The Curious Case of Stacking Boosted Relational Dependency Networks

Siwen Yan
The University of Texas at Dallas
siwen.yan@utdallas.edu

Devendra Singh Dhami
TU Darmstadt
devendra.dhami@cs.tu-darmstadt.de

Sriraam Natarajan
The University of Texas at Dallas
sriraam.natarajan@utdallas.edu

Abstract

Reducing bias while learning and inference is an important requirement to achieve generalizable and better performing models. The method of stacking took the first step towards creating such models by reducing inference bias but the question of combining stacking with a model that reduces learning bias is still largely unanswered. In statistical relational learning, ensemble models of relational trees such as boosted relational dependency networks (RDN-Boost) are shown to reduce the learning bias. We combine RDN-Boost and stacking methods with the aim of reducing both learning and inference bias subsequently resulting in better overall performance. However, our evaluation on three relational data sets shows no significant performance improvement over the baseline models.

1 Introduction

Statistical relational learning (SRL) [10, 17] combines the two formalisms of probability (capturing uncertainty) and first order logic (capturing symmetries). This has resulted in the ability to represent complex relations between objects or entities. One of the key success stories inside SRL models have been ensembles of relational regression trees that overcome the assumption of a propositional representation of the data. This model called boosted relational dependency networks (RDN-Boost) [15], has been shown to reduce the learning bias since it combines multiple prediction models while learning thereby resulting in far stronger discriminative relational models. A common assumption is that simple reduction in the learning bias is not generally sufficient to learn generalizable models especially in relational domains. A recent work presented theoretical frameworks for relational models that can reduce the bias in both learning and inference time [6]. Kou et al. [11] proposed the method of relational stacking for collective inference which is a meta learning algorithm consisting of a base learner, that can be any relational machine learning model. The base learner is augmented by expanding an instance's features with predictions on other related instances. Stacking has shown to improve collective inference by reducing the inference bias [7] and thus we hypothesized that a combination model that integrates the stacking model (reduces inference bias) with RDN-Boost (reduces learning bias) can result in better performing models.

We propose the combination of the stacking method and RDN-Boost using three relational templates which are dependent on the notion of distance between entities in the given relational data set. This is based on the observation that the entire data set can be viewed as a graph $G = (V, E)$ where V is the set of nodes denoting the entities and E is the set of edges denoting the relations between the entities. This transformation allows for efficient computation of the distances. We test our model with three relational data sets and show that, against expectations, using any of these relational templates do not

1st I Can't Believe It's Not Better Workshop (ICBINB@NeurIPS 2020), Vancouver, Canada.

improve the performance of the combination model when compared to the underlying RDN-Boost model. We speculate that the reason could be the direct dependence of the stacking model on the *strength of the underlying base model*.

We make a few key contributions: (1) We propose a combination model of relational ensemble learner RDN-Boost (lower learning bias) and stacking (lower inference bias). (2) We propose 3 relational templates for effective combination of RDN-Boost and stacking models. (3) Our empirical results demonstrate that such a combination does not yield in a statistically significant increase in performance.

2 Related work

Relational dependency networks An effective and efficient way of learning relational dependency networks (RDNs) [16] by turning learning RDNs into a sequence of relational function-approximations was introduced in [15]. The key idea was to learn each conditional distribution using a set of relational gradient-boosted trees. Given that dependency networks approximate the true joint distribution as a product of the conditionals, i.e., $P(x_1, ..., x_n) = \pi P(y_i|y\text{ß})$, one could imagine learning each conditional separately and combine them using Gibbs sampling [9]. This has the added advantage that unlike a directed probabilistic model such as a Bayesian network, one does not have to ensure acyclicity, thus resulting in faster learning and inference.

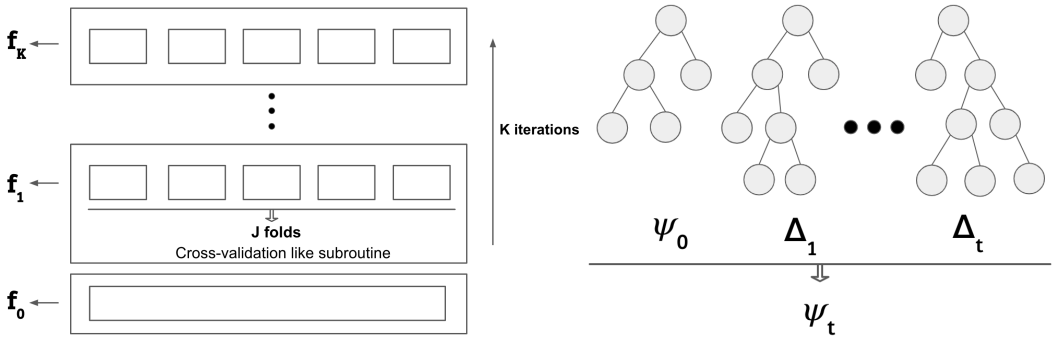
Schulte et al. [20] transformed Bayesian networks (BNs) to RDNs. RDNs have further been used for problems pertaining to hybrid domains [18], collective classification [12] and applications such as relation extraction [21].

Stacking method Kou et al. [11] introduced stacked graphical learning, which takes an efficient way of approaching the problem of collective classification by composing additional feature sets from predictions on related instances to enhance the performance of base learner. The stacked graphical learning combines the underlying approaches by Cohen [3] and Wolpert [23] to build stacks and predict labels for instances respectively. Fast et al. [7] demonstrated further that the stacked graphical learning can achieve effectiveness while being efficient. On synthetic data, RDN gets much lower learning bias while stacked graphical model gets much lower inference bias. Brophy et al. [2] combines stacked graphical learning with probabilistic graphical models to build a relational model for social network spam.

Eldardiry et al. [6] evaluated the performance of relational ensemble models via both theoretical analysis and experimental results. It is shown that relational ensemble learning approach combined with relational ensemble inference approach can target the error in both learning and inference processes resulting in a better performing model. As mentioned earlier, Natarajan et al. [15] proposed boosted relational dependency networks (RDN-Boost) by learning relational dependency networks through boosting approach, which reduces the learning bias of relational dependency network and as analyzed in [6]. Combination of gradient boosting with collective inference has been investigated in [1] focusing on continuous prediction. A recent work [19] transfers RDN-Boost by mapping predicates in learnt trees from one domain to another similar domain and modifying the mapped trees to achieve better performance.

3 Stacking Boosted RDNs

Stacking [11] can reduce the inference bias with respect to the underlying base algorithm but combining it with relational machine learning algorithms that reduce learning bias is still largely an unexplored problem. In this work, we combine the stacking method with RDN-Boost [15] as the base learner which have shown to reduce the learning bias.



(a) Learning Stacked models

(b) RDN-Boost

Figure 1: A general overview of methods used in our approach.

Algorithm 1: Stacked RDN-Boost

Stacked-RDN-B-learn($\mathbf{D_{train}}$, $\mathbf{C}$, $\mathbf{J}$, $\mathbf{K}$);
$\mathbf{D_{train}}$ is training data, $\mathbf{C}$ is relational template, $\mathbf{J}$ is the number of folds, $\mathbf{K}$ is the number of iterations;
learn a local model RDN-B_0 $\leftarrow$ RDN-B($\mathbf{D_{train}}$);
learn stacked models RDN-$B_1 \cdots$ RDN-$B_{\mathbf{K}}$;
for i=$1 \cdots \mathbf{K}$ **do**
  randomly evenly split $\mathbf{D_{train}}$ into $\mathbf{J}$ folds: $\mathbf{D_{train}}_1, \cdots, \mathbf{D_{train}}_{\mathbf{J}}$;
  for j=$1 \cdots \mathbf{J}$ **do**
    learn RDN-B$'_j$ $\leftarrow$ RDN-B($\mathbf{D_{train}} \backslash \mathbf{D_{train}}_j$);
    predict $\hat{y}^j$ = RDN-B$'_j$($\mathbf{D_{train}}_j$) ;
  end
  $\mathbf{D_{train}}'$ = $\mathbf{D_{train}}$;
  $\hat{y}$ = $\hat{y}^1 \cup \cdots \cup \hat{y}^{\mathbf{J}}$;
  ($\mathbf{N}$ is the number of instances) ;
  for n=$1 \cdots \mathbf{N}$ **do**
    new predicate of instance$_n$ = $\mathbf{C}$($\hat{y}$, instance$_n$, $\mathbf{D_{train}}$) ;
    $\mathbf{D_{train}}'$ = $\mathbf{D_{train}}' \cup$ {new predicate of instance$_n$} ;
  end
  learn RDN-B_i $\leftarrow$ RDN-B($\mathbf{D_{train}}'$);
end
Stacked-RDN-B = ensemble(RDN-B_0,RDN-B_1,$\cdots$,RDN-$B_{\mathbf{K}}$);
;
Stacked-RDN-B-infer(Stacked-RDN-B, $\mathbf{D_{test}}$);
$\hat{y}^0$ = RDN-B_0($\mathbf{D_{test}}$);
for i=$1 \cdots \mathbf{K}$ **do**
  for n=$1 \cdots \mathbf{N}$ **do**
    new predicate of instance$_n$ = $\mathbf{C}$($\hat{y}^{i-1}$, instance$_n$, $\mathbf{D_{test}}$) ;
    $\mathbf{D_{test}}'$ = $\mathbf{D_{test}}' \cup$ {new predicate of instance$_n$} ;
  end
  predict $\hat{y}^i$ = RDN-B_i($\mathbf{D_{test}}'$) ;
end
inference = $\hat{y}^{\mathbf{K}}$;
;
$\mathbf{C}$($\hat{y}$, instance, $\mathbf{D}$) ;
find all related instances {instance$_m$, $m \in \mathbf{M}$} ($\mathbf{M}$ is the set of the indices of all related instances to this instance) of this instance in the data $\mathbf{D}$ using relational templates ;
aggregate predictions $\{\hat{y}_m, m \in \mathbf{M}\}$ of those related instances to generate new predicate for this instance ;

An overview of the stacking approach is shown in figure 1a. Stacking consists of two components: the learning component and the inference component. Both of these components need to be consistent to guarantee collective classification. During the learning step of stacking, initially a local model $\mathbf{f_0}$ is learned from the given training data. After learning the local model, $\mathbf{K}$ stacked models $\mathbf{f_1} \cdots \mathbf{f_K}$ are learned in $\mathbf{K}$ iterations. For each iteration, label predictions of instances are generated by following a procedure similar to cross-validation and a new feature/predicate is created for each instance according to label predictions of its related instances. A stacked model $\mathbf{f_i}$ is learned from the training data including the new predicates for this iteration $\mathbf{i}$. In the subroutine, the training data is split into $\mathbf{J}$ folds, where, one fold $\mathbf{j} \in \mathbf{J}$ constitutes the testing data and the other folds make up the training data. A model is then trained on the training data and predict on the testing data. The predictions on the testing data become the label predictions for this fold $\mathbf{j}$. After $\mathbf{J}$ iterations, instances in each fold have their label predictions.

Figure 1b shows an overview of RDN-Boost which learns a new first-order logic regression tree Δ_j based on all previous relational regression trees $F(\psi_0, \Delta_1, \cdots, \Delta_{j-1})$. At each step, a new relational regression tree Δ_j is fitted to reduce the predicting error $(I - F)$ further from the combination of previous trees $F(\psi_0, \Delta_1, \cdots, \Delta_{j-1})$ and is run for a specified number of iterations, say $\mathbf{t}$, to learn $\mathbf{t}$ trees. The overall model is an ensemble of all the learned trees. For each gradient step, regression examples are first updated using the functional gradient $\frac{\partial log P(y_i;\mathbf{x}_i)}{\psi(y_i=1;\mathbf{x}_i)} = I(y_i = 1; \mathbf{x}_i) - P(y_i = 1; \mathbf{x}_i)$, where y_i is the target predicate, $\mathbf{x}_i$ is other predicates and $\psi(y_i = 1; \mathbf{x}_i)$ is the potential function. Then a relational regression tree Δ_j is fitted on the regression examples and the ensemble model is updated with the new fitted relational regression tree. The final model is simply a sum of these trees. Note that the trees follow single path semantics – each example will satisfy only one branch of the trees. Thus for every example, each tree returns a single regression value and all the values are then summed to get the final value.

The stacking framework requires the design of a relational template. The relational template is important for the performance of the system, since it is used to collect predictions from related examples. In [11], the collected predictions are aggregated to generate a new feature for assisting the inference. Since RDN-Boost learns and extracts logic rules from the data, a new predicate is generated for assisting the inference.

We present our method *Stacked RDN-Boost* in algorithm 1 which consists of 2 procedures describing the learning process **Stacked-RDN-B-learn** and the inference process **Stacked-RDN-B-infer**. The learning procedure, **Stacked-RDN-B-learn**, consists of first learning a local model RDN-B_0 by fitting the base learner RDN-Boost (RDN-B) on the training data $\mathbf{D_{train}}$. The stacked models, RDN-$\mathrm{B}_1 \cdots$ RDN-$\mathrm{B}_\mathbf{K}$, are then learned in K iterations. For each iteration $i = 1 \cdots \mathbf{K}$, the training data $\mathbf{D_{train}}$ is split into $\mathbf{J}$ folds where every fold forms the testing data with rest of the folds serving as training data. We then learn RDN-$\mathrm{B'}_j$ on $\mathbf{D_{train}} \backslash \mathbf{D_{train}}_j$ using the base learner RDN-B, and predict on $\mathbf{D_{train}}_j$ to get $\hat{y}^j$ for this fold $\mathbf{D_{train}}_j$. After $\mathbf{J}$ iterations and thus predictions on the $\mathbf{J}$ folds, let us denote the label predictions for all instances as $\hat{y}$. For each instance $n = 1 \cdots \mathbf{N}$, we create a new predicate by aggregating label predictions of related instances retrieved by relational template $\mathbf{C}$ on the relations (facts) in $\mathbf{D_{train}}$. The new predicates are included into $\mathbf{D_{train}}$ although only for the present iteration, say i. At the end of this iteration, we learn the stacked model RDN-B_i on $\mathbf{D_{train}}'$ (=$\mathbf{D_{train}} \cup$ {all new predicates}) using the base learner RDN-B.

For the inference process **Stacked-RDN-B-infer**, we first predict on the testing data $\mathbf{D_{test}}$ using the local model RDN-B_0 to get label predictions $\hat{y}^0$. Then, for each instance $n = 1 \cdots \mathbf{N}$, a new predicate is created via aggregating label predictions from $\hat{y}^{i-1}$ of related instances retrieved by relational template $\mathbf{C}$, as in the training phase, on the relations in $\mathbf{D_{test}}$. These new predicates are included into $\mathbf{D_{test}}$ only for this iteration i as in training. At the end of the iteration, we predict on the $\mathbf{D_{test}}'$ (=$\mathbf{D_{test}} \cup$ {all new predicates}) using the stacked model RDN-B_i, and obtain the label predictions $\hat{y}^i$. The inference result $\hat{y}^\mathbf{K}$ of the last iteration is the final inference result of our model. The relational template $\mathbf{C}$ is used to find all related instances $\{\text{instance}_m, m \in \mathbf{M}\}$ ($\mathbf{M}$ is the set of the indices of all related instances to the current instance) to the current instance via relations in the data $\mathbf{D}$, and then generate a new predicate for the current instance by aggregating over label predictions $\{\hat{y}_m, m \in \mathbf{M}\}$ of those related instances. The details of the different relational templates considered in this work are given next.

3.1 Relational template

We propose 3 relational templates to combine the methods of stacking and RDN-Boost effectively.

3.1.1 Aggregate over adjacent entities

For each entity, all other entities which appear in any relation with this entity are retrieved and their label predictions are collected. If the majority of predictions of related entities is positive, then a new predicate is created for this entity. Otherwise, no new predicate is created. This is akin to a majority vote from related entities. The predictions for entities here are inferred by the model during each cross-validation step similar to the subroutine of stacking method.

3.1.2 Aggregate over entities within fixed depth

The given relational data set can be treated as a graph $G = (V, E)$ where relations are treated as edges E and the entities are treated as nodes V. The depth is the least number of edges between two entities. For each entity, all other entities within the the depth are retrieved and their label predictions are collected. If the majority of predictions of related entities is positive, a new predicate is created for this entity. Otherwise, no new predicate is created for this entity.

3.1.3 Aggregate by Hsu-Lyuu-Flandrin-Li distance

This distance metric [4, 13] is based on the concept of m-diameter, and is given as,

$$\begin{aligned} D_m^G(a,b) = \min_Q(\max_q(\text{len}(\text{path}_q(a,b)))), \exists Q\, \forall q \in Q, |Q| \geq m, \\ \cup_{x,y \in Q} \cap_{x \neq y} (\text{path}_x(a,b), \text{path}_y(a,b)) = \{a,b\} \end{aligned} \tag{1}$$

The definition of graph $G = (V, E)$ is similar as *aggregate over entities within fixed depth*. Relations are treated as edges E and entities in the relations are treated as nodes V. The m-diameter $D_m^G(a, b)$ between node a and node b is the minimum integer d such that there are at least m internally disjoint paths of length at most d between two nodes [13]. Since the time complexity of this distance metric is very high and there is no available implementation, to simplify this distance metric, m is set to be 1 and we have a special case. Then the *Hsu-Lyuu-Flandrin-Li distance* between two nodes $D_1^G(a, b)$ is simplified to be the length of the shortest path between two nodes. Thus, for every entity e_i, breadth-first search is applied here to find the length d_j of the shortest path between entity e_i and entity e_j. The positive and negative label predictions $\hat{y}$ are mapped to +1 and -1. For each entity e_i, the inverse $1/d_j$ of distance between entity e_i and entity e_j is used as the weight w_j of entity e_j. A weighted voting $\sum_{j=1,j\neq i}^{N} w_j \cdot \hat{y}_j$ is done over all other entities. If the weighted sum is larger than or equal to 0, then a new predicate is created for entity e_i else no new predicate is created.

4 Experiments and analysis

4.1 Datasets

We consider 3 relational data sets: **Carcino**: is a biomedical data set [22] of the structures of various chemical compounds and the task is to predict if these compounds are carcinogenic in nature. **PPMI**: is a study [14] designed to identify bio-markers that impact Parkinson's and the task is to predict if a patient has Parkinson's [5]. **UW-CSE**: is a task over a representation of staff, students and faculty of 5 different computer science departments; the target is to predict whether someone is a professor.

4.2 Experiments setup

Each dataset is split into training data and testing data with a ratio of 60% - 40%. We varied relational templates, number of trees of RDN-Boost, number of iterations and number of folds of stacking method. The details of the three relational templates are displayed in the section 3. We ran the experiments on a Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz with 32 GB RAM machine. We implemented our approach in Python. For the base learner/baseline RDN-Boost, we used the srlearn python wrapper [8] and implemented the stacking approach.

Table 1: Evaluation on the Carcino data set with relational template *aggregate over adjacent entities*. The mean and standard deviation are displayed for each setting of parameters with each model running 5 times. **K** is number of iterations and **J** is number of folds of the stacking method.

		Stacked RDN-Boost (**K**,**J**)			
10 trees	RDN-Boost	(1,3)	(2,3)	(1,5)	(2,5)
Accuracy	**0.398** (0.007)	0.396 (0.003)	0.397 (0.006)	0.396 (0.004)	0.394 (0.004)
Recall	0.485 (0.020)	0.483 (0.035)	0.493 (0.031)	**0.5** (0.037)	0.490 (0.030)
Precision	0.197 (0.006)	0.195 (0.009)	0.198 (0.008)	**0.199** (0.010)	0.197 (0.008)
F1	0.280 (0.010)	0.278 (0.015)	0.283 (0.013)	**0.285** (0.016)	0.281 (0.013)
AUC-PR	0.228 (0.002)	0.228 (0.004)	0.229 (0.003)	0.222 (0.021)	**0.230** (0.002)
AUC-ROC	0.399 (0.007)	0.397 (0.014)	0.401 (0.010)	**0.407** (0.009)	0.403 (0.007)

Table 2: Evaluation on the dataset PPMI with relational template *aggregate over adjacent entities*. The mean and standard deviation are displayed for each setting of parameters with each model running 5 times. **K** is number of iterations and **J** is number of folds of the stacking method.

		Stacked RDN-Boost (**K**,**J**)			
10 trees	RDN-Boost	(1,3)	(2,3)	(1,5)	(2,5)
Accuracy	0.914 (0.008)	0.917 (0.014)	0.918 (0.006)	**0.919** (0.009)	0.915 (0.003)
Recall	0.864 (0.053)	0.868 (0.052)	0.867 (0.050)	**0.879** (0.036)	0.839 (0.020)
Precision	0.865 (0.020)	0.870 (0.005)	0.876 (0.040)	0.868 (0.014)	**0.886** (0.013)
F1	0.863 (0.018)	0.868 (0.026)	0.870 (0.010)	**0.873** (0.016)	0.862 (0.007)
AUC-PR	0.937 (0.007)	0.937 (0.009)	0.933 (0.010)	0.936 (0.007)	**0.940** (0.012)
AUC-ROC	0.971 (0.006)	0.971 (0.007)	0.970 (0.005)	0.972 (0.004)	**0.974** (0.003)

4.3 Results

We first present our results with the relational template *aggregate over adjacent entities*. The number of trees of RDN-Boost is set at 10 and the values of **K** and **J** are varied as on [(1,3), (2,3), (1,5), (2,5)], since as shown in [11], large **K** does not help the performance. And the results on the dataset UW-CSE with this relational template are not shown due to lack of space as well as because there is no significant improvement compared to RDN-Boost, which is similar in behavior with the data sets Carcino and PPMI, the results for which are shown in tables 1 and 2 respectively.

From these results it can be seen that the improvement is not significant or consistent as expected. A better relational template may be needed to exploit the strengths of stacking method. Thus, we next implemented the relational template *aggregate over entities within fixed depth*. The depth is varied, such as 2 or 3, to evaluate the model and the performance is not improved as expected. The experiment results with this relational template are omitted since there is no significant improvement in the results and are similar to previous results. Since assisting prediction from neighbors in a fixed range does not work well, the next approach may be to try an appropriate distance metric for aggregating predictions from related entities.

We next used the *Hsu-Lyuu-Flandrin-Li distance* [4, 13] for the relational template. Besides the updating of relational template, parameters of RDN-Boost and stacking method are explored more in order to capture the performance trends of our approach. For the base learner RDN-Boost, we changed the number of trees from small to large (2 to 20) in order to see the effect of different learning ability of base learner on the system. For each such setting of the number of trees, we generated results for settings of the number of iteration **K** and the number of folds **J** of stacking method as [(1,3), (2,3), (1,5), (2,5), (3,5)]. The results for the 3 data sets are shown in tables 3, 4 and 5. The results for the setting of (**K**,**J**) of stacking method = (3,5) is not shown since there is no significant difference with other (**K**,**J**) settings. As mentioned in Kou et al. [11], the stacking method converges very fast and the larger number of iterations does not raise the level of performance. The results of the number of trees as 5 are not shown since the results are also similar and the case of the number of trees as 2 can cover the cases with small number of trees.

Table 3: Evaluation on the dataset Carcino with relational template *aggregate by Hsu-Lyuu-Flandrin-Li distance*. The mean and standard deviation are displayed for each setting of parameters with each model running 5 times. **K** is number of iterations and **J** is number of folds of the stacking method.

		Stacked RDN-Boost (**K**,**J**)			
2 trees	RDN-Boost	(1,3)	(2,3)	(1,5)	(2,5)
Accuracy	0.391 (0.005)	**0.394** (0.004)	0.392 (0.003)	0.392 (0.003)	0.391 (0.007)
Recall	0.507 (0.044)	0.517 (0.044)	0.480 (0.016)	0.507 (0.044)	**0.527** (0.053)
Precision	0.199 (0.011)	0.203 (0.011)	0.194 (0.003)	0.200 (0.011)	**0.204** (0.014)
F1	0.286 (0.018)	0.291 (0.018)	0.276 (0.006)	0.287 (0.018)	**0.294** (0.023)
AUC-PR	**0.204** (0.004)	**0.204** (0.004)	0.202 (0.002)	0.203 (0.005)	0.202 (0.004)
AUC-ROC	0.397 (0.017)	**0.401** (0.017)	0.389 (0.013)	0.398 (0.019)	0.396 (0.019)
10 trees					
Accuracy	0.395 (0.003)	0.396 (0.006)	**0.397** (0.004)	**0.397** (0.0)	**0.397** (0.004)
Recall	0.466 (0.020)	0.476 (0.023)	**0.493** (0.043)	0.451 (0.0)	**0.493** (0.043)
Precision	0.191 (0.004)	0.194 (0.006)	**0.198** (0.011)	0.188 (0.0)	**0.198** (0.011)
F1	0.271 (0.007)	0.275 (0.010)	**0.282** (0.018)	0.265 (0.0)	**0.282** (0.018)
AUC-PR	0.202 (0.002)	**0.205** (0.002)	0.204 (0.002)	0.203 (0.002)	**0.205** (0.004)
AUC-ROC	0.395 (0.008)	0.403 (0.010)	0.398 (0.010)	0.395 (0.008)	**0.404** (0.013)
20 trees					
Accuracy	**0.403** (0.007)	0.399 (0.006)	0.402 (0.006)	0.399 (0.007)	0.398 (0.004)
Recall	**0.498** (0.005)	0.493 (0.043)	0.461 (0.022)	0.493 (0.031)	0.468 (0.024)
Precision	**0.201** (0.004)	0.199 (0.012)	0.192 (0.007)	0.199 (0.009)	0.193 (0.007)
F1	**0.287** (0.004)	0.283 (0.019)	0.271 (0.011)	0.283 (0.014)	0.273 (0.011)
AUC-PR	**0.206** (0.002)	0.205 (0.002)	0.205 (0.002)	**0.206** (0.003)	0.204 (0.002)
AUC-ROC	**0.406** (0.008)	0.402 (0.010)	0.400 (0.009)	0.403 (0.011)	0.397 (0.009)

Table 4: Evaluation on the dataset PPMI with relational template *aggregate by Hsu-Lyuu-Flandrin-Li distance*. The mean and standard deviation are displayed for each setting of parameters with each model running 5 times. **K** is number of iterations and **J** is number of folds of the stacking method.

		Stacked RDN-Boost (**K**,**J**)			
2 trees	RDN-Boost	(1,3)	(2,3)	(1,5)	(2,5)
Accuracy	0.878 (0.028)	0.876 (0.022)	0.876 (0.020)	**0.886** (0.016)	0.865 (0.022)
Recall	0.764 (0.123)	0.770 (0.131)	0.751 (0.140)	**0.816** (0.111)	0.707 (0.095)
Precision	0.840 (0.030)	0.836 (0.051)	**0.852** (0.052)	0.831 (0.044)	0.845 (0.028)
F1	0.795 (0.063)	0.793 (0.056)	0.789 (0.056)	**0.817** (0.046)	0.766 (0.051)
AUC-PR	0.842 (0.025)	0.839 (0.034)	0.842 (0.044)	**0.850** (0.026)	0.794 (0.040)
AUC-ROC	0.916 (0.019)	**0.921** (0.015)	0.915 (0.039)	0.915 (0.023)	0.879 (0.034)
10 trees					
Accuracy	0.917 (0.005)	**0.920** (0.013)	0.917 (0.003)	0.914 (0.008)	0.914 (0.009)
Recall	0.854 (0.028)	**0.866** (0.064)	0.859 (0.036)	0.830 (0.035)	0.836 (0.026)
Precision	0.881 (0.011)	0.882 (0.020)	0.878 (0.023)	**0.892** (0.012)	0.886 (0.019)
F1	0.867 (0.010)	**0.872** (0.026)	0.868 (0.008)	0.859 (0.016)	0.860 (0.015)
AUC-PR	0.927 (0.009)	0.937 (0.010)	0.936 (0.008)	**0.941** (0.003)	0.933 (0.011)
AUC-ROC	0.966 (0.009)	**0.972** (0.006)	0.971 (0.005)	**0.972** (0.001)	0.968 (0.006)
20 trees					
Accuracy	0.921 (0.007)	0.921 (0.008)	**0.927** (0.014)	**0.927** (0.005)	0.920 (0.006)
Recall	**0.884** (0.027)	0.85 (0.037)	0.868 (0.039)	0.866 (0.015)	0.851 (0.024)
Precision	0.869 (0.015)	0.895 (0.010)	0.898 (0.019)	**0.900** (0.005)	0.892 (0.015)
F1	0.876 (0.012)	0.871 (0.016)	**0.883** (0.024)	**0.883** (0.008)	0.871 (0.011)
AUC-PR	0.950 (0.003)	**0.954** (0.005)	0.952 (0.005)	0.949 (0.007)	0.945 (0.003)
AUC-ROC	0.977 (0.002)	**0.980** (0.002)	0.979 (0.003)	0.978 (0.003)	0.976 (0.002)

Table 5: Evaluation on the dataset UW-CSE with relational template *aggregate by Hsu-Lyuu-Flandrin-Li distance*. The mean and standard deviation are displayed for each setting of parameters with each model running 5 times. **K** is number of iterations and **J** is number of folds of the stacking method.

		Stacked RDN-Boost (**K**,**J**)			
2 trees	RDN-Boost	(1,3)	(2,3)	(1,5)	(2,5)
Accuracy	0.959 (0.014)	0.953 (0.004)	0.960 (0.012)	**0.964** (0.017)	0.951 (0.005)
Recall	0.984 (0.036)	0.984 (0.036)	**1.0** (0.0)	0.984 (0.036)	0.984 (0.036)
Precision	0.856 (0.040)	0.837 (0.008)	0.852 (0.041)	**0.874** (0.048)	0.832 (0.016)
F1	0.915 (0.028)	0.904 (0.011)	0.920 (0.023)	**0.925** (0.035)	0.901 (0.012)
AUC-PR	0.947 (0.017)	0.924 (0.045)	0.933 (0.051)	**0.956** (0.015)	0.894 (0.075)
AUC-ROC	0.990 (0.004)	0.986 (0.007)	0.988 (0.009)	**0.992** (0.003)	0.980 (0.015)
10 trees					
Accuracy	0.964 (0.006)	**0.966** (0.013)	0.964 (0.006)	**0.966** (0.004)	0.964 (0.006)
Recall	0.992 (0.018)	0.984 (0.036)	0.992 (0.018)	**1.0** (0.0)	0.992 (0.018)
Precision	0.867 (0.014)	**0.879** (0.030)	0.867 (0.014)	0.868 (0.014)	0.867 (0.0145)
F1	0.925 (0.013)	0.928 (0.029)	0.925 (0.013)	**0.929** (0.008)	0.925 (0.013)
AUC-PR	**0.993** (0.005)	0.979 (0.024)	0.991 (0.002)	0.986 (0.007)	0.989 (0.005)
AUC-ROC	**0.998** (0.002)	0.995 (0.005)	0.997 (0.001)	0.997 (0.002)	0.997 (0.001)
20 trees					
Accuracy	0.966 (0.004)	**0.968** (0.008)	0.959 (0.008)	0.964 (0.0)	0.964 (0.006)
Recall	**1.0** (0.0)	0.992 (0.018)	0.976 (0.036)	**1.0** (0.0)	0.992 (0.018)
Precision	0.868 (0.014)	**0.880** (0.018)	0.859 (0.005)	0.862 (0.0)	0.867 (0.014)
F1	0.929 (0.008)	**0.932** (0.017)	0.914 (0.018)	0.926 (0.0)	0.925 (0.013)
AUC-PR	0.992 (0.005)	**0.995** (0.004)	0.993 (0.005)	0.991 (0.002)	0.991 (0.009)
AUC-ROC	0.997 (0.002)	**0.999** (0.001)	0.998 (0.002)	0.997 (0.001)	0.998 (0.002)

The improvement of performance is not statistically significant contrary to our hypothesis and expectations. Most importantly, the improvement is not consistent through all settings of the number of iterations **K** and the number of folds **J** of the stacking method. There appears to be no discernable pattern in the results that would provide a deeper understanding of the effects of stacking. From tables 3 and 4, it can be seen that though the variation of the number of trees of RDN-Boost changes its learning ability, it still performs fairly well on both data sets. *The selected data sets may be not complex enough for the strong base learner RDN-Boost so that there is not much space left for the stacking method to improve*.

As analyzed in Fast et al. [7], on synthetic data, RDN gets lower learning bias and higher inference bias; stacked graphical learning gets higher learning bias and lower inference bias. Combining them together may not necessarily improve the overall performance since their under-performance on inference and learning respectively also decides the final performance. On real data, it shows that stacked graphical learning gets much lower inference variance than RDN which is not obvious in our experiments either. *Since the system was run only 5 times under each setting to get the mean and standard deviation, more runs may be required to get a precise evaluation of variance*.

5 Conclusion

Initial results clearly show that the approach of combining RDN-Boost with stacking method does not perform significantly better than the baseline i.e. RDN-Boost as expected. The reasons might be from the design of the approach or the complexity of data. The relational template is key part of the stacking method. A proper relational template could potentially facilitate the combination of RDN-Boost and stacking method. On a different note, the data sets in the section 4 may not necessarily be complex. Since RDN-Boost is a strong base learner, though the paper [6] mentioned that boosted approach introduces inference bias, the inference bias on those data sets might not be significant enough in performance. Investigating this deeper is an interesting direction.

Acknowledgments

SY, DSD and SN gratefully acknowledge DARPA Minerva award FA9550-19-1-0391. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA or the US government.

Broader Impact

Machine learning models encounter issues of bias, variance and noise while handling real problems. Advancing boosted-RDN with stacking method attempts to leverage strengths of different deliberate approaches to target multiple aspects of the error. An approach may be outstanding at one part of solving problems while the cooperation of different approaches makes the whole system more complete and robust.

References

[1] I. Alodah and J. Neville. Combining gradient boosting machines with collective inference to predict continuous values. *arXiv preprint arXiv:1607.00110*, 2016.

[2] J. Brophy and D. Lowd. Eggs: A flexible approach to relational modeling of social network spam. *arXiv preprint arXiv:2001.04909*, 2020.

[3] W. W. Cohen. Stacked sequential learning. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 2005.

[4] M. M. Deza and E. Deza. Encyclopedia of distances. In *Encyclopedia of distances*, pages 1–583. Springer, 2009.

[5] D. S. Dhami, A. Soni, D. Page, and S. Natarajan. Identifying parkinson's patients: A functional gradient boosting approach. In *AIME*, 2017.

[6] H. Eldardiry, J. Neville, and R. A. Rossi. Ensemble learning for relational data. *Journal of Machine Learning Research*, 21(49):1–37, 2020.

[7] A. Fast and D. Jensen. Why stacked models perform effective collective classification. In *2008 Eighth IEEE International Conference on Data Mining*, pages 785–790. IEEE, 2008.

[8] A. L. Hayes. srlearn: A python library for gradient-boosted statistical relational models. *arXiv preprint arXiv:1912.08198*, 2019.

[9] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1(Oct):49–75, 2000.

[10] D. Koller, N. Friedman, S. Džeroski, C. Sutton, A. McCallum, A. Pfeffer, P. Abbeel, M.-F. Wong, D. Heckerman, C. Meek, et al. *Introduction to statistical relational learning*. MIT press, 2007.

[11] Z. Kou and W. W. Cohen. Stacked graphical models for efficient inference in markov random fields. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 533–538. SIAM, 2007.

[12] A. Kuwadekar and J. Neville. Relational active learning for joint collective classification models. In *ICML*, 2011.

[13] C. Lu, J. Xu, and K. Zhang. On (d, 2)-dominating numbers of binary undirected de bruijn graphs. *Discrete applied mathematics*, 105(1-3):137–145, 2000.

[14] K. Marek, D. Jennings, S. Lasch, A. Siderowf, C. Tanner, T. Simuni, C. Coffey, K. Kieburtz, E. Flagg, S. Chowdhury, et al. The parkinson progression marker initiative (ppmi). *Progress in neurobiology*, 2011.

[15] S. Natarajan, T. Khot, K. Kersting, B. Gutmann, and J. Shavlik. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*, 86(1):25–56, 2012.

[16] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8(Mar):653–692, 2007.

[17] L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2016.

[18] I. Ravkic, J. Ramon, and J. Davis. Learning relational dependency networks in hybrid domains. *Machine Learning*, 2015.

[19] R. A. Santos, A. Paes, and G. Zaverucha. *Transfer learning by mapping and revising boosted relational dependency networks*. PhD thesis, Springer, 2019.

[20] O. Schulte, Z. Qian, A. E. Kirkpatrick, X. Yin, and Y. Sun. Fast learning of relational dependency networks. *Machine Learning*, 103(3):377–406, 2016.

[21] A. Soni, D. Viswanathan, J. Shavlik, and S. Natarajan. Learning relational dependency networks for relation extraction. In *ILP*, 2016.

[22] A. Srinivasan, R. D. King, S. Muggleton, and M. J. Sternberg. Carcinogenesis predictions using ilp. In *ILP*, 1997.

[23] D. H. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.