss of information. Sources that link the same nodes will provide redundant link information so the performance can saturate as more sources are used. In addition, too many (merged) links in the resulting graph may lead to inference bias due to propagation of errors [13].

Therefore, while the two aforementioned data fusion approaches combine information from the various sources in a simple and efficient manner, their main drawback is that using more sources of information will not necessarily improve the quality of the model. Next, we present two classification fusion approaches that attempt to overcome the drawbacks of both data fusion methods.

## 2.3 Classification Fusion

In classification fusion, the goal is to use the available training graphs to learn an *ensemble* of models, one for each graph, and then apply the models simultaneously on the testing graphs to produce a set of predictions over the nodes of the testing graphs. Since the testing graphs contain the same set of nodes but different link types, this provides the opportunity to combine the model predictions in an ensemble approach to improve predictive accuracy. We present two classification fusion methods below: *Decision Fusion* and *Collective Fusion*.

### 2.3.1 Decision Fusion

*Decision Fusion* is a model aggregation approach that learns an ensemble of models on the various link sources. An ensemble of models  $\{M_1, M_2, \dots, M_k\}$  are learned on the training graphs  $\{S_{1_{train}}, S_{2_{train}}, \dots, S_{k_{train}}\}$ , one model per graph. Then the set of models are applied on the testing graphs  $\{S_{1_{test}}, S_{2_{test}}, \dots, S_{k_{test}}\}$  respectively to produce a set of probability estimates for the nodes predictions  $\{P_1, P_2, \dots, P_k\}$ . Then the predictions are aggregated to get the final predictions  $P$ , by averaging the resulting set of predictions for each node independently. Figure 3 illustrates how this approach works.

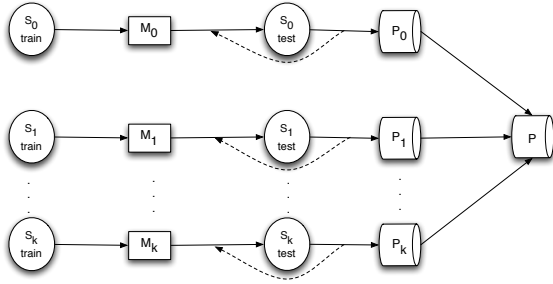


Figure 3: Graphic illustration of decision fusion process; solid lines indicate the flow of data, dashed lines indicate the collective inference process.

### 2.3.2 Collective Fusion

*Collective Fusion* is an inference aggregation approach that learns the same models as *Decision Fusion*, but aggregates predictions from the various ensembles simultaneously during the collective inference process. Collective inference methods cycle over all nodes in a given graph during approximate inference (e.g., Gibbs sampling). In this process, at each step for a given node  $v$ , a prediction is made based on the current inferences for each of its neighbors  $\mathcal{N}_v$ . Given an ensemble of models, one learned on each data source, we can apply the models simultaneously during inference. This approach provides evidence about more than just neighbor predictions during the inference process—current predictions for  $v$  from the other models are also available.

Figure 4 illustrates the *Collective Fusion* approach. An ensemble of models  $\{M_1, M_2, \dots, M_k\}$  are learned on the training graphs  $\{S_{1_{train}}, S_{2_{train}}, \dots, S_{k_{train}}\}$  and applied on  $\{S_{1_{test}}, S_{2_{test}}, \dots, S_{k_{test}}\}$  respectively to produce a set of probability estimates for the nodes predictions  $\{P_1, P_2, \dots, P_k\}$ . To calculate prediction  $p_v^{m_i} \in P_i$  for node  $v \in V_i$ , model  $M_i$  uses:

1.  $p_v^{m_i} \forall v' \in \mathcal{N}_v(E_i)$  (i.e., *collective inference*), and
2.  $p_v^{m_j} \forall P_j$  s.t.  $j \neq i$  (*collective fusion*)

In other words, the prediction for a node  $v$  and a model  $M_i$  is computed in the normal collective inference fashion (based on the current predictions for its neighbors in source  $S_i$ ) and then averaged with the most recent predictions for  $v$  from all other models  $M_{j \neq i}$  before being stored in  $P_i$ . Then the final predictions  $\{P_1, P_2, \dots, P_k\}$  are aggregated to get the output predictions  $P$  as in *Decision Fusion*.

In the next section, we evaluate the approaches we describe here in a collective classification setting, where we use relational dependency network (RDN) models [12] as  $M$ . RDNs are joint inference models that use pseudolikelihood learning techniques to estimate an efficient approximation of the full joint distribution.

## 3. EXPERIMENTAL EVALUATION

To evaluate our proposed fusion methods, we applied them on relational classification tasks in both synthetic and real data sets where the network consists of multiple link sources.

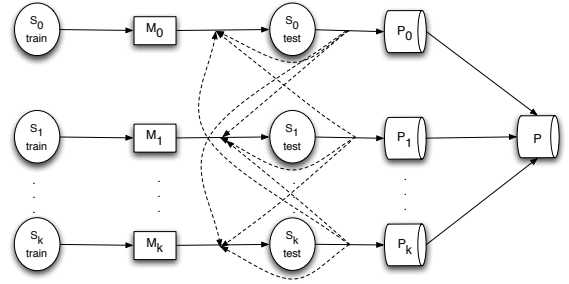


Figure 4: Graphic illustration of collective fusion process; solid lines indicate the flow of data, dashed lines indicate the collective inference process.

We learned RDN models and applied them for collective inference. The goals of these experiments are to investigate:

- The performance gains achieved by using each of the fusion methods compared to just using a single source individually.
- The effects of relational autocorrelation, linkage, missing labels, and number of sources on the performance of the various fusion methods.

The fusion approaches we investigate are *Feature Fusion*, *Link Fusion*, *Decision Fusion* and *Collective Fusion*. We compare to a “no fusion” approach, where we just use a single source to learn and apply the given model. We refer to it as the *Single Source* technique. To evaluate the methods, we generated synthetic datasets with a group structure, and we fixed the levels of autocorrelation and linkage, in the manner described in Section 3.1.1. We also evaluate the fusion approaches on a real world dataset, a subset of the facebook network which is presented in Section 3.2.1. We compare the classification accuracies by measuring area under the ROC curve (AUC).

## 3.1 Synthetic Data Experiments

### 3.1.1 Data

Our synthetic datasets are homogeneous data graphs with autocorrelation due to an underlying (hidden) group structure. Each object has a group membership  $G$  and a boolean class label  $C$ . Each group has an associated type  $T$ . We used the generative process described in Table 1 to generate a dataset with  $N_O$  objects and  $G_S$  average group size, and linkage using the setting specified below. The procedure uses a simple model where  $C$  has an autocorrelation level of 0.75. The following parameter settings were used:

$$\begin{aligned}
 N_O &= 500 \\
 G_S &= 50 \\
 p(T) &= \begin{cases} p(T=red) = 0.50 \\ p(T=blue) = 0.50 \end{cases} \\
 p(C|T_G) &= \begin{cases} p(C=1|T_G=red) = 0.90 \\ p(C=0|T_G=blue) = 0.90 \end{cases} \\
 p(E_{jk}=1|G_j, G_k) &= \begin{cases} p(E=1|G_j = G_k) = 0.40 \\ p(E=1|G_j \neq G_k) = 0.004 \end{cases}
 \end{aligned}$$

**Table 1: Synthetic data generation**


---

For each group $g$ , $1 \leq g \leq (N_G = N_O/G_S)$ :
Choose a value for group type $t_g$ from $p(T)$
For each object $i$ , $1 \leq i \leq N_O$ :
Choose a group $g_i$ uniformly in $[1, N_G]$
Choose a class value $C_i$ from $p(C T_{G_i})$
For each object $j$ , $1 \leq j \leq N_O$ :
For each object $k$ , $j < k \leq N_O$ :
Choose whether the two objects are linked from $p(E G_j = G_k)$

---

The final datasets are homogeneous—there is only one object type and one link type in each generated dataset, and each object has one class label and two attributes. After the groups are used to generate the data, we delete them from the data—the groups are not available to the classification algorithms. We generate the same set of objects for all the sources, but generate a different linkage structure and a different link type for each source.

### 3.1.2 Methodology

We generated 10 independent link sources. Each source is made up of 5 training and 5 testing datasets. Each dataset has size  $N_O = 500$  and an average group size  $G_S = 50$  as described in the algorithm in Table 1. For each experiment, we randomly label the specified percentage of the testing dataset 5 times, and we run 5 trials for each random labeling. Then we measure the area under the ROC (AUC) to assess the prediction accuracy of the model each time. Each point in the results plots represents the average of  $5 \times 5 = 25$  trials.

We test the effect of including more link sources on the fusion techniques by varying the number of sources at 1, 3, 6 and 9. Where 1 corresponds to a *Single Source* (i.e., no fusion). In this evaluation, we report results with 10% labeled nodes in the test set, autocorrelation of 0.75, and low linkage setting.

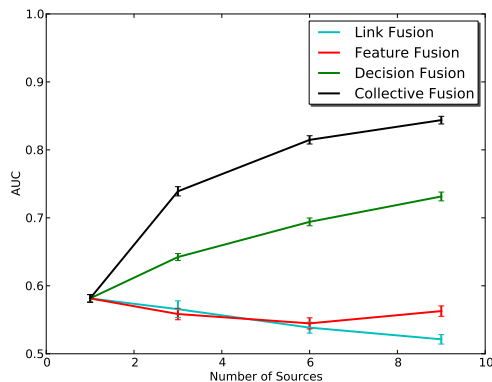
We also test the robustness of the fusion techniques to missing labels (in the test sets) by varying the proportion of labeled test data at 10%, 50% and 90%. For this experiment we report results using 3 link sources, with autocorrelation of 0.75, and low linkage.

Since collective inference in general, and the RDN specifically, have been shown to exploit relational autocorrelation and linkage in relational data [6], we investigate the effects of increasing both levels on the various fusion methods. We varied the autocorrelation level at 0.25 and 0.75 using 3 sources, each with low linkage and 10% labeled data. Then we varied the linkage level in the data from low to high, using 3 sources, each with 0.75 autocorrelation and 10% labeled data.

### 3.1.3 Results

The main finding across all experiments is that *Collective Fusion* consistently results in significantly more accurate models than all other fusion approaches.

Figure 5 shows that as more link sources are used in the fusion process, classification fusion techniques (*Collective Fusion* and *Decision Fusion*) improve overall model per-



**Figure 5: Synthetic data results as the number of network sources is varied.**

formance, where *Collective Fusion* achieves significantly higher accuracies than *Decision Fusion* ( $p < 0.01$ ). On the other hand, *Feature Fusion*’s performance saturates, while that of *Link Fusion* degrades. These results can be explained by the fact that *Decision Fusion* aggregates predictions, so as more model predictions are available, the overall predictions will improve. *Collective Fusion* can be thought of as *Decision Fusion* with an extra step before the final predictions aggregation. This extra step, in which predictions are aggregated across the models during the collective inference basically enhances the quality of predictions a step further. However, aggregating more sources at the data level will not always learn a better model. Aggregating more features in *Feature Fusion* likely increases the variance in the learned model, causing a degradation and eventually a saturation in model performance. In addition, aggregating an increased number of links in *Link Fusion* likely results in increased inference bias, due propagation of errors during collective inference, leading to continued performance degradation.

Figure 6 shows that *Collective Fusion* is the most robust fusion technique to missing labels. Again *Collective Fusion* significantly outperforms *Decision Fusion* at all label proportions ( $p < 0.01$ ). When the networks are very sparse (10% labeled nodes), both data fusion techniques do no better than the single source model. *Decision Fusion* and *Collective Fusion* however, both result in more accurate models, with *Collective Fusion* significantly outperforming *Decision Fusion*. As more data is labeled (50%), both data fusion techniques begin to improve over the single source models. And although *Decision Fusion* also improves model performance, it does not do better than data fusion. At 90% labeled sources, *Decision Fusion*, *Feature Fusion* and *Link Fusion* continue to perform better than *Single Source* models. The proportion of labeled nodes has the same effect on both data fusion techniques, and all fusion techniques perform much better than *Single Source* models when there is enough labeled data (90%). *Collective Fusion* is the most robust to missing labels. As the number of labeled instances decreases, the quality of predictions of the individual models reduces. *Decision Fusion* only averages those predictions at the end of inference, while *Collective*

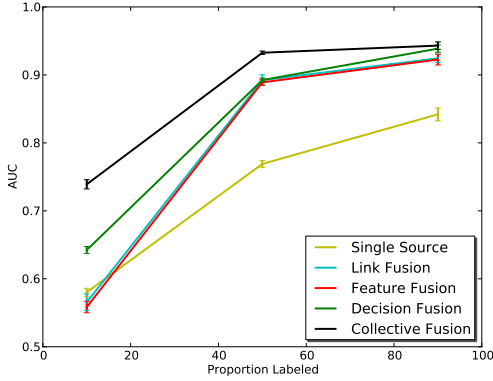


Figure 6: Synthetic data results as proportional of labeled test data is varied.

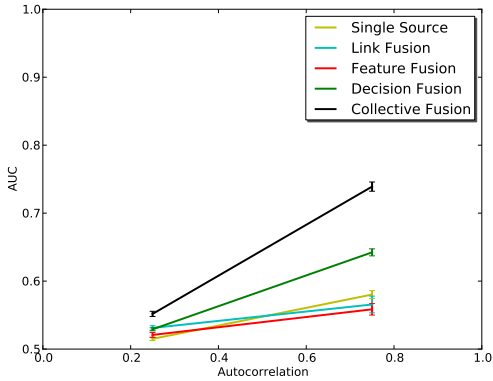


Figure 7: Synthetic data results as autocorrelation is varied.

*Fusion* exploits the given label information across models during inference, which improves the accuracy of the predictions before averaging them. This explains why *Collective Fusion* does better than *Decision Fusion* when the link sources are sparsely labeled.

Figure 7 shows that *Collective Fusion* is able to exploit relational autocorrelation more effectively than the other methods. The performance of *Single Source* models improve as autocorrelation increases, because RDNs use collective inference, which exploits autocorrelation to use predictions of related instances to improve one another. *Decision Fusion* aggregates those improved predictions and hence improves the overall predictions accuracy. *Collective Fusion* improves node predictions even further, using predictions made by other models simultaneously during collective inference. That explains why *Collective Fusion* again performs significantly better than *Decision Fusion* at both low and high autocorrelation levels ( $p < 0.01$ ). The data fusion approaches do improve as autocorrelation increases, but they are not able to exploit the increase in autocorrelation as much as the classification fusion methods.

Figure 8 shows that *Collective Fusion* also exploits in-

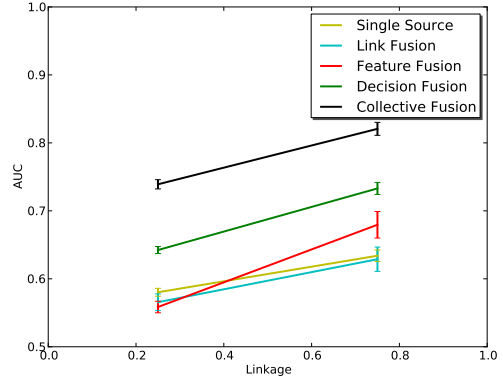


Figure 8: Synthetic data results as linkage is varied.

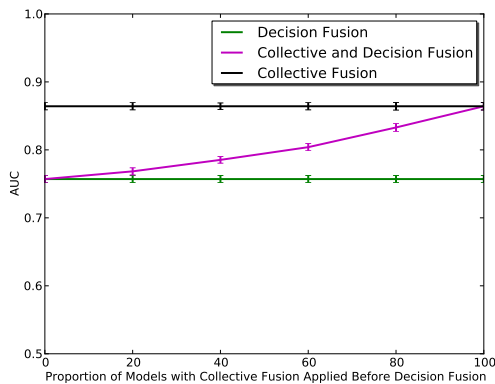
creases in linkage effectively, with similar improvements in *Decision Fusion*. This is because both approaches are directly influenced by any improvement to the performance of the *Single Source* models. Since RDN models exploit linkage to spread the information during collective inference as explained before, increased linkage improves the performance level of the *Single Source* models. Aggregating links from sources that already have high levels of linkage seems to have a redundant effect on the performance of *Link Fusion* so its performance does not increase as much as linkage increases. In contrast, aggregating features in *Feature Fusion* improves as more linkage exists. We note that *Collective Fusion* continues to significantly outperform *Decision Fusion* ( $p < 0.01$ ).

**Illustration of How Collective Fusion Improves over Decision Fusion:** *Collective Fusion* averages predictions across the various models during inference, which adds an extra level of variance reduction and increases the quality of the final predictions that are averaged at the end of inference (compared to *Decision Fusion*, which does not do the intermediate averaging). Based on this insight, we expect *Collective Fusion*'s performance to have a lower bound equal to that of *Decision Fusion*. We ran an experiment to investigate this conjecture, where we evaluated a hybrid fusion approach in which we learned an ensemble of 10 models on 10 link sources. In the hybrid approach, we always applied *Decision Fusion* to the final predictions of the ensembles. However, we varied the number of ensemble models that we applied *Collective Fusion* to during inference. When we apply *Collective Fusion* to 0 models, the approach is equivalent to *Decision Fusion*. When we apply *Collective Fusion* to 10 models, the approach is equivalent to *Collective Fusion*. In between these two extremes, the hybrid model performance shows the utility of merging predictions during inference. Figure 9 shows a smooth increment in the overall performance as the proportion of merged predictions increases, which illustrates how the two approaches relate.

## 3.2 Real Data Experiments

### 3.2.1 Data

We also evaluated our approach on data collected from the public Purdue Facebook network. Facebook is a popular on-

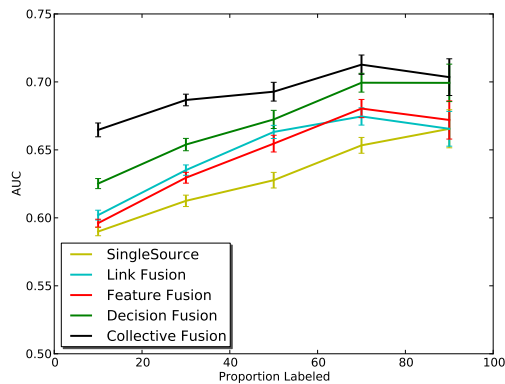


**Figure 9: Hybrid model to illustrate performance change as the amount of collective fusion is varied.**

line social network site with over 150 million members worldwide. Members create and maintain a personal profile page, which contains information about their views, interests, and friends, and can be listed as private or public. Friendship links are undirected and are formed through an invitation by one user along with a confirmation by the other. We sampled the set of 56061 public Facebook users belonging to Purdue University network in March 2008. To be affiliated with a University network, users must have a valid email account within the appropriate domain (e.g., purdue.edu), thus the members consist of students, faculty, staff, and alumni.

The public Purdue network comprised more than 3 million public friendship links among the members. Users had an average and median degree of 46 and 81 respectively. In addition to the friendship graph, we considered two transactional graphs recording interactions among friends. First, the wall graph consists of links that are extracted from users public message board on their profile page. This message board is called the wall and is a place where other users can write small messages to their friends. From the wall postings in the period 03/01/07-03/01/08, we constructed directed links in the wall graph from the sender to the receiver. Second, the picture graph consists of links that are extracted from users public photo page. The photo page can contain both photos of the member and their photo albums. The section that displays the photos of the member, consist of both photos posted by the member herself and photos posted in other users albums that are tagged as containing the member. From these tagged photos, we constructed directed links from the album owner to the member.

From these data we constructed samples of four networks: [Purdue Alum '07, Purdue '08, Purdue '09, Purdue '10] of sizes: [921, 827, 1268, 1384] users respectively. Each user has a boolean class label which indicates whether their political view is 'Conservative'. In addition, we considered nine node features and two link features. The object features record user profile information (e.g., interested in, looking for). Wall links have one link feature that counts the number of wall posts exchanged between any two users, while picture links have one link feature that counts the number of photos shared between any two users.



**Figure 10: Facebook results as proportional of labeled test data is varied.**

### 3.2.2 Methodology

The three link sources we use in our experiments are the friendship, wall and picture graphs. We train the models using all of the features, and distinguish between the features based on the link graph it comes from. We use the four Purdue networks such that we train on the two closest years to the test set; for example we train on Purdue Alum '07 and Purdue '09 and test on Purdue '08. We predict the value of the class label 'Conservative' and we vary the proportion of labeled data (10%, 50% and 90%) and measure the area under the ROC curve (AUC).

### 3.2.3 Results

Figure 10 shows results for the facebook data, that follow the observation from the synthetic data. *Collective Fusion* continues to outperform *Decision Fusion* across all levels of labeling ( $p < 0.01$  for 10-70%,  $p < 0.05$  for 90%).

## 4. RELATED WORK

Much of the past work on data fusion [1, 4, 7, 8, 17] can be categorized as *Feature Fusion*, where the main focus is to combine features. We discuss how these methods differ from our proposed *Feature Fusion* approach below.

Methods developed for applications in biological [7] and web-based datasets [17] work by compiling the data from each source into a matrix of similarity scores for each pair of entities (e.g., proteins, webpages). And then each source is weighted appropriately during aggregation of the matrices. This work differs from our *Feature Fusion* since it has focused on fusing datasets with widely varying types of information (e.g., time-series expression data, DNA strings) and the task is to assess the similarity between all pairs of entities in the data. We are instead interested in fusing *link* information (e.g., phone calls, email) between pairs of *related* entities to improve learning.

There has also been some work in the bioinformatics community on developing machine learning techniques for multi-source data. For example, for the task of gene finding the JIGSAW system [1] computes the relative weight of different sources of evidence and then combines the weighted evidence for to produce more accurate predictions than either

pipeline or single-source systems. This differs from our proposed *Feature Fusion* method because it has focused on independent predictions in data with a linear graph structure (i.e., genes in DNA sequences). Our task will require source combination in heterogeneous graph structures with interdependent predictions.

General statistical relational approaches can model heterogeneous data with multiple link and node types. For example Singh and Gordon [16] have considered multiple relations to predict links where each relation type is between different node types. This is different from our *Feature Fusion* approach because all relation types belong to the same source. In addition, each relation is between different types of nodes, so they cannot be combined to improve predictions of node attributes. Other work [8, 4] combine data from more than one source. Macskassy [8] adds in a second source of information generated from attribute similarity and combines links to improve learning. While Eliassi et. al [4] generate additional links (e.g., 2-hop paths).

The data fusion methods discussed above combine information during learning. In contrast, classification fusion methods combine models during inference. They overcome the increased learning variance by learning and applying an ensemble of models, one from each source. Past work on classification fusion [3, 14, 18] can be categorized as *Decision Fusion*, where the main idea is to combine the predictions of the ensembles to come up with more accurate overall predictions. The main difference between these methods and our proposed *Decision Fusion* is that they combine predictions of different kinds of classifiers, while our *Decision Fusion* method applies the same model, just on a different source.

For example, Shoaib et. al [14] present a decision-based fusion model to classify BRCA1, BRCA2 and Sporadic genetic mutations for breast and ovarian cancer. Different ensembles of base classifiers using the stacked generalization technique have been proposed. And a Generalized Regression Neural Networks (GRNN) is then applied to predict the mutation type based on the outputs of base classifiers. Their method then selects the best classifier and removes weak ones. Chen et. al [3] also propose an SVM classifier fusion model to combine multiple SVMs by applying the knowledge of fuzzy logic and genetic algorithms. They have applied their model to colon tumor data and ovarian cancer data.

In addition to *Decision Fusion*, we consider a novel *Collective Fusion* method which combines the predictions of the ensembles during collective inference and achieves significant performance gains over all other fusion methods. To the best of our knowledge, no previous work has considered *Collective Fusion*.

Past work in multi-view learning is also related to multi-source fusion. For example, Blum and Mitchell [2] propose a co-training approach in which two classifiers are trained on two independent sources of information. Then the training set is augmented with a subset of what one classifier labels, and the augmented set is used to train the other classifier. Their work is similar to our work in that both works assume independent sources/views of information, describing the same set of objects. However, they use inferences to

improve learning in sparsely labeled domains, while we use inferences to improve prediction in collective classification domains. The former is possible because their approach focuses on using the unlabeled data, while the latter is possible because our approach utilizes collective inference models. In future work, we will consider theoretical connections between both works in more detail.

## 5. DISCUSSION AND CONCLUSIONS

Much of the recent work in statistical relational learning has focused on analyzing single source networks, even though many relational domains naturally consist of multiple observed networks, where each network *source* records a different type of relationship between the same set of entities. Since collective classification models exploit network structure to propagate information and improve predictions, it is important to explore new ways to combine the relational information observed in the various link sources during both learning and inference.

In this paper, we outlined several alternative methods to combine relational graphs from multiple sources for collective classification. Specifically, we propose a novel *Collective Fusion* method, that is based on a unique opportunity offered by collective classification across multiple graphs—the ability to interleave and aggregate across the collective inference processes.

Our experimental results, on both synthetic and real world data, show that our proposed *Collective Fusion* method significantly outperforms the alternative fusion approaches, over *all* conditions that we investigated. Furthermore, the improvement over other fusion methods increases as auto-correlation, linkage, or the number of sources increases. Finally, *Collective Fusion* is the method that is most robust to missing labels in the test set.

## 6. ACKNOWLEDGMENTS

This research is supported by DARPA under contract number NBCH1080005. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA or the U.S. Government.

## 7. REFERENCES

- [1] J. Allen and S. Salzberg. Jigsaw: integration of multiple sources of evidence for gene prediction. *Bioinformatics*, 21(18):3596–3603, 2005.
- [2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, 1998.
- [3] X. Chen, Y. Li, R. Harrison, and Y. Zhang. Genetic fuzzy classification fusion of multiple svms for biomedical data. *Journal of Intelligent and Fuzzy Systems*, 18(6):527–541, 2007.
- [4] T. Eliassi-Rad, B. Gallagher, H. Tong, and C. Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *In Proceedings of the*

*14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

- [5] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.
- [6] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–598, 2004.
- [7] G. Lanckriet, T. D. Bie, N. Cristianini, M. Jordan, and W. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.
- [8] S. Macskassy. Improving learning in networked data by combining explicit and mined links. *Association for the Advancement of Artificial Intelligence*, 2007.
- [9] S. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8(May):935–983, 2007.
- [10] A. McGovern, L. Friedland, M. Hay, B. Gallagher, A. Fast, J. Neville, and D. Jensen. Exploiting relational structure to understand publication patterns in high-energy physics. *SIGKDD Explorations*, 5(2):165–172, 2003.
- [11] J. Neville, O. Şimşek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg. Using relational knowledge discovery to prevent securities fraud. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 449–458, 2005.
- [12] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.
- [13] J. Neville and D. Jensen. A bias/variance decomposition for models using collective inference. *Machine Learning Journal*, 2008.
- [14] M. S. B. Sehgal, I. Gondal, and L. Dooley. Support vector machine and generalized regression neural network based classification fusion models for cancer diagnosis. In *Proceedings of 4th International Conference on Hybrid Intelligent Systems*, 2004.
- [15] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.
- [16] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.
- [17] Z. Xu, I. King, and M. Lyu. Web page classification with heterogeneous data fusion. In *Proceedings of the World Wide Web Conference*, 2007.
- [18] P. A. Zhilkin and R. L. Somorjai. Application of several methods of classification fusion to magnetic resonance spectra. *Connection Science*, 8(3, 4):427–442, 1996.